# 新架构ivr交互查询

## 1 需求描述

新架构ivr交互查询兼容老架构98,99查询被叫协议，并在此基础上新增了三种协议，分别是95,96,97。

通用的修改需求：

1. 新架构企业，通过ccgeid获取企业信息
2. 新架构企业，通过CR-Web提供的查询响应接口返回数据
3. 新架构企业，确定用户请求和用户响应格式

### 1.1 95(交互收键模式)

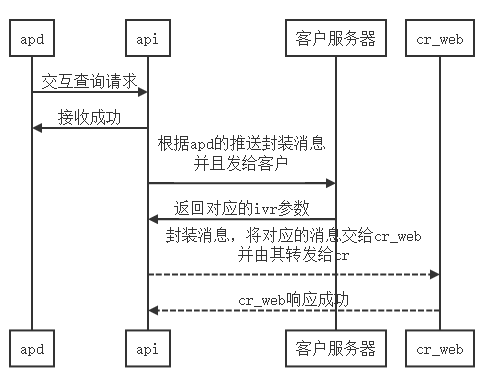
### 1.2 96(被叫查询模式)

### 1.3 97(通用交互模式)

### 1.4 98(AMNB或者AMXNB模式)

### 1.5 99(AMB或者AMXB模式)

## 2 交互流程



## 3 消息定义

### 3.1 apd->api

#### 3.1.1 95(交互收键模式)

{  
 "type" : 95,  
 "ccgeid" : 1576,  
 "callId" : "apixxxxxxxxxxxx",  
 "ccNumber" : "13260278209conf0\_1519439781636",  
 "callType" : 0,  
 "ivrFlowId" : 245,  
 "ivrQueryId" : 1,  
 "caller" : "13260278209",  
 "callerType" : 1,  
 "switchNumber" : "02566699794",  
 "called" : "1001",  
 "calledType" : 2,  
 "timestamp" : "1519439787",  
 "userQueryId" : "id\_0000001",  
 "inputKeys" : "1000",  
 "variables" : [  
 {"id\_number" : "110108198703127621" },  
 {"name" :"" },  
 {"address":"" }  
 ]  
}

#### 3.1.2 96(被叫查询模式)

{  
 "type" : 96,  
 "ccgeid" : 1576,  
 "callId" : "apixxxxxxxxxxxx",  
 "ccNumber" : "13260278209conf0\_1519439781636",  
 "callType" : 0,  
 "ivrFlowId" : 245,  
 "ivrQueryId" : 1,  
 "caller" : "13260278209",  
 "callerType" : 1,  
 "switchNumber" : "02566699794",  
 "called" : "1001",  
 "calledType" : 2,  
 "timestamp" : "1519439787",  
 "userQueryId" : "id\_0000001",  
 "inputKeys" : "1000",  
 "variables" : [  
 {"id\_number" : "110108198703127621" },  
 {"name": "" },  
 {"address": "" }  
 ]  
}

#### 3.1.3 97(通用交互模式)

{   
 "type" : 97,   
 "ccgeid" : 1576,   
 "callId" : "apixxxxxxxxxxxx",   
 "ccNumber" : "13260278209conf0\_1519439781636",   
 "callType" : 0,   
 "ivrFlowId" : 245,   
 "ivrQueryId" : 1,   
 "caller" : "13260278209",   
 "callerType" : 1,   
 "switchNumber" : "02566699794",   
 "called" : "1001",   
 "calledType" : 2,   
 "timestamp" : "1519439787",   
 "userQueryId" : "id\_0000001",   
 "inputKeys" : "1000",   
 "variables" : [   
 {"id\_number" : "110108198703127621" },   
 {"name" :"" },   
 {"address":"" }   
 ]   
}

#### 3.1.4 98(AMNB或者AMXNB模式)

{  
 "type" : 98,  
 "ccgeid" : 1576,  
 "callId" : "apixxxxxxxxxxxx",  
 "ccNumber" : "13260278209conf0\_1519439781636",  
 "ivrFlowId" : 34,  
 "ivrQueryId" : 35,  
 "switchNumber" : "02566699794",  
 "useNumber" : "02566699794",  
 "caller" : "13260278209",  
 "callerType" : 1,  
 "timestamp" : "1519439787",  
 "path" : "1000"  
}

#### 3.1.5 99(AMB或者AMXB模式)

{  
 "type" : 98,  
 "ccgeid" : 1576,  
 "callId" : "apixxxxxxxxxxxx",  
 "ccNumber" : "13260278209conf0\_1519439781636",  
 "ivrFlowId" : 34,  
 "ivrQueryId" : 35,  
 "switchNumber" : "02566699794",  
 "useNumber" : "02566699794",  
 "caller" : "13260278209",  
 "callerType" : 1,  
 "timestamp" : "1519439787",  
 "path" : "1000"  
}

### 3.2 api内部消息转换

根据前面的[需求描述](#header-n643)

1. 根据ccgeid从企业表中获取企业信息

根据ccgeid获取企业信息，我们不需要通过callId或者总机号才能获取到企业信息，需要注意的是。后面处理时还会尝试获取一次企业的详细信息，对于新架构，我们需要将企业的信息发送过去，以减少一次数据库操作（enterprise结构体的大小显然对于推送的消息队列来说过于庞大了，我还是妥协后面再一次获取企业的信息）。

1. 新架构的通用ivr交互新增了部分通用字段，有些字段api必须转发给客户。需要新增一个结构体用来存储这些信息

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 通话类型 | IVR流程 | callType | caller | callerType | called | calledType |
| 呼入 | 正常流程 | 5 | 客户号码 | 1 | - | - |
| 呼入 | 子IVR流程 | 5 | 客户号码 | 1 | 坐席话机号 | 1/2 |
| 呼出 | 子IVR流程 | 0或7 | 坐席话机号 | 1/2 | 客户号码 | 1 |
| 语音通知 | 语音通知流程 | 3 | 总机号码 | 1 | 客户号码 | 1 |

typedef enum  
{  
 IVR\_CALLER\_AND\_CALLLED\_TYPE\_OUTLINE = 1,  
 IVR\_CALLER\_AND\_CALLLED\_TYPE\_INLINE = 2，  
}IVRCallerAndCalledType;  
  
typedef struct \_\_push\_post\_data\_ivr\_moudle\_\_  
{  
 DB\_CALL\_RECORD\_CALL\_TYPE calltype; //联合calltype caller called，可以得知在此次ivr请求中，哪个是客户，哪个是坐席  
 /\*  
 | 通话类型 | IVR流程 | callType | caller | callerType | called | calledType |  
 | -------- | ------------ | -------- | ---------- | ---------- | ---------- | ---------- |  
 | 呼入 | 正常流程 | 5 | 客户号码 | 1 | - | - |  
 | 呼入 | 子IVR流程 | 5 | 客户号码 | 1 | 坐席话机号 | 1/2 |  
 | 呼出 | 子IVR流程 | 0或7 | 坐席话机号 | 1/2 | 客户号码 | 1 |  
 | 语音通知 | 语音通知流程 | 3 | 总机号码 | 1 | 客户号码 | 1 |  
 \*/  
 unsigned long ivrFlowId; //ivr流程id  
 unsigned long ivrQueryId; //ivr查询节点id  
 char userQueryId[64]; //用于向客户服务器确定id对应下一步的操作是什么  
 char inputKeys[64]; //推送上次输入的按键  
 char \*variables;  
 /\*  
 用户自定义查询请求变量值集合，是{"variables" : [ {"id\_number" : "110108198703127621"}, {:"name" :""}, {"address":""} ]}，  
 存入对空间中，用完释放  
 \*/  
}ModCallPushPostDataIVRMoudle;  
  
typedef struct \_\_push\_post\_data\_\_  
{  
 unsigned char callId[CALL\_RECORD\_CALLID\_MAX\_LEN];  
 unsigned char ccNumber[CALL\_RECORD\_CALLID\_MAX\_LEN];  
 unsigned char caller[PHONE\_NUMBER\_MAX\_LEN];  
 unsigned char called[PHONE\_NUMBER\_MAX\_LEN];  
 int xferTimes;  
 BOOL extCaller;  
 BOOL extCalled;  
 BOOL isCaller;  
 DB\_CALL\_RECORD\_CALL\_TYPE type;  
 DB\_CALL\_RECORD\_STATUS status;  
 unsigned char useNumber[PHONE\_NUMBER\_MAX\_LEN];  
 unsigned char switchNumber[PHONE\_NUMBER\_MAX\_LEN];  
 unsigned char subNumber[PHONE\_NUMBER\_MAX\_LEN];  
 unsigned char virtNumber[PHONE\_NUMBER\_MAX\_LEN];  
 unsigned long enterpriseId;  
 unsigned long ccgeid;  
 unsigned long ringTime;  
 unsigned long startTime;  
 unsigned long endTime;  
 unsigned long timestamp;  
 unsigned long duration;  
 unsigned long reason;  
 unsigned long gid;  
 unsigned long pbxCallLogId;  
 char feedback[CALL\_RECORD\_FEEDBACK\_MAX\_LEN];  
 ModCallPushPostDataIVRMoudle \*ivr\_argv; //新架构通用ivr扩展字段  
  
 BOOL realtimeData;  
 BOOL isCheckData;  
 BOOL isCommonEnterprise;  
 BOOL hangup2calling;  
 int index;  
 int lwpid;  
 int mes\_type;  
 int failed\_delay\_time;  
 unsigned char path[CALL\_RECORD\_FEEDBACK\_MAX\_LEN]; //99协议 ，按键 (可能包含二级按键，比如:2-9)  
  
 unsigned long app\_id;  
 unsigned long provinceId;  
 char number[USER\_NUMBER\_MAX\_LEN];  
 char mobile[PHONE\_NUMBER\_MAX\_LEN];  
 char destNumber[USER\_NUMBER\_MAX\_LEN];  
 unsigned long ngnReason; //201708-N02细化通话失败和挂断原因  
 char batchCallId[CALL\_RECORD\_CALLID\_MAX\_LEN];  
 char batchCallUserData[USER\_DATA\_MAX\_LEN];  
 char batchCallTaskId[BATCH\_TASK\_ID\_MAX\_LEN];  
} ModCallPushPostData;

### 3.2 api->用户服务器

请求格式和用户响应格式由客户进行配置，可以是json或者xml

#### 3.2.1 请求客户

1. 99/98 99和98向客户的请求信息保持不变

{   
 'appId':'b23abb6d451346efa13370172d1921ef',   
 'callId':'api1234059445aDbbJxIdbT',   
 'accountSid':'c5dc4b87f33ef2ef37c8e974793ad8e5',   
 'caller':'18769874345',   
 'path':'1-2',   
 'type':0,   
 'callType':99,   
 'useNumber':'02566687987',   
 'switchNumber' : "02566699794",   
 'userData':"FE87D3"   
}

1. 95,96和97消息需要使用新架构新的消息格式

新架构的查询场景不再限于呼入场景，api客户可能需要知道主叫或被叫是坐席还是客户，callType需要传给客户

{   
 "type" : 95/96/97,   
 //"initialCallType": 1,   
 //一开始发起的通话类型   
 "callType" : 1,   
 "accountSid": "c5dc4b87f33ef2ef37c8e974793ad8e5",   
 "subAccountSid": "c5dc4b87f33ef2ef37c8e974793ad8e5",   
 "callId" : "apixxxxxxxxxxxx",   
 "caller" : "13260278209",   
 "called" : "1001",   
 "useNumber" : "02566699794",   
 "userQueryId" : "id\_0000001",   
 "inputKeys" : "1000",   
 "variables" : [   
 {"id\_number" : "110108198703127621" },   
 {"name" :"" },   
 {"address":"" }   
 ],   
}   
说明：在上例中，有3个全局变量：id\_number、name和address，id\_number已经赋值，name和address未赋值，需要用户服务器返回，并在IVR其它节点中引用。

#### 3.2.2 客户响应

1. 98/99返回数据

{   
 //0:允许通话，否则失败   
 "retcode": 0,   
 //返回被叫   
 "action": 0,   
 "called": "1\*\*\*\*211",   
 "number": "1002",   
 "workNumber": "test",   
 "waitTime": 20,   
 "outNumber":"02566687\*\*1",   
 "reason": "原因描述",   
 "userdata": "用户数据"   
}   
//其中number>workNumber>called,这三个都是被叫选项，三者选一。被叫如果传多个，会按顺序拨打直至拨打完毕

1. 95/96/97

{   
 "retcode": 0,   
 //客户响应数据都将透传给cr，具体含义见api->cr\_web的定义   
 //可以为空，也可以和请求保重的userQueryId相同，也可以不同   
 //不为空时，下次的交互查询节点会直接赋值userQueryId   
 "userQueryId":"id\_0000002",   
   
 //variables是请求包中variables的子集，需要赋值的参数必须要通过这个列表中返回   
 "variables": [   
 {"id\_number" : "110108198703127621" },   
 {"name" :"" },   
 {"address":"" }   
 ],   
 //虚拟键值，交互收键模式时的响应参数   
 "virtualKey":"1111",   
 //用户返回的下一步参数，这个数据应当由api透传给cr，现cr已经定义好响应数据，所以和需求文档中的格式略有不同，具体数据参数见api->cr\_web的消息定义   
 "nextAction" : {   
 "action" : 1,   
 "paras" : {   
 "voiceId" : "播放语音文件id",   
 "voiceName" : "播放语音文件唯一名称",   
 "allowBreak" : "是否允许打断: 0-不允许 1-允许"   
 }   
 },   
 "userData":"FE87D3"   
}

### 3.3 api->cr\_web

新架构cr采用统一的json格式，所以不管是95、96、97、98还是99，api需要将客户返回的消息转换成cr需要的格式

以下是cr根据action定义的响应消息内容，95、96、98、99需要根据这些重新生成给cr的响应，然后通过cr\_web透传给cr

为方便路由，要求将ccgeid和cc\_number作为头域参数

action

typedef enum  
{  
 0, invalid  
 1, 放音响应  
 2, 放音按键响应  
 3, 转技能组响应  
 4, 转坐席响应  
 5, 转外线响应  
 6, 转其他IVR流程响应  
 7, 流程结束响应  
}IvrActionType;

#### 3.3.1 交互收键模式响应消息

{   
 "rspCode" : 0,   
 “type” : “95”,   
 “ccgeid” : “124”,   
 "userQueryId" : "id\_0000001",   
 “ccNumber” : “11222”,   
 "virtualKey" : "5",   
 "variables" : [   
 { "id\_number" : "110108198703127621" },   
 { "name" : "张三" },   
 {"address" : "江苏省南京市江宁区" }   
 ]   
 "reason" : "test",   
 "userdata" : "test"   
}

#### 3.3.2 被叫查询和通用交互模式响应消息

##### 3.3.2.1 放音响应

json to cr

{   
 "rspCode" : 0,   
 "ccgeid" : 123,   
 "ccNumber" : "21212",   
 "userQueryId" : "id\_0000001",   
 "variables" : [   
 { "id\_number" : "110108198703127621" },   
 { "name" :"张三" },   
 { "address":"江苏省南京市江宁区" }   
 ],   
 "nextAction" : {   
 "action" : 1,   
 "paras" : {   
 //下面四个参数选择其中之一   
 "voiceId" : "播放语音文件id",   
 "voiceName" : "播放语音文件唯一名称",   
 "voiceTempId": "播放语音模板id",   
 "voiceTempName": "播放语音模板名称",   
 //语音模板参数，多个参数见用英文","号隔开，当选择voiceTempId或者voiceTempName时需要传入   
 "voiceTempParams": "语音模板参数",   
   
 "allowBreak" : "是否允许打断: 0-不允许 1-允许"   
 }   
 },   
 "reason" : "test",   
 "userdata" : "test"   
}

##### 3.3.2.2 放音收键响应

json to cr

{   
 "rspCode" : 0,   
 “ccgeid” : “123”,   
 "ccNumber" : "21212",   
 "userQueryId" : "id\_0000001",   
 "variables" : [   
 { "id\_number" : "110108198703127621" },   
 { "name" :"张三" },   
 { "address":"江苏省南京市江宁区" }   
 ],   
 "nextAction" : {   
 "action" : 2,   
 "paras" : {   
 //下面四个参数选择其中之一   
 "voiceId" : "播放语音文件id",   
 "voiceName" : "播放语音文件唯一名称",   
 "voiceTempId": "播放语音模板id",   
 "voiceTempName": "播放语音模板名称",   
 //语音模板参数，多个参数见用英文","号隔开，当选择voiceTempId或者voiceTempName时需要传入   
 "voiceTempParams": "语音模板参数",   
   
 "allowBreak" : "是否允许打断: 0-不允许 1-允许",   
 "getKeyNumber" : "获取按键位数",   
 "getKeyTimeout" : "收键超时时间",   
 "endWithHashKey" : "是否以#号键结束, 0-不是, 1-是"   
 }   
 },   
 "reason" : "test",   
 "userdata" : "test"   
}

##### 3.3.2.3 转技能组响应

json to cr

{   
 "rspCode" : 0,   
 “ccgeid” : “123”,   
 “ccNumber” : :21212”,   
 "userQueryId" : "id\_0000001",   
 "variables" : [   
 { "id\_number" : "110108198703127621" },   
 { "name" :"张三" },   
 { "address":"江苏省南京市江宁区" }   
 ]   
 "nextAction" : {   
 "action" : 3,   
 "paras" : {   
 "acdId" : "技能组id",   
 "acdName" : "技能组名称",   
 "useAcdValue" : "0-不使用技能组配置 1-使用技能组配置",   
 "queueTime" : "排队超时时长",   
 "switchTimes" : "坐席流转次数",   
 "ringTimeout" : "坐席振铃超时时长",   
 "customerMemory" : "0-不记忆 1-优先熟客记忆 2-强制熟客记忆"   
 }   
 }   
 "reason" : "test",   
 "userdata" : "test"   
}

##### 3.3.2.4 转座席响应

json to cr

{   
 "rspCode" : 0,   
 “ccgeid” : “123”,   
 “ccNumber” : :21212”,   
 "userQueryId" : "id\_0000001",   
 "variables" : [   
 { "id\_number" : "110108198703127621" },   
 { "name" :"张三" },   
 { "address":"江苏省南京市江宁区" }   
 ]   
 "nextAction" : {   
 "action" : 4,   
 "paras" : {   
 "workNumber" : "1001,1002,1003",   
 "number" : "1001,1002,1003",   
 "queueTime" : "坐席忙时排队时长",   
 "ringTimeout" : "多坐席情况下，坐席振铃超时时长"   
 }   
 }   
 "reason" : "test",   
 "userdata" : "test"   
}

##### 3.3.2.5 转外线响应

json to cr

{   
 "rspCode" : 0,   
 “ccgeid” : “123”,   
 "ccNumber" : :"21212”,   
 "userQueryId" : "id\_0000001",   
 "variables" : [   
 { "id\_number" : "110108198703127621" },   
 { "name" :"张三" },   
 { "address":"江苏省南京市江宁区" }   
 ]   
 "nextAction" : {   
 "action" : 5,   
 "paras" : {   
 "called" : "外线被叫号码",   
 "outNumber" : "呼出总机号码"   
 }   
 }   
 "reason" : "test",   
 "userdata" : "test"   
}

##### 3.3.2.6 转其他IVR流程响应

json to cr

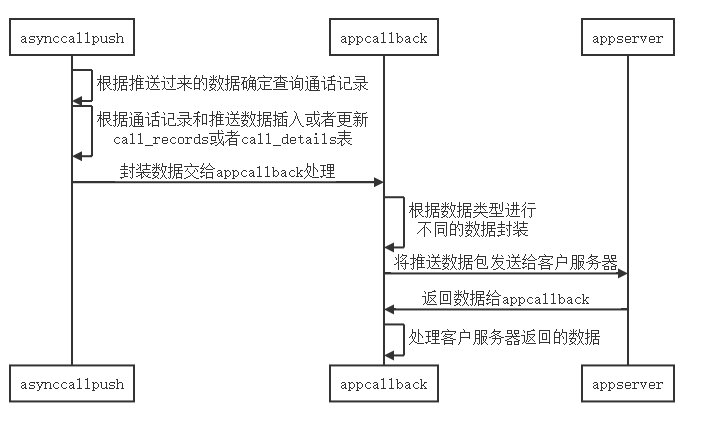
{   
 "rspCode" : 0,   
 "ccgeid” : “123”,   
 “ccNumber” : "21212”,   
 "userQueryId" : "id\_0000001",   
 "variables" : [   
 { "id\_number" : "110108198703127621" },   
 { "name" :"张三" },   
 { "address":"江苏省南京市江宁区" }   
 ]   
 "nextAction" : {   
 "action" : 6,   
 "paras" : {   
 "ivrFlowId" : "IVR流程id",   
 "ivrFlowName" : "IVR流程名称"   
 }   
 }   
 "reason" : "test",   
 "userdata" : "test"   
}

##### 3.3.2.7 流程结束响应

{   
 "rspCode" : 0,   
 “ccgeid” : “123”,   
 “ccNumber” : “21212”,   
 "userQueryId" : "id\_0000001",   
 "variables" : [   
 { "id\_number" : "110108198703127621" },   
 { "name" :"张三" },   
 { "address":"江苏省南京市江宁区" }   
 ]   
 "nextAction" : {   
 "action" : 7   
 }   
 "reason" : "test",   
 "userdata" : "test"   
}

## 4 处理流程

* api callpush模块时序图



经过考察我们主要的修改是：

1. 发向appcallback队列的数据会进行扩展，存储ivr扩展信息。
2. 对于非呼入的数据，不能更新通话记录
3. 回调接口扩展，新建新的协议和对应的回调函数
4. 查询结果封装并返回，通过cr\_web接口

### 4.1 95(交互收键模式)

95和97模式与以往的通话类请求有所不同，其中的数据不应该入库。api对于这两种数据只做透传处理。唯一记录的地方是api给客户的推送表，但其中并不会记录推送内容。

### 4.2 96(被叫查询模式)

被叫查询api会直接根据消息数据入库，和之前的处理方式保持一致即可。

### 4.3 97(通用交互模式)

95和97模式与以往的通话类请求有所不同，其中的数据不应该入库。api对于这两种数据只做透传处理。唯一记录的地方是api给客户的推送表，但其中并不会记录推送内容。

### 4.4 98(AMNB或者AMXNB模式)

被叫查询api会直接根据消息数据入库，和之前的处理方式保持一致即可。

### 4.5 99(AMB或者AMXB模式)

被叫查询api会直接根据消息数据入库，和之前的处理方式保持一致即可。

## 5 涉及代码