新架构ivr交互查询

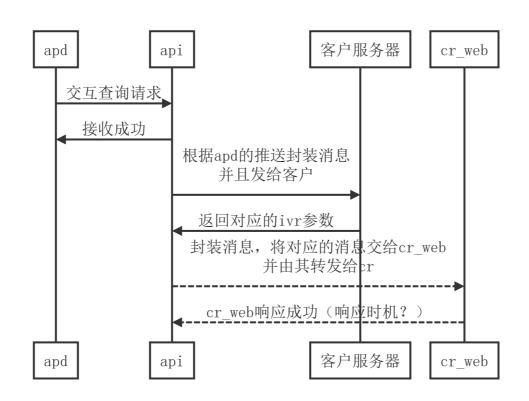
1需求描述

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)

```
1
    {
 2
        "type": 95,
 3
        "ccgeid": 1576,
        "callId": "apixxxxxxxxxxxxx",
 4
        "ccNumber": "13260278209conf0_1519439781636",
        "callType" : 0,
 6
 7
        "ivrFlowId" : 245,
        "ivrQueryId" : 1,
 8
 9
        "caller": "13260278209",
10
        "callerType" : 1,
        "switchNumber" : "02566699794",
11
12
        "called" : "1001",
13
        "calledType" : 2,
        "timestamp": "1519439787",
15
        "userQueryId" : "id_0000001",
        "inputKeys" : "1000",
16
        "variables" : [
17
            {"id_number" : "110108198703127621" },
18
19
            {:"name" :"" },
20
            {"address":"" }
        ]
21
    }
22
23
```

3.1.2 96(被叫查询模式)

```
1
    {
 2
        "type": 96,
 3
        "ccgeid": 1576,
 4
        "callId": "apixxxxxxxxxxxxx",
 5
        "ccNumber": "13260278209conf0_1519439781636",
 6
        "callType" : 0,
        "ivrFlowId" : 245,
 7
        "ivrQueryId" : 1,
 9
        "caller": "13260278209",
        "callerType" : 1,
10
11
        "switchNumber": "02566699794",
        "called" : "1001",
12
        "calledType" : 2,
13
14
        "timestamp" : "1519439787",
        "userQueryId" : "id_0000001",
15
        "inputKeys" : "1000",
16
        "variables" : [
17
            {"id_number" : "110108198703127621" },
18
19
            {:"name" :"" },
20
            {"address":"" }
21
        ]
22 }
```

3.1.3 97(通用交互模式)

```
1
 2
        "type" : 97,
 3
        "ccgeid": 1576,
        "callId": "apixxxxxxxxxxxxx",
 4
        "ccNumber": "13260278209conf0_1519439781636",
 5
 6
        "callType": 0,
        "ivrFlowId" : 245,
 7
        "ivrQueryId" : 1,
 8
9
        "caller": "13260278209",
10
        "callerType" : 1,
        "switchNumber" : "02566699794",
11
        "called" : "1001",
12
13
        "calledType" : 2,
14
        "timestamp": "1519439787",
        "userQueryId" : "id_0000001",
15
        "inputKeys" : "1000",
16
        "variables" : [
17
18
            {"id_number" : "110108198703127621" },
19
            {:"name" :"" },
            {"address":"" }
20
21
        ]
22 }
```

3.1.4 98(AMNB或者AMXNB模式)

```
1 | {
 2
        "type": 98,
 3
        "ccgeid": 1576,
 4
        "ccNumber": "13260278209conf0_1519439781636",
        "ivrFlowId" : 34,
 5
        "ivrQueryId" : 35,
 6
 7
        "switchNumber": "02566699794",
        "useNumber": "02566699794",
9
        "caller": "13260278209",
        "callerType" : 1,
10
11
        "timestamp": "1519439787",
        "path" : "1000"
12
13 }
```

3.1.5 99(AMB或者AMXB模式)

```
1
   {
 2
        "type": 98,
 3
        "ccgeid" : 1576,
        "ccNumber" : "13260278209conf0_1519439781636",
 4
 5
        "ivrFlowId" : 34,
        "ivrQueryId" : 35,
 6
        "switchNumber": "02566699794",
 7
 8
        "useNumber": "02566699794",
        "caller": "13260278209",
9
10
        "callerType" : 1,
        "timestamp" : "1519439787",
11
        "path" : "1000"
12
13
   }
```

3.2 api内部消息转换

根据前面的需求描述

1. 我们需要在某些结构体中新增ccgeid字段

新增ccgeid这个参数的意图应该是利用ccgeid代替enterpriseld和provinceld做hash运算确定处理线程,因为provinceld字段的获取需要查找数据库。但是事实是,入库日志数据库时目前enterpriseld和provinceld是必须要填写的。这个操作在apache模块进行,同样的还有通知给checkback模块的消息。可能需要在日志数据库中新增一个索引字段ccgeid,或者调整这些操作的位置。

然后我们就可以定义结构体信息:

```
//日志数据库内容,利用ccgeid代替enterpriseId和provinceId入库
 2
    typedef struct _emicalldev_db_callpushpost_data_
 3
                       id;
 4
        unsigned long
 5
        BOOL
                       isCommonEnterprise;
        unsigned long app_id;
 6
 7
        unsigned long provinceId;
 8
        unsigned long enterpriseId;
 9
        unsigned long
                       ccgeid; //新架构新增ccgeid字段,用以标识企业
        unsigned long callLogId;
10
11
        int
                       type;
12
        unsigned char callid[CALL_RECORD_CALLID_MAX_LEN];
13
        unsigned char
                       switchNumber[PHONE_NUMBER_MAX_LEN];
14
        DB_CALLPUSH_POSTDATA_STATUS
                                      status;
15
        unsigned char
                       post_data[HTTP_CONTENT_BUFFER_SIZE];
16
        unsigned long createTime;
17
        unsigned long
                       updateTime;
18
    } EMICALLDEV_DB_CALLPUSH_POST_DATA;
```

```
//推送内容,利用ccgeid代替enterpriseId和provinceId查询企业的详细信息
 2
    typedef struct __push_post_data__
 3
 4
        unsigned char
                                     callid[CALL_RECORD_CALLID_MAX_LEN];
 5
        unsigned char
                                     ccNumber[CALL_RECORD_CALLID_MAX_LEN];
        unsigned char
                                     caller[PHONE_NUMBER_MAX_LEN];
 6
        unsigned char
 7
                                     called[PHONE_NUMBER_MAX_LEN];
 8
        int
                                     xferTimes;
 9
        BOOL
                                     extCaller;
10
        BOOL
                                     extCalled;
11
        BOOL
                                     isCaller;
12
        DB_CALL_RECORD_CALL_TYPE
                                     type;
13
        DB_CALL_RECORD_STATUS
                                     status;
14
        unsigned char
                                     useNumber[PHONE_NUMBER_MAX_LEN];
15
        unsigned char
                                     switchNumber[PHONE_NUMBER_MAX_LEN];
16
        unsigned char
                                     subNumber[PHONE_NUMBER_MAX_LEN];
17
        unsigned char
                                     virtNumber[PHONE_NUMBER_MAX_LEN];
        unsigned long
18
                                     enterpriseId;
19
        unsigned long
                                     ccgeid;
                                                 //新架构新增ccgeid字段,用以标识企业
20
        unsigned long
                                     ringTime;
21
        unsigned long
                                     startTime;
22
        unsigned long
                                     endTime;
        unsigned long
23
                                     timestamp;
24
        unsigned long
                                     duration;
25
        unsigned long
                                     reason;
```

```
26
        unsigned long
                                     gid;
27
        unsigned long
                                     pbxCallLogId;
                                     feedback[CALL_RECORD_FEEDBACK_MAX_LEN];
28
        char
29
30
        BOOL
                                     realtimeData;
31
        BOOL
                                     isCheckData;
32
        BOOL
                                     isCommonEnterprise;
33
        BOOL
                                     hangup2calling;
34
        int
                                     index;
35
        int
                                     lwpid;
        int
36
                                     mes_type;
37
        int
                                     failed_delay_time;
                                                                            //99协
38
        unsigned char
                                     path[CALL_RECORD_FEEDBACK_MAX_LEN];
    议, 按键 (可能包含二级按键, 比如:2-9)
39
        unsigned long
40
                                     app_id;
41
        unsigned long
                                     provinceId;
                                     number[USER_NUMBER_MAX_LEN];
42
        char
43
        char
                                     mobile[PHONE_NUMBER_MAX_LEN];
44
        char
                                     destNumber[USER_NUMBER_MAX_LEN];
        unsigned long
45
                                     ngnReason;
     //201708-N02细化通话失败和挂断原因
46
        char
                                     batchCallId[CALL_RECORD_CALLID_MAX_LEN];
47
        char
                                     batchCalluserData[USER_DATA_MAX_LEN];
        char
                                     batchCallTaskId[BATCH_TASK_ID_MAX_LEN];
49
    } ModCallPushPostData;
```

2. 新架构的通用ivr交互新增了部分通用字段,这些字段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

```
1
    typedef enum
2
       IVR_CALLER_AND_CALLLED_TYPE_OUTLINE = 1,
3
4
       IVR_CALLER_AND_CALLLED_TYPE_INLINE = 2,
5
   }IVRCallerAndCalledType;
6
7
   typedef struct __push_post_data_ivr_moudle__
8
9
                                               //标记主叫是内线还是外线
       IVRCallerAndCalledType
                                callerType;
10
       IVRCallerAndCalledType
                                calledType;
                                               //标记被叫是内线还是外线
11
       DB_CALL_RECORD_CALL_TYPE
                                  calltype;
                                               //联合calltype callertype
    calledtype,可以得知在此次ivr请求中,哪个是客户,哪个是坐席
```

```
12
       | 通话类型 | IVR流程 | callType | caller | callerType | called
13
      | calledType |
     14
    ---- | ----- |
                                     | 客户号码
15
      | 呼入
               | 正常流程
                             | 5
                                                 | 1
                                                             | -
               1
16
      | 呼入
               | 子IVR流程
                            l 5
                                       | 客户号码
                                                | 1
                                                             1 坐席话机号
    | 1/2
              | 呼出
17
               | 子IVR流程 | O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 |
18
   1
19
20
       unsigned long ivrFlowId;
                                //ivr流程id
21
       unsigned long ivrQueryId; //ivr查询节点id
22
       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
23
       char
                   inputKeys[64]; //推送上次输入的按键
24
                   *variables;
       char
25
       /*
       用户自定义查询请求变量值集合,是{"variables": [{"id_number":
26
    "110108198703127621"}, {:"name" :""}, {"address":""} ]},
27
       存入对空间中, 用完释放
       */
28
29
   }ModCallPushPostDataIVRMoudle;
30
31
   typedef struct __push_post_data__
   {
32
33
       unsigned char
                                callid[CALL_RECORD_CALLID_MAX_LEN];
34
       unsigned char
                                ccNumber[CALL_RECORD_CALLID_MAX_LEN];
       unsigned char
35
                                caller[PHONE_NUMBER_MAX_LEN];
36
       unsigned char
                                called[PHONE_NUMBER_MAX_LEN];
37
       int
                                xferTimes:
38
       BOOL
                                extCaller:
39
       BOOL
                                extCalled;
40
       BOOL
                                isCaller;
41
       DB_CALL_RECORD_CALL_TYPE
                                type;
42
       DB_CALL_RECORD_STATUS
                                status;
43
       unsigned char
                                useNumber[PHONE_NUMBER_MAX_LEN];
44
       unsigned char
                                switchNumber[PHONE_NUMBER_MAX_LEN];
45
       unsigned char
                                subNumber[PHONE_NUMBER_MAX_LEN];
46
       unsigned char
                                virtNumber[PHONE_NUMBER_MAX_LEN];
47
       unsigned long
                                enterpriseId;
48
       unsigned long
                                ccgeid;
49
       unsigned long
                                ringTime;
       unsigned long
50
                                startTime;
51
       unsigned long
                                endTime;
52
       unsigned long
                                timestamp;
53
       unsigned long
                                duration;
54
       unsigned long
                                reason;
       unsigned long
55
                                qid;
56
       unsigned long
                                pbxCallLogId;
                                feedback[CALL_RECORD_FEEDBACK_MAX_LEN];
57
       char
58
                                    *ivr_argv; //新架构通用ivr扩展字段
       ModCallPushPostDataIVRMoudle
59
60
       BOOL
                                realtimeData:
61
       BOOL
                                isCheckData;
62
       BOOL
                                isCommonEnterprise;
```

```
63
        BOOL
                                    hangup2calling;
64
        int
                                    index;
65
        int
                                    lwpid;
66
        int
                                    mes_type;
67
        int
                                    failed_delay_time;
68
        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)
69
70
        unsigned long
                                    app_id;
71
        unsigned long
                                    provinceId;
72
        char
                                    number[USER_NUMBER_MAX_LEN];
73
        char
                                    mobile[PHONE_NUMBER_MAX_LEN];
74
        char
                                    destNumber[USER_NUMBER_MAX_LEN];
75
        unsigned long
                                    ngnReason;
     //201708-N02细化通话失败和挂断原因
76
        char
                                    batchCallId[CALL_RECORD_CALLID_MAX_LEN];
77
        char
                                    batchCalluserData[USER_DATA_MAX_LEN];
78
        char
                                    batchCallTaskId[BATCH_TASK_ID_MAX_LEN];
79
   } ModCallPushPostData;
```

3.2 api->用户服务器

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

```
1
    {
 2
         'appId': 'b23abb6d451346efa13370172d1921ef',
 3
         'callId': 'api1234059445aDbbJxIdbT',
        'accountSid':'c5dc4b87f33ef2ef37c8e974793ad8e5',
 4
 5
        'caller':'18769874345',
        'path': '1-2',
 6
 7
         'callType':99,
 8
         'type':0,
 9
         'useNumber':'02566687987',
         'switchNumber': "02566699794",
10
        'userData':FE87D3
11
    }
12
```

2. 95,96和97消息需要使用新架构新的消息格式

```
1
    {
 2
        "type": 95/96/97,
        "ccgeid": 1576,
 3
 4
        "callId" : "apixxxxxxxxxxxxx",
 5
        "ccNumber": "13260278209conf0_1519439781636",
 6
        "callType" : 0,
        "ivrFlowId" : 245,
 7
        "ivrQueryId" : 1,
 8
 9
        "caller": "13260278209",
10
        "callerType" : 1,
        "switchNumber" : "02566699794",
11
        "called" : "1001",
12
        "calledType" : 2,
13
```

```
"timestamp" : "1519439787",
14
15
       "userQueryId" : "id_0000001",
16
       "inputKeys" : "1000",
       "variables" : [
17
           {"id_number" : "110108198703127621" },
18
           {"name" :"" },
19
           {"address":"" }
20
21
       ]
22
   }
23
   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

actionType

```
1 typedef enum
2
   {
3
     0, invalid
      1, 放音响应
4
5
     2, 放音按键响应
     3, 转技能组响应
6
      4, 转坐席响应
7
8
     5, 转外线响应
      6, 转其他IVR流程响应
9
10
      7, 流程结束响应
11 }IvrActionType;
```

3.2.2.1 放音响应

json to cr

```
1 {
 2
       "rspCode" : 0,
 3
        "ccgeid": "123",
        "ccNumber" : :21212",
 4
        "userQueryId" : "id_0000001",
 5
 6
        "variables" : [
 7
           { "id_number" : "110108198703127621" },
           { "name":"张三"},
 8
           { "address":"江苏省南京市江宁区" }
9
        1
10
        "nextAction" : {
11
12
           "action" : 1,
           "paras" : {
13
               "voiceId": "播放语音文件id",
14
               "voiceName": "播放语音文件唯一名称",
15
               "allowBreak": "是否允许打断: 0-不允许 1-允许"
16
17
           }
18
        "reason" : "test",
19
       "userdata" : "test"
20
21 }
```

3.2.2.2 放音收键响应

```
1
   {
        "rspCode" : 0,
 2
        "ccgeid": "123",
 3
 4
        "ccNumber" : :21212",
 5
        "userQueryId" : "id_0000001",
 6
        "variables" : [
           { "id_number" : "110108198703127621" },
 7
           { "name" :"张三" },
 8
9
           { "address":"江苏省南京市江宁区" }
10
        "nextAction" : {
11
           "action" : 2.
12
13
           "paras" : {
               "voiceId": "播放语音文件id",
14
               "voiceName": "播放语音文件唯一名称",
15
               "allowBreak": "是否允许打断: 0-不允许 1-允许",
16
               "getKeyNumber": "获取按键位数",
17
               "getKeyTimeout": "收键超时时间",
18
19
               "endwithHashKey": "是否以#号键结束, 0-不是, 1-是"
           }
20
21
        "reason" : "test",
22
23
        "userdata" : "test"
24
   }
25
```

3.2.2.3 转技能组响应

json to cr

```
1
   {
 2
       "rspCode" : 0,
 3
       "ccgeid": "123",
       "ccNumber" : :21212",
 4
       "userQueryId" : "id_0000001",
 5
       "variables" : [
 6
 7
           { "id_number" : "110108198703127621" },
 8
           { "name":"张三"},
 9
           { "address":"江苏省南京市江宁区" }
10
       "nextAction" : {
11
           "action" : 3,
12
13
           "paras" : {
14
               "acdId": "技能组id",
               "acdName": "技能组名称",
15
               "useAcdValue": "0-不使用技能组配置 1-使用技能组配置",
16
               "queueTime": "排队超时时长",
17
               "switchTimes": "坐席流转次数",
18
               "ringTimeout": "坐席振铃超时时长",
19
               "customerMemory": "0-不记忆 1-优先熟客记忆 2-强制熟客记忆"
20
           }
21
22
        "reason" : "test",
23
       "userdata" : "test"
24
25
   }
```

3.2.2.4 转座席响应

json to cr

```
1
    {
 2
        "rspCode" : 0,
        "ccgeid": "123",
 3
        "ccNumber" : :21212",
 4
 5
        "userQueryId" : "id_0000001",
        "variables" : [
 6
 7
            { "id_number" : "110108198703127621" },
            { "name":"张三"},
 8
            { "address":"江苏省南京市江宁区" }
 9
10
11
        "nextAction" : {
12
            "action": 4,
13
            "paras" : {
                "workNumber" : "1001,1002,1003",
14
15
                "number" : "1001,1002,1003",
                "queueTime": "坐席忙时排队时长",
16
17
                "ringTimeout": "多坐席情况下,坐席振铃超时时长"
18
            }
19
        "reason" : "test",
20
        "userdata" : "test"
21
22 }
```

3.2.2.5 转外线响应

json to cr

```
1
    {
 2
        "rspCode" : 0,
 3
        "ccgeid": "123",
        "ccNumber" : :"21212",
 4
        "userQueryId" : "id_0000001",
 5
        "variables" : [
 6
            { "id_number" : "110108198703127621" },
 7
            { "name":"张三"},
 8
            { "address":"江苏省南京市江宁区" }
 9
10
        "nextAction" : {
11
12
            "action" : 5,
            "paras" : {
13
                "called": "外线被叫号码",
14
                "outNumber": "呼出总机号码"
15
16
            }
17
        "reason": "test",
18
        "userdata" : "test"
19
20
    }
21
```

3.2.2.6 转其他IVR流程响应

```
1 | {
        "rspCode" : 0,
 2
        "ccgeid": "123",
 3
 4
        "ccNumber": "21212",
 5
        "userQueryId" : "id_0000001",
        "variables" : [
 6
 7
           { "id_number" : "110108198703127621" },
            { "name" :"张三" },
9
            { "address":"江苏省南京市江宁区" }
10
11
        "nextAction" : {
           "action" : 6,
12
13
            "paras" : {
                "ivrFlowId": "IVR流程id",
14
               "ivrFlowName": "IVR流程名称"
15
            }
16
17
        }
18
        "reason" : "test",
19
        "userdata" : "test"
20 }
```

3.2.2.7 流程结束响应

```
1 {
 2
        "rspCode" : 0,
3
        "ccgeid": "123",
 4
        "ccNumber": "21212",
 5
       "userQueryId" : "id_0000001",
        "variables" : [
 6
           { "id_number" : "110108198703127621" },
 7
            { "name":"张三"},
8
            { "address":"江苏省南京市江宁区" }
9
10
       "nextAction" : {
11
           "action" : 7
12
13
14
        "reason" : "test",
15
        "userdata" : "test"
16
17
```

3.3 api->cr_web

api到cr_web,只需要将json数据外包装一个头,然后透传给cr_web就可以了

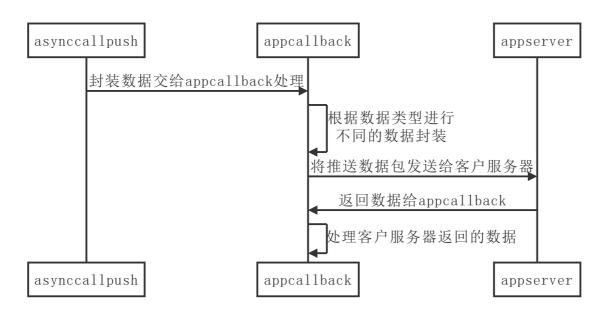
4 处理流程

cr给api的推送目前分为通话推送和坐席状态推送。

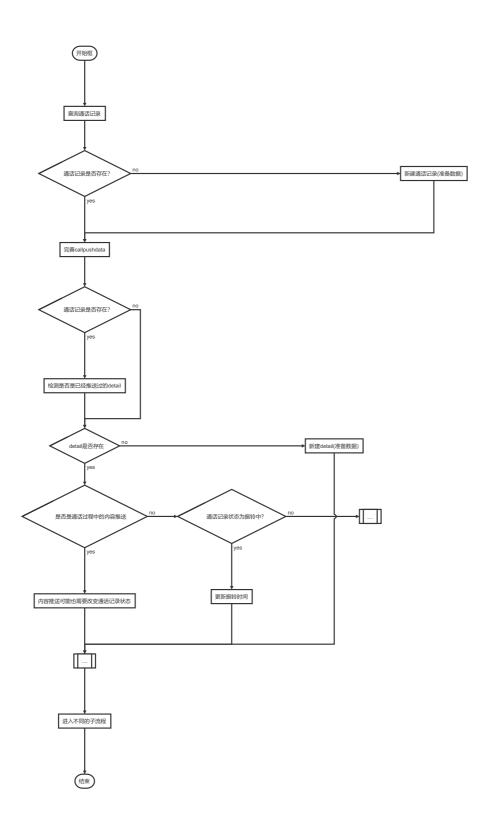
查询被叫默认按照通话推送进行处理,并且按照振铃推送的处理逻辑进行处理。

之前的查询被叫协议只会出现在呼入场景中,而现在的交互式ivr呼入呼出都有涉及,不仅仅只是查询被叫,同样的会有和通话无关的查询请求,首先考察是否会对通话记录产生影响

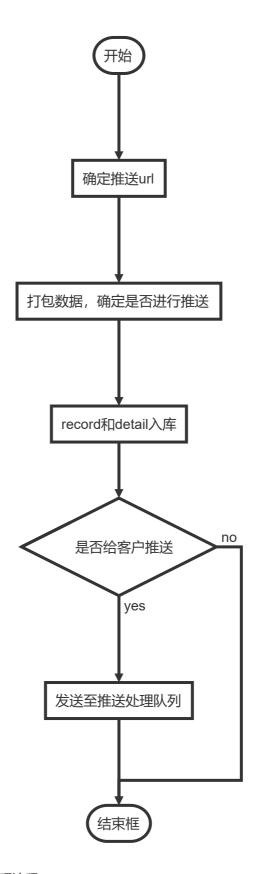
• api callpush模块时序图



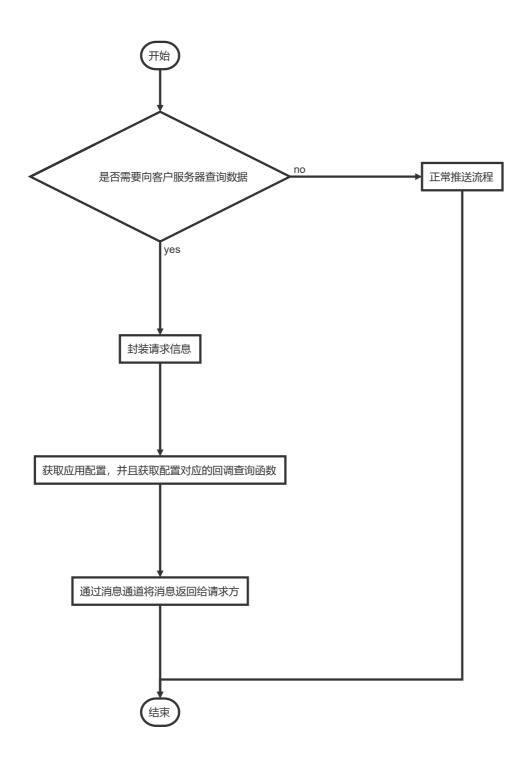
• asynccallpush流程图(status为0的分支)







• appcallback请求消息处理流程



- 1. 发向appcallback队列的数据会进行扩展,存储ivr扩展信息
- 2. 回调接口扩展,新建新的协议和对应的回调函数
- 3. 查询结果返回

4.1 95(交互收键模式)

- 4.2 96(被叫查询模式)
- 4.3 97(通用交互模式)

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

```
if( data->type == DB_CALL_RECORD_TYPE_QUERY_CALLED_BASE | |
 2
       data->type == DB_CALL_RECORD_TYPE_QUERY_CALLED_BY_PBX ||
 3
       data->type == DB_CALL_RECORD_TYPE_QUERY_CALLED_BY_USENUMBER ||
 4
       data->type == DB_CALL_RECORD_TYPE_QUERY_CALLED_BY_VIRTNUMBER )
   {
 6
        strcpy(call_record->useNumber, data->useNumber);
 7
 8
   else if(data->type == DB_CALL_RECORD_TYPE_QUERY_CALLED_BY_SUBNUMBER)
 9
        strcpy(call_record->subNumber, data->subNumber);
10
    else if(data->type == DB_CALL_RECORD_TYPE_QUERY_CALLED_BY_VIRTNUMBER)
11
        strcpy(call_record->virtNumber, data->virtNumber);
```

```
else if(IsOnlineCallPushType(data->type))

{
    if(call_record->status < DB_CALL_RECORD_STATUS_CALL_ESTABLISHED)
    data->status = call_record->status; //如果是查被叫的话重新改为振铃
    else
        data->status = DB_CALL_RECORD_STATUS_CALL_ESTABLISHED;

}
```

```
1 if(call_record->type >= DB_CALL_RECORD_TYPE_QUERY_CALLED_BASE)
2 {
```