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)

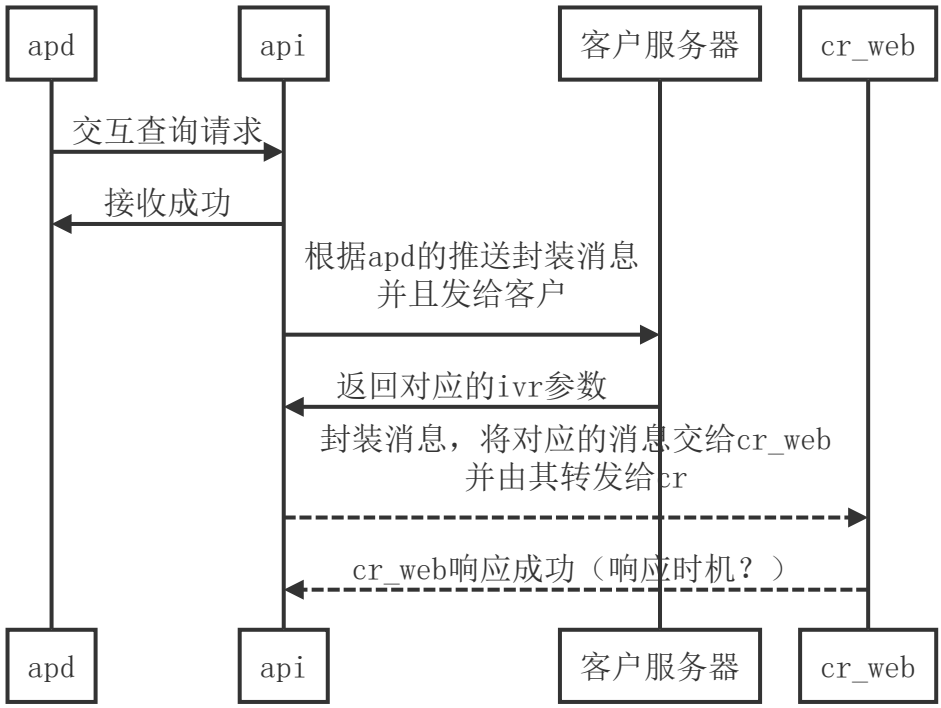
1.2 96(被叫查询模式)

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" : "apixxxxxxxxxxx",
    "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" : "apixxxxxxxxxxx",
    "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" : "apixxxxxxxxxxx",
    "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,
    "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,
    "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

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enterpriseId和provinceId做hash运算确定处理线程，因为provinceId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enterpriseId和provinceId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
typedef struct _emicalldev_db_callpushpost_data_
{
    unsigned long   id;
    BOOL            isCommonEnterprise;
    unsigned long   app_id;
    unsigned long   provinceId;
    unsigned long   enterpriseId;
    unsigned long   ccgeid; //新架构新增ccgeid字段，用以标识企业
    unsigned long   callLogId;
    int             type;
    unsigned char   callId[CALL_RECORD_CALLID_MAX_LEN];
    unsigned char   switchNumber[PHONE_NUMBER_MAX_LEN];
    DB_CALLPUSH_POSTDATA_STATUS    status;
    unsigned char   post_data[HTTP_CONTENT_BUFFER_SIZE];
    unsigned long   createTime;
    unsigned long   updateTime;
} 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
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;     //新架构新增ccgeid字段，用以标识企业
    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];

    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;
```

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

```
typedef enum
{
    IVR_CALLER_AND_CALLLED_TYPE_OUTLINE = 1,
    IVR_CALLER_AND_CALLLED_TYPE_INLINE = 2,
}IVRCallerAndCalledType;

typedef struct __push_post_data_ivr_moudle__
{
    IVRCallerAndCalledType    callerType;     //标记主叫是内线还是外线
    IVRCallerAndCalledType    calledType;     //标记被叫是内线还是外线
    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
```

```
    /*
    | 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-NO2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

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',
    'callType':99,
    'type':0,
    'useNumber':'02566687987',
    'switchNumber' : "02566699794",
    'userData':FE87D3
}
```

2. 95,96和97消息需要使用新架构新的消息格式

```
{
    "type" : 95/96/97,
    "ccgeid" : 1576,
    "callId" : "apixxxxxxxxxxx",
    "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":"" }
    ]
}
```

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

```
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;
```

3.2.2.1 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" : {
            "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"
        }
    }
    "reason" : "test",
    "userdata" : "test"
}
```

3.2.2.2 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" : {
            "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",
            "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"
}

```

3.2.2.3 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"
}
```

```
26
```

3.2.2.4 转座席响应

json to cr

```
{
    "rspCode" : 0,
    “ccgeid” : “123”,
    “ccNumber” : :21212”,
    "userQueryId" : "id_0000001",
    "variables" : [
        { "id_number" : "1101081987031276 21" },
        { "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"
}
```

3.2.2.5 转外线响应

json to cr

```
{
    "rspCode" : 0,
    “ccgeid” : “123”,
    "ccNumber" : :"21212”,
    "userQueryId" : "id_0000001",
    "variables" : [
        { "id_number" : "1101081987031276 21" },
        { "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"
}

```

3.2.2.6 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"
}
```

3.2.2.7 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"
}

```

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

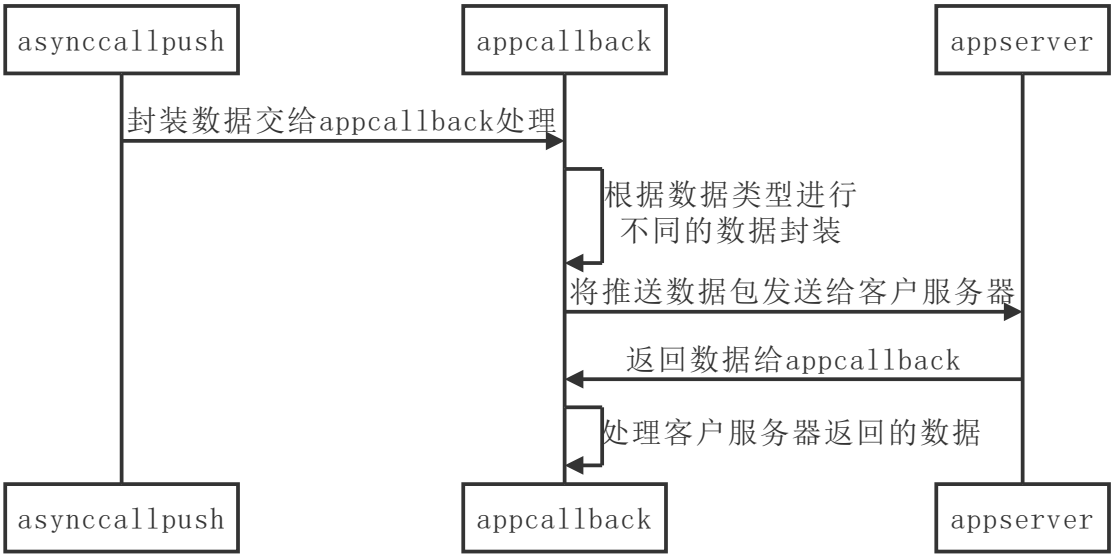
4 处理流程

cr给api的推送目前分为通话推送和坐席状态推送。

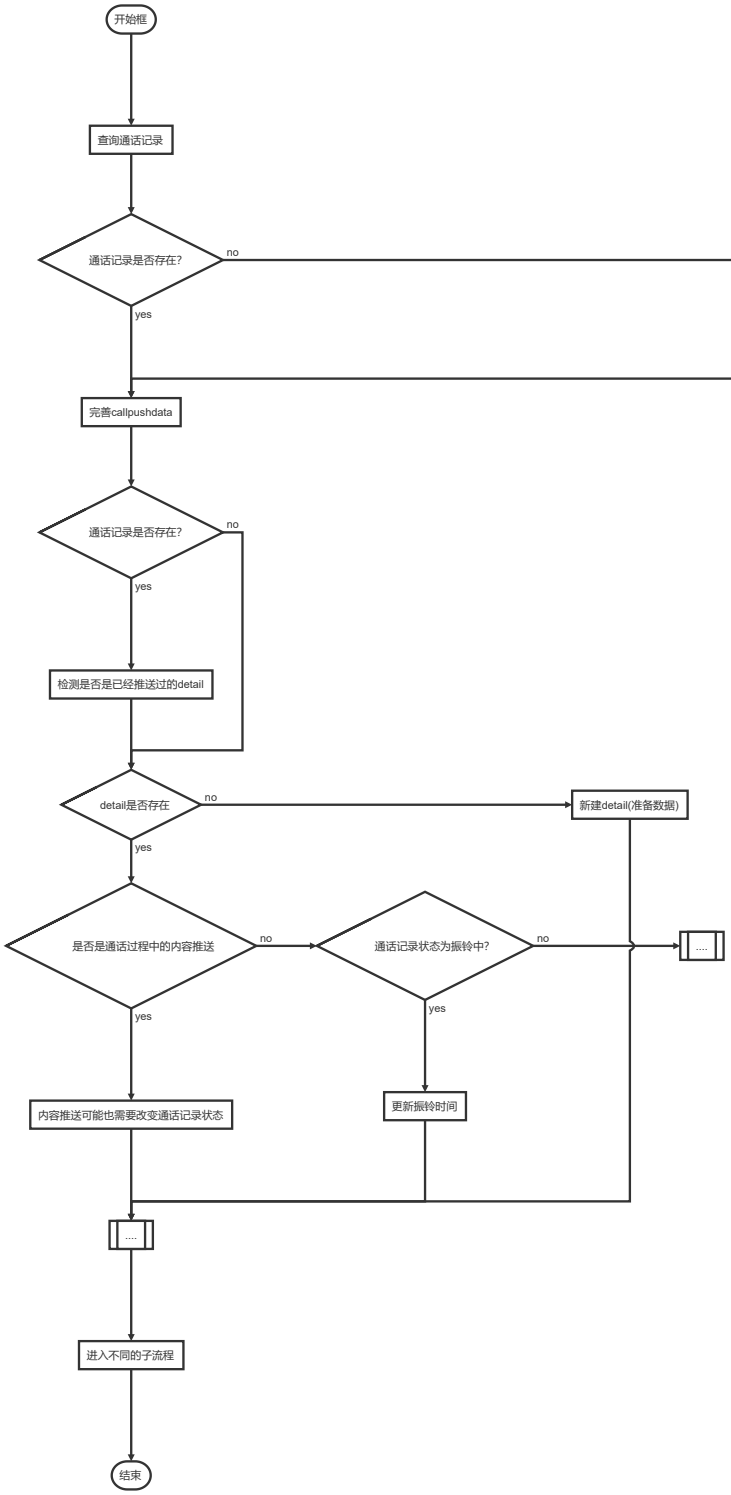
查询被叫默认按照通话推送进行处理，并且按照振铃推送的处理逻辑进行处理。

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

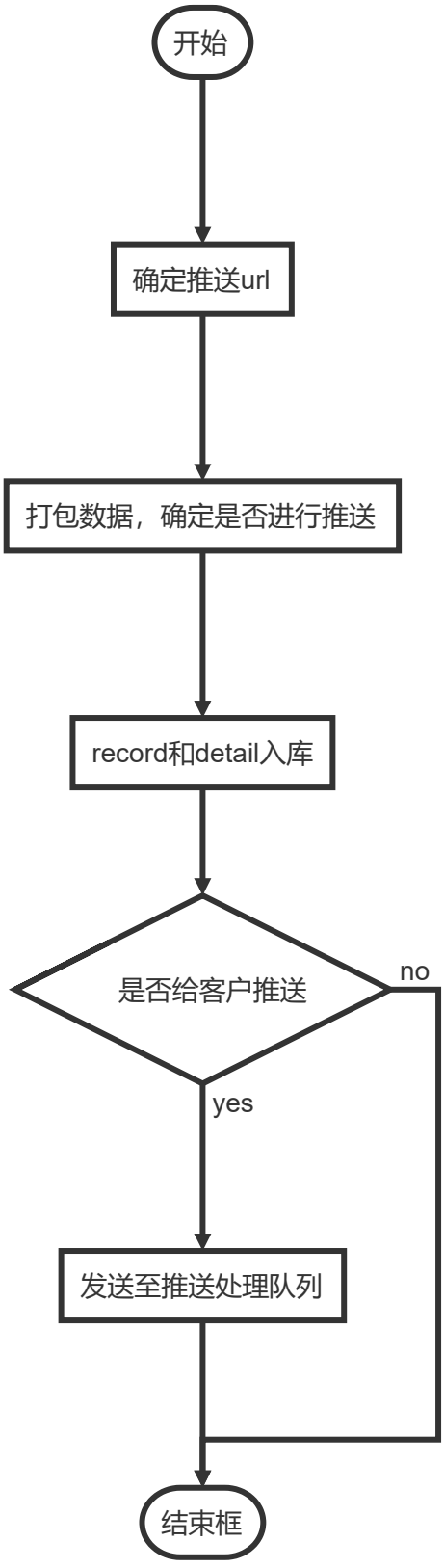
- api callpush模块时序图



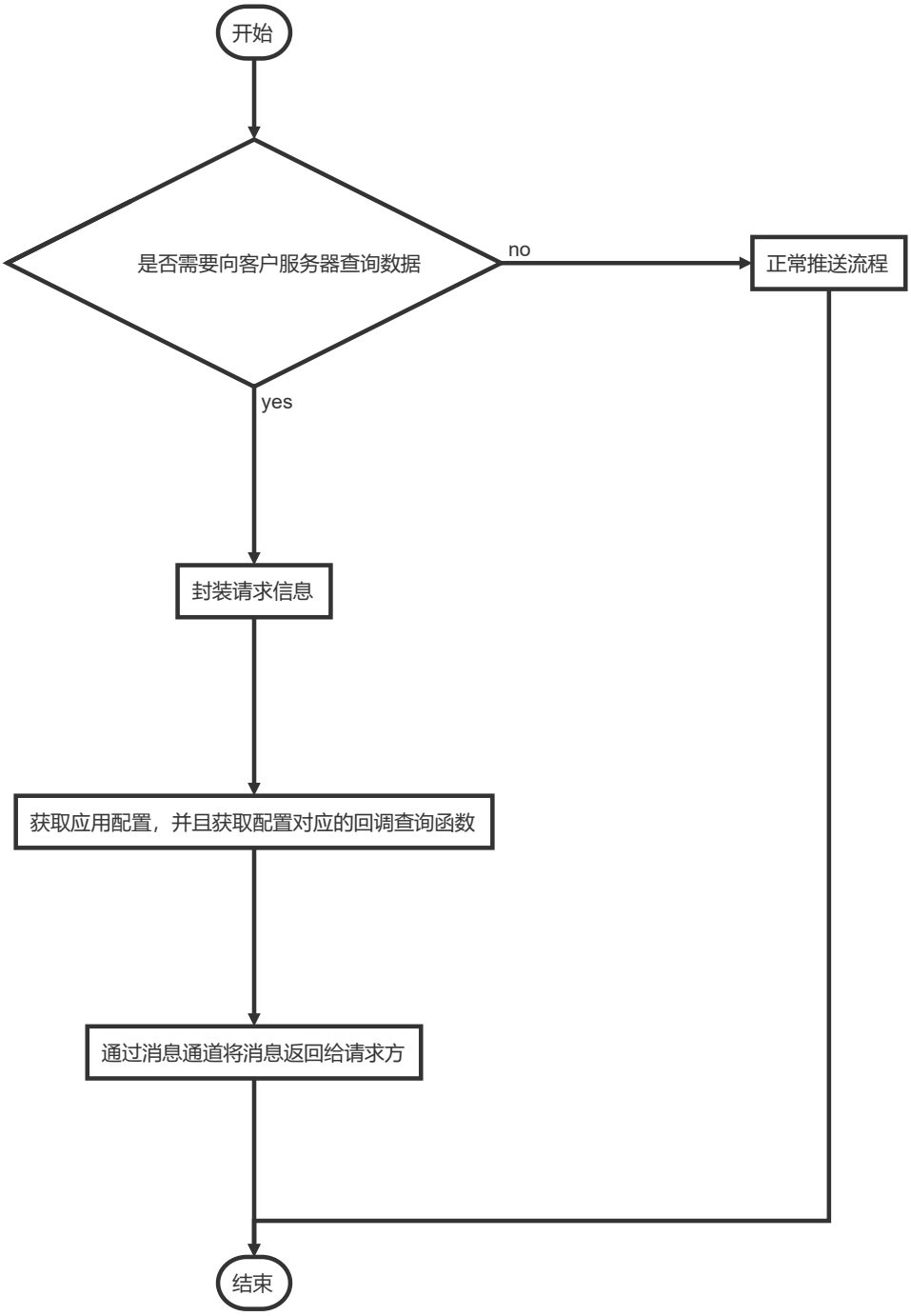
- asynccallpush流程图(status为0的分支)


开始框
查询通话记录
通话记录是否存在?
no
新建通话记录(准备数据)
yes
完善callpushdata
通话记录是否存在?
no
yes
检测是否是已经推送过的detail
detail是否存在
no
新建detail(准备数据)
yes
是否是通话过程中的内容推送
no
通话记录状态为振铃中?
no
....
yes
yes
内容推送可能也需要改变通话记录状态
更新振铃时间
....
进入不同的子流程
结束

- 通话振铃推送子流程图



- appcallback请求消息处理流程



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. 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 ||
   data->type == DB_CALL_RECORD_TYPE_QUERY_CALLED_BY_PBX ||
   data->type == DB_CALL_RECORD_TYPE_QUERY_CALLED_BY_USENUMBER ||
   data->type == DB_CALL_RECORD_TYPE_QUERY_CALLED_BY_VIRTNUMBER )
{
    strcpy(call_record->useNumber, data->useNumber);
}
else if(data->type == DB_CALL_RECORD_TYPE_QUERY_CALLED_BY_SUBNUMBER)
    strcpy(call_record->subNumber, data->subNumber);
else if(data->type == DB_CALL_RECORD_TYPE_QUERY_CALLED_BY_VIRTNUMBER)
    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;
}
```

```
else if(NULL != callreqUrl ||
        ( call_record->type >= DB_CALL_RECORD_TYPE_QUERY_CALLED_BASE &&
          DB_APPLICATION_CALLBACK_COMMON_PROTOCOL == appInfo->protocol) )
```

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