主要内容：

1 闹钟列表，每次下发的是全量列表（用户定过的所有闹钟），不再区分闹铃与提醒

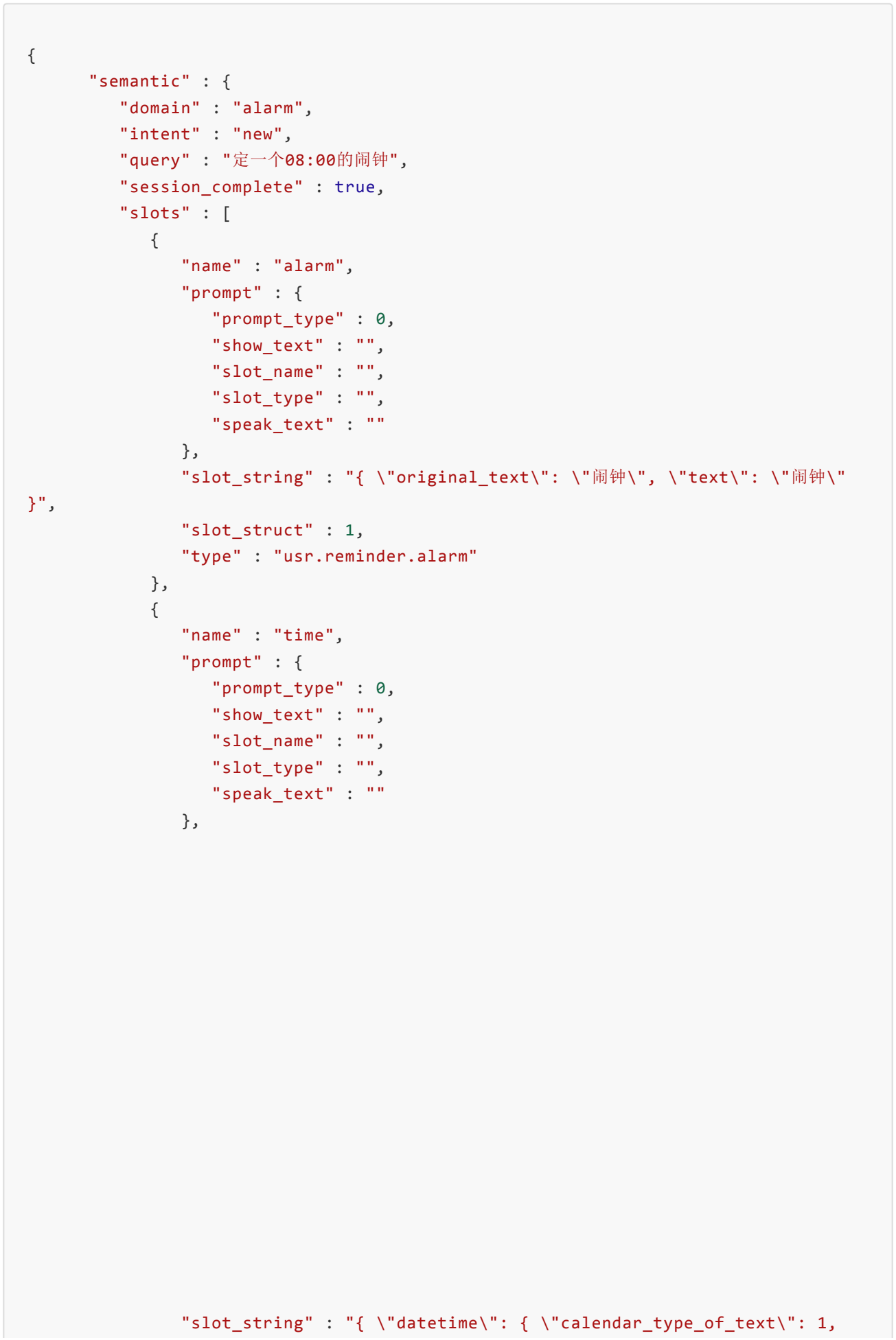
2 闹钟响铃策略

3 闹钟播报语

整体数据示例

语料：定一个08:00的闹钟

```
{
      "semantic" : {
         "domain" : "alarm",
         "intent" : "new",
         "query" : "定一个08:00的闹钟",
         "session_complete" : true,
         "slots" : [
            {
               "name" : "alarm",
               "prompt" : {
                  "prompt_type" : 0,
                  "show_text" : "",
                  "slot_name" : "",
                  "slot_type" : "",
                  "speak_text" : ""
               },
               "slot_string" : "{ \"original_text\": \"闹钟\", \"text\": \"闹钟\"
}",
               "slot_struct" : 1,
               "type" : "usr.reminder.alarm"
            },
            {
               "name" : "time",
               "prompt" : {
                  "prompt_type" : 0,
                  "show_text" : "",
                  "slot_name" : "",
                  "slot_type" : "",
                  "speak_text" : ""
               },
```


```
               "slot_string" : "{ \"datetime\": { \"calendar_type_of_text\": 1,
```

\"date\": \"\", \"day\": -1, \"express_type\": 1, \"hour\": 8, \"min\": 0, \"mon\": -1, \"original_text\": \"\", \"period_of_day\": 0, \"sec\": 0, \"time\": \"08:00:00\", \"week\": 44, \"year\": -1 }, \"interval\": { \"end\": { \"calendar_type_of_text\": 1, \"date\": \"\", \"day\": -1, \"express_type\": 1, \"hour\": -1, \"min\": -1, \"mon\": -1, \"original_text\": \"\", \"period_of_day\": 0, \"sec\": -1, \"time\": \"\", \"week\": -1, \"year\": -1 }, \"start\": { \"calendar_type_of_text\": 1, \"date\": \"\", \"day\": -1, \"express_type\": 1, \"hour\": -1, \"min\": -1, \"mon\": -1, \"original_text\": \"\", \"period_of_day\": 0, \"sec\": -1, \"time\": \"\", \"week\": -1, \"year\": -1 } }, \"is_valid\": true, \"old_json\": \"{\\\"entity\\\":\\\"08:00\\\",\\\"value\\\":\\\"08:00:00\\\",\\\"calendar_type\\\":\\\"solar\\\",\\\"express_type\\\":\\\"normal_time\\\",\\\"is_valid\\\":1}\", \"original_text\": \"08:00\", \"repeat\": { \"interval\": { \"end\": { \"calendar_type_of_text\": 1, \"date\": \"\", \"day\": -1, \"express_type\": 1, \"hour\": -1, \"min\": -1, \"mon\": -1, \"original_text\": \"\", \"period_of_day\": 0, \"sec\": -1, \"time\": \"\", \"week\": -1, \"year\": -1 }, \"start\": { \"calendar_type_of_text\": 1, \"date\": \"\", \"day\": -1, \"express_type\": 1, \"hour\": -1, \"min\": -1, \"mon\": -1, \"original_text\": \"\", \"period_of_day\": 0, \"sec\": -1, \"time\": \"\", \"week\": -1, \"year\": -1 } }, \"repeat_datetime_type\": 1 }, \"type\": 1 }",
            "slot_struct" : 0,
            "type" : "sys.time.freq"
         },
         {
            "name" : "date",
            "prompt" : {
               "prompt_type" : 0,
               "show_text" : "",
               "slot_name" : "",
               "slot_type" : "",
               "speak_text" : ""
            },
            "slot_string" : "{ \"datetime\": { \"calendar_type_of_text\": 1,

```
\"date\": \"2017-10-26\", \"day\": 26, \"express_type\": 1, \"hour\": 8, \"min\":
0, \"mon\": 10, \"original_text\": \"\", \"period_of_day\": 0, \"sec\": 0,
\"time\": \"08:00:00\", \"week\": 44, \"year\": 2017 }, \"interval\": { \"end\": {
\"calendar_type_of_text\": 1, \"date\": \"\", \"day\": -1, \"express_type\": 1,
\"hour\": -1, \"min\": -1, \"mon\": -1, \"original_text\": \"\",
\"period_of_day\": 0, \"sec\": -1, \"time\": \"\", \"week\": -1, \"year\": -1 },
\"start\": { \"calendar_type_of_text\": 1, \"date\": \"\", \"day\": -1,
\"express_type\": 1, \"hour\": -1, \"min\": -1, \"mon\": -1, \"original_text\":
\"\", \"period_of_day\": 0, \"sec\": -1, \"time\": \"\", \"week\": -1, \"year\":
-1 } }, \"is_valid\": true, \"old_json\": \"
{\\\"entity\\\":\\\"08:00\\\",\\\"value\\\":\\\"08:00:00\\\",\\\"calendar_type\\\"
:\\\"solar\\\",\\\"express_type\\\":\\\"normal_time\\\",\\\"is_valid\\\":1}\",
\"original_text\": \"\", \"repeat\": { \"interval\": { \"end\": {
\"calendar_type_of_text\": 1, \"date\": \"\", \"day\": -1, \"express_type\": 1,
\"hour\": -1, \"min\": -1, \"mon\": -1, \"original_text\": \"\",
\"period_of_day\": 0, \"sec\": -1, \"time\": \"\", \"week\": -1, \"year\": -1 },
\"start\": { \"calendar_type_of_text\": 1, \"date\": \"\", \"day\": -1,
\"express_type\": 1, \"hour\": -1, \"min\": -1, \"mon\": -1, \"original_text\":
\"\", \"period_of_day\": 0, \"sec\": -1, \"time\": \"\", \"week\": -1, \"year\":
-1 } }, \"repeat_datetime_type\": 1 }, \"type\": 1 }",
                "slot_struct" : 0,
                "type" : "sys.date.freq"
            }
         ]
      },
      "stCalendarData" : {
         "sSpeakTips" : "好的，已为你添加12小时51分钟后的闹钟，我会在明天上午8点提醒你",
         "stBroadData" : {
            "mBroadData" : {
               "1508976000" : "time##的闹钟时间到了"
            }
         },
         "stFormatRinging" : {
            "iCount" : 3,
            "iTotleLength" : 60000,
            "mDoingPolicy" : {
               "0" : false,
               "1" : true,
               "2" : true
            },
            "mInterPolicy" : {
               "1" : 0,
               "2" : 0,
               "3" : 0
            },
            "mPause" : {
               "1" : 600000,
               "2" : 300000
            },

            "vFormatRingingCell" : [
```

```
            {
               "eLightEffect" : 0,
               "vUseTime" : [ 500 ],
               "vUseType" : [ 2 ]
            },
            {
               "eLightEffect" : 1,
               "vUseTime" : [ 2000, 5000, 1000 ],
               "vUseType" : [ 0, 3, 1 ]
            },
            {
               "eLightEffect" : 2,
               "vUseTime" : [ 0 ],
               "vUseType" : [ 4 ]
            },
            {
               "eLightEffect" : 0,
               "vUseTime" : [ 500 ],
               "vUseType" : [ 2 ]
            },
            {
               "eLightEffect" : 1,
               "vUseTime" : [ 2000, 5000, 1000 ],
               "vUseType" : [ 0, 3, 1 ]
            },
            {
               "eLightEffect" : 2,
               "vUseTime" : [ 0 ],
               "vUseType" : [ 4 ]
            },
            {
               "eLightEffect" : 0,
               "vUseTime" : [ 500 ],
               "vUseType" : [ 2 ]
            },
            {
               "eLightEffect" : 1,
               "vUseTime" : [ 2000, -1 ],
               "vUseType" : [ 0, 3 ]
            }
         ]
      },
      "vReminderCell" : [
         {
            "eRepeatType" : 0,
            "lEnd" : 1508976000,
            "lId" : 1508929760407.0,
            "lStart" : 1508976000,
            "lUpdate" : 1508929760407.0,

            "sNote" : "",
```

```
                "sUrlId" : "",
                "vSelfData" : []
            }
        ]
    },
    "terminal_operation" : 0
}
```

结构一览

闹钟领域比较复杂，其整体数据结构为：

```
struct Clock {
  Semantic semantic;
  EnumCalendarTerminalOperation terminal_operation;
  CalendarData  stCalendarData;   // 最新接入方式使用该字段数据，其他字段忽略
};
struct CalendarData {
  string sSpeakTips;                          // 回复语
  vector<ReminderCell> vReminderCell;         // 闹钟列表（重要）
  FormatRinging stFormatRinging;              // 闹钟响铃策略
  BroadData stBroadData;                      // 闹钟播报语（重要）
};
```

最新接入方式使用stCalendarData字段。

闹钟列表解读

```
struct CalendarData {
  string sSpeakTips;                          // 回复语
  vector<ReminderCell> vReminderCell;         // 闹钟列表
  FormatRinging stFormatRinging;              // 闹钟响铃策略
  BroadData stBroadData;                      // 闹钟播报语
};
struct ReminderCell {
  long lId;                                  // 闹钟唯一ID
  long lStart;                               // 闹钟时间
  long lEnd;                                     // 闹钟失效时间
  long lUpdate;                              // 数据最近一次更新时间
  E_REPEAT_TYPE eRepeatType;                     // 闹钟类别（重复类型）
  vector<E_SELF_DATA> vSelfData;                 // 未启用，预留字段
  string sNote;                                  // 事项内容
  string sUrlId;                                 // 音乐Id
};
enum E_REPEAT_TYPE{
  E_REPEAT_ONCE = 0,                                    // 一次性闹钟
  E_REPEAT_DAY = 1,                                     // 按天重复
  E_REPEAT_WEEK = 2,                                    // 按周重复
  E_REPEAT_MONTH = 3,                                   // 按月重复
  E_REPEAT_WORKDAY = 4,                                 // 按工作日重复
  E_REPEAT_WEEKEND = 5,                                 // 按非工作日重复
  E_REPEAT_SELF = 6,                                    // 未启用
  E_REPEAT_HOUR = 7                                     // 按小时重复
};
```

示例：

```
"vReminderCell": [
  {
    "eRepeatType": 2,                    //闹钟类型：按周重复
    "lEnd": 3082327200,                  //失效时间：2067/9/4 10:0:0
    "lId": 1505461314068,                //唯一ID：1505461314068
    "lStart": 1505527200,                //响铃时间：2017/9/16 10:0:0
    "lUpdate": 1505461314068,            //最近一次更新时间：2017/9/15 15:41:54
    "sNote": "学习",                     //提醒事项：学习
    "sUrlId": "410316",                  //音乐ID 用于向音乐服务请求可播放的URL
    "vSelfData": [ ]
  },
  {
    "eRepeatType": 0,                    //闹钟类型：一次性闹钟
    "lEnd": 1506909600,
    "lId": 1505461853718,
    "lStart": 1506909600,
    "lUpdate": 1505462594878,
    "sNote": "",
    "sUrlId": "410316",
    "vSelfData": [ ]
  }
]
```

响铃策略解读

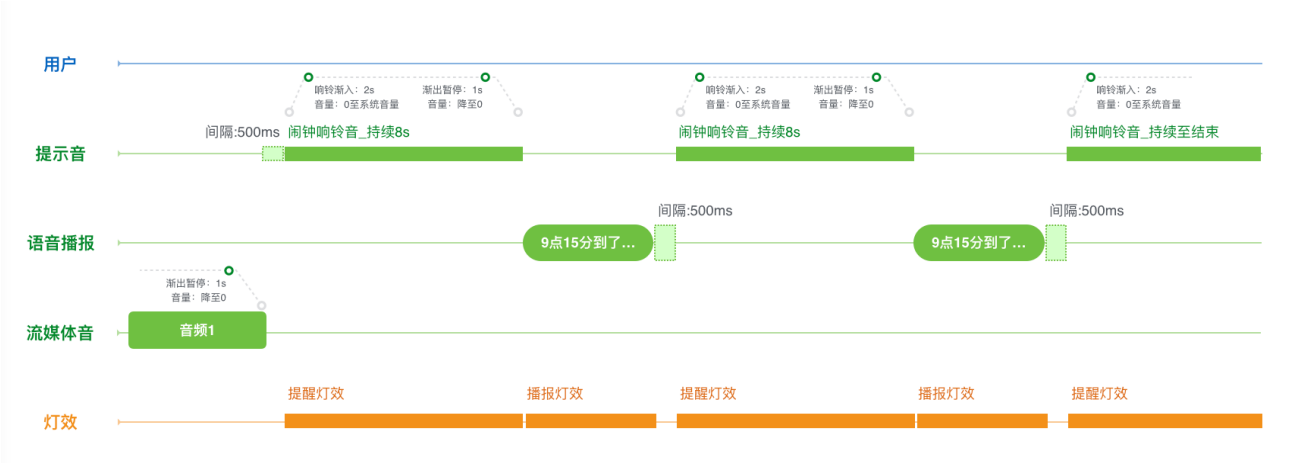
```
struct CalendarData {
  string sSpeakTips;                              // 回复语
  vector<ReminderCell> vReminderCell;             // 闹钟列表（重要）
  FormatRinging stFormatRinging;                  // 闹钟响铃策略
  BroadData stBroadData;                          // 闹钟播报语（重要）
};
struct FormatRinging{
  vector<FormatRingingCell> vFormatRingingCell;
  int iTotleLength;                                 // 单次响铃总时长
  int iCount;                                       // 最多响铃次数
  map<int, int> mPause;                              // 响铃之间的间隔，key为第N次响铃与第N+1次响铃之间间隔。
  map<int, E_INTERRUPT_TYPE> mInterPolicy;          // 打断后策略，key为第N次响铃
  map<int, bool> mDoingPolicy;          // 准备响铃时当前任务是否暂停 true表示暂停。任务类型见E_DOING_TYPE
};
// vUseType vUseTime 位置一一对应 前者表示元素性质 后者表示元素存在时间
struct FormatRingingCell{
  vector<E_VOICE_TYPE> vUseType;    // 响铃策略中元素组成
  vector<int> vUseTime;             // 单位ms -1时表示一直持续 0表示读完数据即可 例如：播报语播报
  E_LIGHT_EFFECT_TYPE eLightEffect; // 灯效，生命周期内灯效不变
};
enum E_VOICE_TYPE{
  E_VOICE_GETTING_IN,               // 声音渐入
  E_VOICE_FADE_OUT,                 // 声音渐出
  E_VOICE_PAUSE,                    // 暂停
  E_VOICE_LAST,                     // 声音持续
  E_VOICE_CONTENT,                  // 播报语
};
enum E_LIGHT_EFFECT_TYPE{
  E_LIGHT_EFFECT_NONE = 0,          // 无灯效
  E_LIGHT_EFFECT_RING = 1,          // 提示音灯效
  E_LIGHT_EFFECT_BROADCAST = 2,     // 播报语灯效
};
enum E_INTERRUPT_TYPE{
  E_INTERRUPT_STOP = 0,             // 打断后延时闹铃不再生效 流程终止
  E_INTERRUPT_CONTINUE = 1          // 打断后仅停止本次响铃流程 延时响铃策略依然生效
};
// FormatRinging结构体中字段mDoingPolicy 的 key值对应关系在这里取
enum E_DOING_TYPE{
  E_DOING_NONE = 0,                 // 无动作
  E_DOING_INPUT = 1,                // 输入动作
  E_DOING_PLAY = 2                  // 播放动作
};
```

示例：

```
"stFormatRinging": {
    "iCount": 3,                          //最多响铃三次
    "iTotleLength": 60000,                //单次响铃最长声明周期为1min
    "mDoingPolicy": {
        "0": false,                       //响铃时 无动作 不需要额外处理
        "1": true,                        //响铃时 正在进行输入动作 暂停输入
        "2": true                         //响铃时 正在播放 暂停播放
    },
    "mInterPolicy": {
        "1": 0,                           //第一次响铃时，打断，终止所有（三次）响铃流程
        "2": 0,                           //第二次响铃时，打断，终止所有（三次）响铃流程
        "3": 0                            //第三次响铃时，打断，终止所有（三次）响铃流程
    },
    "mPause": {
        "1": 600000,                      //第一次和第二次响铃间隔10min
        "2": 300000                       //第二次和第三次响铃间隔10min
    },
    "vFormatRingingCell": [
      {
          "eLightEffect": 0,              //无灯效
          "vUseTime": [ 500 ],            //持续时间500ms
          "vUseType": [ 2 ]               //E_VOICE_PAUSE 暂停动作
      },
      {
          "eLightEffect": 1,                      //提示音灯效
          "vUseTime": [ 2000, 5000, 1000 ],       //分别持续时间 2s,5s,1s 与下一项对应
          "vUseType": [ 0, 3, 1 ]                 //分别执行动作 声音渐入 持续 渐出
      },
      {
          "eLightEffect": 2,                      //播报语灯效
          "vUseTime": [ 0 ],                      //持续时间：播放完播报语
          "vUseType": [ 4 ]                       //E_VOICE_CONTENT 播报语
      },
      {
          "eLightEffect": 0,              //无灯效
          "vUseTime": [ 500 ],            //持续时间500ms
          "vUseType": [ 2 ]               //E_VOICE_PAUSE 暂停动作
      },
      {
          "eLightEffect": 1,                      //提示音灯效
          "vUseTime": [ 2000, 5000, 1000 ],       //分别持续时间 2s,5s,1s 与下一项对应
          "vUseType": [ 0, 3, 1 ]                 //分别执行动作 声音渐入 持续 渐出
      },
      {
          "eLightEffect": 2,                      //播报语灯效
          "vUseTime": [ 0 ],                      //分别持续时间 2s,5s,1s 与下一项对应
          "vUseType": [ 4 ]                       //分别执行动作 声音渐入 持续 渐出
      },
      {
          "eLightEffect": 0,              //无灯效
```

```
            "vUseTime": [ 500 ],            //持续时间500ms
            "vUseType": [ 2 ]               //E_VOICE_PAUSE 暂停动作
        },
        {
            "eLightEffect": 1,                      //提示音灯效
            "vUseTime": [ 2000, -1 ],               //分别持续时间 2s, -1 一直到流程结束
            "vUseType": [ 0, 3 ]                    //分别执行动作 声音渐入 持续
        }
    ]
}
```

附图示例说明：



闹钟播报语解读

```
struct CalendarData {
  string sSpeakTips;                            // 回复语
  vector<ReminderCell> vReminderCell;           // 闹钟列表
  FormatRinging stFormatRinging;                // 闹钟响铃策略
  BroadData stBroadData;                        // 闹钟播报语
};
struct BroadData{
  map<long, string> mBroadData;      // key表示触发时间（单位：秒） value表示播报语 其实“time##”需要客户端自行填充播报时时间 替换字符串
};
```

示例：

```
"stBroadData": {
    "mBroadData": {
        "1505527200": "time##到了，记得去学习", //"time##" 需要替换为客户端播报时的当前
时间
        "1505534400": "time##的闹钟时间到了",    // 例如：9点的闹钟时间到了
        "1506909600": "time##的闹钟时间到了"     // 1506909600 与 闹钟列表中 lStart 对
应
    }
}
```