IVR日志分析

# IVR日志介绍

## IVR日志所在的路径

UNIX&Linux环境下，IVR日志保存在平台用户下icddir/log目录下的以IVR 的ProgID命名目录。

Win环境下，IVR日志保存在huawei/IVR下的以IVR的ProgID命名目录。

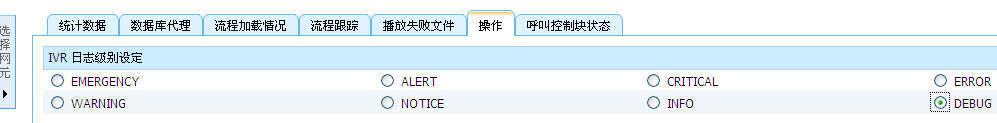
## IVR日志的分类以及生成方法

IVR日志分为IVR运行日志、IVR呼叫日志、IVR的Trace日志。

**IVR运行日志：**

名字为ivrXXX.log 或ivrXXX\_YY.log，（XXX代表的是当前IVR的ProgID，YY表示当前日志的编号），在IVR运行日志中，记录一些IVR运行的全局信息，例如IVR的启动、停止、配置数据的读取、去CCS/CTIServer注册等信息。

对于IVR运行日志，只要一启动就会生成。默认打印的是告警级别的日志，如果需要DEBUG级别的运行日志，则请使用有监控权限的座席登陆WAS系统监控台，在如下界面设置级别为debug即可。Debug级别的日志对IVR性能有较大影响，在呼叫量较高时，不推荐开启。



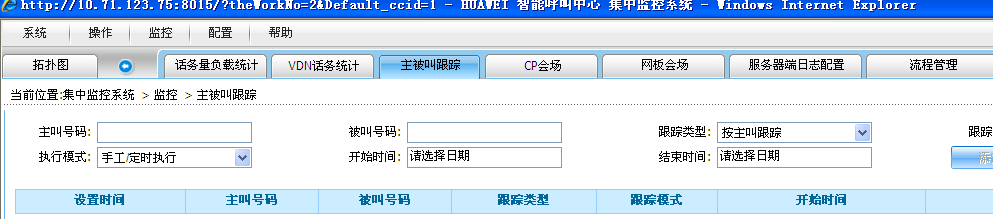
**IVR呼叫日志：**

名字为ivr\_Call.log（XXX代表的是当前IVR的ProgID，YY表示当前日志的编号）。其中记录呼叫相关信息，例如呼叫对应的系统接入码VDN信息等。

对于IVR呼叫日志，只要有呼叫到达IVR就会生成。但是只有在debug级别下的呼叫日志才会记录呼叫的全部信息，IVR呼叫日志debug级别设置的方法和IVR运行日志设置为debug级别的方法一样。

另外还有两种情况会在呼叫日志中记录呼叫详细信息：

1. 在WAS系统监控台如下页面进行主被叫跟踪时，生成的呼叫详细日志也会记录到call日志中。具体方法请参看WAS监控台帮助

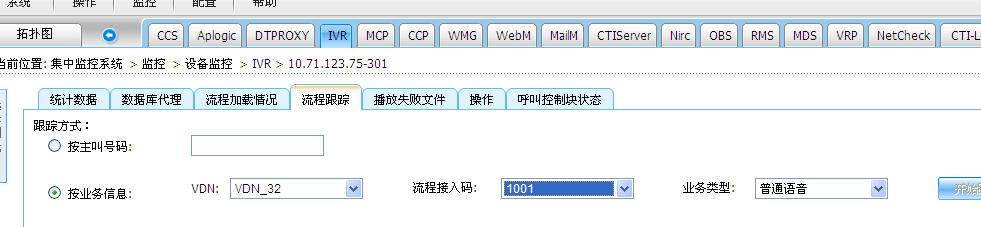


1. 如果IVR有呼叫吊死，则IVR对吊死呼叫记录一段时间的呼叫详细日志。

本文之后的呼叫日志描述都是debug级别。Debug级别的日志对IVR性能有较大影响，在呼叫量较高时，不推荐开启。

**IVR的Trace日志**

名字为ivrXXX\_trace.log 或ivrXXX\_ trace\_YY.log，（XXX代表的是当前IVR的ProgID，YY表示当前日志的编号）。其中记录其中记录呼叫相关信息，例如呼叫对应的系统接入码VDN信息等。只有对IVR进行流程跟踪时才会生成。则请使用有监控权限的座席登陆WAS系统监控台，在如下界面进行流程跟踪。

****

# IVR运行日志解释

2011-04-07 11:39:59 **[5204]** [RUN] [INFO] [0] ==========IVR START ICDV300R005C15==========

IVR启动时会打印此日志，并打出版本号，其中[5204]表示的是此IVR的PID，之后此IVR的所有日志都加有[5204]这个PID，如果PID变化，则说明IVR重启。

2011-04-07 11:39:59 [5204] [RUN] [INFO] [0] Config Item CCCLoginType=50105,HostType=3,UseMenuPlay=2,HttpThreadNum=50,DocCacheType=3,DocCacheMax=1000,VersionFlag=0 DefExpireTime=10,CheckSuspendCall=1,VxmlElementCacheType=0

2011-04-07 11:39:59 [5204] [RUN] [INFO] [0] Version=ICDV300R005C15 OS=Windows Working directory = C:\Program Files\huawei\IVR

2011-04-07 11:39:59 [5204] [RUN] [INFO] [0] HandShake Thread started: result=1 threadid=6036

2011-04-07 11:39:59 [5204] [RUN] [INFO] [0] Conference manager initialized success

上面日志是IVR内部模块初始过程

2011-04-07 11:40:00 [5204] [RUN] [INFO] [0] IRC logoff CCS send message.

2011-04-07 11:40:00 [5204] [RUN] [INFO] [0] IRC login CCS send message result=0

2011-04-07 11:40:00 [5204] [RUN] [INFO] [0] Login CCS call CccConnectToServerEx result result=0

2011-04-07 11:40:00 [5204] [RUN] [INFO] [0] Login CCS call CccLogon rc rc=0

IVR连接CCS过程

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] [DistNode].EnableFlag=0

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] IsSupportVxml **IsSupportVxml= 1**

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] TotalCICount **totalcicount= 300**

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] Init alarmer ip=1 id=301

IVR获取平台相关配置信息，IsSupportVxml表示此IVR是否支持VXML业务，totalcicount表示配置的此IVR [同时处理呼叫量](javascript:)

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] GSL file version: 530

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] **SCE version info: ICDV300R005C06#**

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] Show Msg=加载流程文件E:/flow/transfertoman.GSL成功 [VdnID=1, Access code=456]

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] Show Msg=加载流程文件E:\flow\PlayDTMF.vxml成功 [VdnID=1, Access code=6688019]

2011-04-07 11:40:02 [5204] [RUN] [ALERT] [0] Alert Type=8016, Level=2, Time=0, Info=加载流程文件失败

IVR加载流程过程，其中SCE version info表示编译此gsl流程所使用的SCE版本

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] Login CCS success!

IVR连接CCS成功

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] Set vxml ftp flag false.

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] Set vxml sync file path and temp file path **filepath=F:,tempfilepath=F:**

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] Set ProxyInfo and session timeout server=,port=8080,proxyUserName=,sessiontimeout=10000

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] Vxmldll version id = 30

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] Set record stat flag, flag=0

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] Vxmldll init session path path=C:/Program Files/huawei/IVR

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] set vxml parser num num=2

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] set Max Wait Time Time=3

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] set version flag=0

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] set default **expireTime=10**

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] start libagent succeed, pid=[4992]

2011-04-07 11:40:02 [5204] [RUN] [ERROR] [0] Load vxmlencapi.dll failed error code 126 in windwos!

VXML模块初始化过程，其中filepath和tempfilepath表示访问文件服务器的共享盘符。expireTime表示获取vxml脚本的超时时间。

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] IRC login CCS send message result=1

2011-04-07 11:40:02 [5204] [RUN] [ERROR] [0] IRC同CCS断连恢复。原因分析：在系统初启，断连恢复后都会有此告警提示。处理建议：无。

2011-04-07 11:40:02 [5204] [RUN] [INFO] [0] Show Msg=IRC连接CCS成功!

IRC模块初始化并连接CCS过程（IRC是用来做智能路由脚本使用的，使用局点不多）。

2011-04-07 11:40:03 [5204] [RUN] [WARN] [0] Try login CTIServer

2011-04-07 11:40:03 [5204] [RUN] [INFO] [0] Login ctiserver... MyIp=10.70.101.150 CtisIp=0.0.0.0 Version=50105 HostType=3

2011-04-07 11:40:03 [5204] [RUN] [INFO] [0] **Login ctiserver Ret=0 MyIp=10.70.101.150 CtisIp=10.70.101.150** Version=50105 **HostType=1** CtiDsn=301 CtiHandle=255

2011-04-07 11:40:03 [5204] [RUN] [INFO] [0] CccIVRSetINNO code result code= 1!123,223,852,251,357,456,6688019,6688016;, result=0

2011-04-07 11:40:03 [5204] [RUN] [INFO] [0] CTIServer connected Ok.

IVR连接CTIServer并上报流程接入码信息。Login ctiserver Ret这条日志中会打印当前IVR的IP地址（MyIp），主用ctiserver的IP地址（Ctisip），以及主机类型（HostType）。

其中HostType=1表示是用于对接AIP的CTIServer，HostType=2表示是用于对接UAP R3版本（不支持视频）的CTIServer，HostType=3表示是用于对接UAP R5版本（支持视频）的CTIServer。

2011-04-07 11:40:12 [5204] [RUN] [WARN] [0] IVR 已经正常启动。原因分析：无。处理建议：无。

至此IVR启动完毕，连接CTIServer、CCS成功。能够正常处理呼叫了。

2011-04-07 11:45:02 [5204] [RUN] [WARN] [0] ActiveCall=0 ,ShakehandOvertime=0, ExceedMaxTalkTime=0 CallIn=0 DisconnectNoResource=0

2011-04-07 11:50:02 [5204] [RUN] [WARN] [0] ActiveCall=0 ,ShakehandOvertime=0, ExceedMaxTalkTime=0 CallIn=0 DisconnectNoResource=0

2011-04-07 11:55:02 [5204] [RUN] [WARN] [0] ActiveCall=0 ,ShakehandOvertime=0, ExceedMaxTalkTime=0 CallIn=0 DisconnectNoResource=0

2011-04-07 12:00:03 [5204] [RUN] [WARN] [0] ActiveCall=0 ,ShakehandOvertime=0, ExceedMaxTalkTime=0 CallIn=0 DisconnectNoResource=0

2011-04-07 12:05:03 [5204] [RUN] [WARN] [0] ActiveCall=0 ,ShakehandOvertime=0, ExceedMaxTalkTime=0 CallIn=0 DisconnectNoResource=0

上面是IVR呼叫统计信息，每隔5分钟会打印一次。

ActiveCall表示打印日志时刻此IVR上的活动呼叫量，

ShakehandOverTime表示是打印日志时刻此IVR上的和CTIServer握手超时的呼叫量（即用户已经拆线了，但是业务没有处理拆线消息，继续运行），

ExceedMaxTalkTime表示超过了最大通话时间的呼叫量（默认最大通话时间是3600s，可以在“VDN配置台－流程”配置页面选择对应的流程，修改其中的“限制最大通话时间”）

CallIn表示这5分钟内分配到此IVR的累计呼叫数

DisconnectNoResource表示这5分钟内，分配到此IVR，但是因为IVR呼叫资源被占满无法服而拆线的累计呼叫数。

一般情况下ShakehandOverTime、ExceedMaxTalkTime、DisconnectNoResource的值都应该为0

2011-04-07 12:05:04 [5212] [RUN] [INFO] [0] MDS send the exit command, ivr will exit forcedly

收到MDS发送的退出请求

。。。

。。。

。。。

2011-04-07 12:05:11 [5212] [RUN] [INFO] [0] !!!!!!!!!!ivr exited. !!!!!!!!!!

IVR退出完毕

# IVR呼叫日志解释

## VXML流程相关呼叫日志解释

如下是IVR呼叫的debug级别日志

2011-04-13 14:18:46.603 **7** ======================XML\_CALL\_BEGIN======================

2011-04-13 14:18:46.603 **7** CallInfo TrunkNO=0,**CLI=753951,CLD=667702012,CLOD=12345678,**

2011-04-13 14:18:46.603  **7** sender\_handle=1,sender\_dsn=7,receiver\_handle=255,receiver\_dsn=21,

2011-04-13 14:18:46.603 **7** **AccessCode=6677020 Exec flow E:\flow\vxml\PlayRecord.vxml Version=ICDV300R005C15 OS=Windows**

上面是到达IVR呼叫的基本信息，CLI表示主叫，CLOD表示原始被叫（即排队机发送给平台的Setup中的org\_CLD字段，并不是IE信息中的原始被叫）

**注意CLD并不是被叫号码而是流程接入码**，ivr日志中显示的被叫号码不是用户拨入的被叫号码，而是流程接入码+后续接入码。比如本例中呼入的被叫号码是100212，匹配到平台配置的被叫1002，对应的流程接入码是6677020，则ivr打印的CLD字段就是667702012。同时从当前呼叫的会话变量session.telephone.dnis就是CLD这个值。如果流程想获得用户拨入的被叫号码，需要调用接口QueryCallInfo，具体方法请参看VXML流程开发指南。

AccessCode表示是流程接入码，Flow表示是当前接入码对应的流程

Version表示是当前IVR的版本号，OS表示是当前IVR所在的操作系统

另外上面的“ 7” 表示是当前呼叫在IVR中的ID，如果当前IVR有多个呼叫同时存在，则IVR日志中这些呼叫的日志就会互相交错，根据这个ID（注意要加上前后的空格）使用UltraEdit将相同ID的日志copy出来，就是单独的一个呼叫的日志了，方便分析。

2011-04-13 14:18:46.603 7 Monitor info ICDDebug=0 icdinsider=0 DebugMode=0 TraceFlag=0

2011-04-13 14:18:46.603 7 Send alerting param: audio=255

2011-04-13 14:18:46.603 7 IVR(1,7)->CCS(255,21): 振铃:

2011-04-13 14:18:46.603 7 CCS(255,21)->IVR(1,7): 呼叫标识

2011-04-13 14:18:46.603 7 CCS(255,21)->IVR(1,7): 通知:

2011-04-13 14:18:46.603 7 RePlace CallID list: ssp=5631 CallID=' 1302675526 25 0 0'

2011-04-13 14:18:46.603 7 Add CallID list: ssp=5631 CallID=' 1302675526 25 0 0'

2011-04-13 14:18:46.619 7 Answer param: SSP=5631, ChgInd=0, ExtraDataLen=0

2011-04-13 14:18:46.619 7 IVR(1,7)->CCS(255,21): 摘机:

2011-04-13 14:18:46.635 7 CCS(255,21)->IVR(1,7): 摘机应答:

上面是呼叫到达IVR后的接续过程，大致的呼叫接续协议如下：

1. CCS通知IVR呼叫到达
2. IVR给CCS回应振铃消息（Alerting）
3. CCS通知IVR呼叫标志和呼叫通知（Notify）
4. **IVR给CCS发送摘机消息（Answer）即计费消息**
5. CCS给IVR发送计费应答消息（AnswerAck）

上面的这些消息交互包括计费是IVR主动触发的，业务无法控制，现场业务往往有需要自主控制计费的需求，则需要在VND配置中配置流程名前面增加“startservice”（具体配置方法参看配置台帮助中对于“startservice”的描述），然后在流程中主动调用“振铃”，然后在需要计费的地方调用“应答”（“振铃”和“应答”调用方法参看VXML开发指南），注意在startservice配置下，如果流程没有调用“振铃”，则流程后续很多操作都是被限制的，如获取呼叫信息等。

2011-04-13 14:18:46.635 7 The CUR\_IVR\_P\_VER is 0x50105. The g\_ulVerNego is 0x50105

2011-04-13 14:18:46.635 7 Interpreter JavaScript JS\_GC Context=0x51b1590...OK!

2011-04-13 14:18:46.635 7 GetIeInfoValue function params: cMsgType = 1, cIeName = 126, cInfoName =0

2011-04-13 14:18:46.635 7 find ET\_USER\_USER return null.

2011-04-13 14:18:46.635 7 **Interpreter get Document**... FullUrl: E:\flow\vxml\PlayRecord.vxml PostData: method: 0 caching type: 0

IVR开始获取对应的脚本

2011-04-13 14:18:46.635 7 Get File, URL: E:\flow\vxml\PlayRecord.vxml?

2011-04-13 14:18:46.635 7 GetURLPage return GetFileContent E:\flow\vxml\PlayRecord.vxml

2011-04-13 14:18:46.635 7 CVxmlCharge::URLHit start

2011-04-13 14:18:46.635 7 CVxmlCharge::URLHit end

2011-04-13 14:18:46.635 7 BEGIN=13842421, END=13842421, TOTIAL=0, GET=E:\flow\vxml\PlayRecord.vxml?

2011-04-13 14:18:46.635 7 **Interpreter get document file succeed!**

IVR获取对应的脚本完毕，IVR开始解析脚本

2011-04-13 14:18:46.635 7 **Begin parser document**...

2011-04-13 14:18:46.635 7 asrserviceid= ServiceId=1

2011-04-13 14:18:46.635 7 sOldAppName=, sAppName=E:\flow\vxml\PlayRecord.vxml?

2011-04-13 14:18:46.635 7 new::VxmlDocument 0x4ff5118, E:\flow\vxml\PlayRecord.vxml?

2011-04-13 14:18:46.635 7 add voice static grammar digits.

2011-04-13 14:18:46.635 7 add dtmf static grammar digits.

2011-04-13 14:18:46.635 7 Begin parser document...OK!

2011-04-13 14:18:46.635 7 1VxmlParserEnd()...OK!

2011-04-13 14:18:46.635 7 **Parse document Succeed!**

2011-04-13 14:18:46.635 7 **BEGIN=13842421, END=13842421, TOTIAL=0**

IVR解析对应的脚本完毕，开始执行脚本

2011-04-13 14:18:46.635 7 Var exec set expr, userid =

2011-04-13 14:18:46.635 7 Var exec set expr, filename =

2011-04-13 14:18:46.635 7 Interpreter now send new DGid, but no new grammar need to send.

2011-04-13 14:18:46.635 7 Interpreter GO ...

2011-04-13 14:18:46.635 7 Current dialog:form[18]

2011-04-13 14:18:46.635 7 **Execute Field: PlayOrRecord**

IVR执行脚本中第一个元素field，对应的名字是PlayOrRecord，对应流程是进行放音收号操作

2011-04-13 14:18:46.635 7 Interpreter gather active DGid, active grammar id: 16908285, grammar file name: dg101digits0fffd

2011-04-13 14:18:46.635 7 ===>MsgCount=1 MaxDigit=1 InterTime=5 EndDelineater=12 FirstTime=5 CancelDigit=255 InputModes=2

2011-04-13 14:18:46.635 7 Enter VXML\_SpeakInput\_E() function, Param: MaxDigit=1,InterTime=5,EndDelineater=12,FirstTime=5,CancelDigit=255,InputModes=2

2011-04-13 14:18:46.635 7 CI Speak&Input max\_digit = 1 , inter\_time = 5 , end\_delineater = 255 , first\_time = 0 <prompt> Count = 1

2011-04-13 14:18:46.635 7 <prompt bargein='1'> strTtsAudio = 你好,请选择0为录音1为放音

这个field是一个放音操作，放的是字符串“你好,请选择0为录音1为放音”

2011-04-13 14:18:46.635 7 Exec VXML\_MenuPlayDrv.

2011-04-13 14:18:46.635 7 MenuPlayDrv iAudio = -1 Play =你好,请选择0为录音1为放音

2011-04-13 14:18:46.635 7 **IVR(1,7)->CCS(255,21):** 放音收号消息:放音识别收号的命令类型为13; VP **放音 收号**, **Max:1, Min:1, Inter:5, First:0, Start:255, Cancel:255, End:255;**

IVR将放音消息发给CCS处理，此消息是放音并而且收号，最大收号长度是1（Max），最小收号长度是1（Min），位间收号超时时间是5s（Inter），首位收号超时时间是0（First），即不会首位收号超时。没有设置开始收号键（Start），没有设置取消收号键（Cancel），没有结束收号键（End）

2011-04-13 14:18:46.650 7 CCS(255,21)->IVR(1,7): ICD录放音应答:放音,成功,VPDsn=65535

CCS返回放音应答成功，注意这个成功表示开始放音收号了，并不是放音结束。

2011-04-13 14:18:46.650 7 Enter IVR\_ProcessEvent() function, Param: UserDialing=1, UserHookOff=1,TimeOut=1, ResAvailable=0, DeviceInfo=1,Notify=0, UserAsr=0

2011-04-13 14:18:46.650 7 [事件处理]

2011-04-13 14:18:51.682 7 CCS(255,21)->IVR(1,7): **收号应答: Cause: Max number, Info: {\*0FFFFFFFFFFFFFFFFFFFFFFF\*}**

等待到用户按键了，CCS给IVR发送了收号应答，收号完毕对应号码是0，注意此处收号信息最多只能显示前两位，用于加密，如果收号超过两位，则后面号码都以E表示，例如如果收号信息是12345，则此处打印是{\*12EEEFFFFFFFFFFFFFFFFFFF\*}

2011-04-13 14:18:51.682 7 ProcessEvent receive msg type=15, mask=251

2011-04-13 14:18:51.682 7 拨号出口

2011-04-13 14:18:51.682 7 Get Message form Queue! Event=拨号出口(0)

2011-04-13 14:18:51.682 7 Exec stop vp play!

2011-04-13 14:18:51.682 7 IVR(1,7)->CCS(255,21): 停止录放音:MSG NOT SPECIFIED(22)

2011-04-13 14:18:51.682 7 CCS(255,21)->IVR(1,7): 停止录放音应答:MSG NOT SPECIFIED(23)

收号完毕要停止放音操作，IVR通知CCS停止放音，CCS返回停止放音成功

2011-04-13 14:18:51.682 7 receive res msg

2011-04-13 14:18:51.682 7 <===SpeakInputResult=0 input: 0\*\*\*0 InputType=1

2011-04-13 14:18:51.682 7 Interpreter get user input: 0\*\*\*0 dtmf

2011-04-13 14:18:51.682 7 Interpreter process phrase

2011-04-13 14:18:51.682 7 Current Dialog:form[18] , PlayOrRecord , form[18]

2011-04-13 14:18:51.682 7 Interpreter, var:PlayOrRecord=$

2011-04-13 14:18:51.682 7 Clear var list:RecordOBJ

2011-04-13 14:18:51.682 7 Current dialog:form[18]

2011-04-13 14:18:51.682 7 Execute Block: block18

2011-04-13 14:18:51.682 7 Current dialog:form[18]

2011-04-13 14:18:51.682 7 **Execute Object: dbtest**

流程执行field后的下一个元素 Object 名字是dbtest，对应流程是调用存储过程访问数据库

2011-04-13 14:18:51.682 7 Enter VXML\_DBStoreProc() function, Param: ProcName=GetRecordFileName,Index=2,Delay=6,pDataSource=ds\_record

对应存储过程名是GetRecordFileName，数据源名是ds\_record

2011-04-13 14:18:51.682 7 [ICD存储过程]

2011-04-13 14:18:51.682 7 SetDBDelayTime DelaySeconds=6.

2011-04-13 14:18:51.682 7 存储过程名：{\*GetRecordFileName\*}

2011-04-13 14:18:51.682 7 name={\*GetRecordFileName\*},num=2,dataSource=ds\_record

2011-04-13 14:18:51.682 7 IVR(1,7)->数据库代理:12(255,65535): 执行存储过程执行存储过程：存储过程名{\*GetRecordFileName\*} 代理节点号**2Proxy(IP=10.70.101.150:201)**

IVR给10.70.101.150上ProgID是201的数据库代理即APLogic发送消息，执行存储过程

2011-04-13 14:18:51.697 7 数据库代理:12(255,65535)->IVR(1,7): 执行存储过程应答MSG NOT SPECIFIED(89)

2011-04-13 14:18:51.697 7 Proc successed Params: [P0]:type=199 value=7\*\*\*(6) [P1]:type=199 value=Y\*\*\*wav(13)

数据库代理返回执行结果成功，日志中会打印执行结果时会进行加密。

2011-04-13 14:18:51.697 7 call VXML\_DBStoreProc() function, result=1

2011-04-13 14:18:51.697 7 OutputParam: filename.value = 'Y:/753951.wav'

2011-04-13 14:18:51.697 7 Current dialog:form[18]

2011-04-13 14:18:51.697 7 **Execute Object: RecordOBJ**

流程执行Object dbtest后的下一个元素 Object 名字是RecordOBJ，对应流程是发起录音操作

2011-04-13 14:18:51.697 7 IVR VP **Record(sFileName=Y:/753951.wav**, nMaxTime=10, nRecordMode=0, nRecordType=1, nRecordObject=0, nRecordObjectSSP=4294967295, nDtmfEnd=1, nBeep=1, pUCtermchar=0, nLocationID=65535, nAssistFlag=255)

流程设置的录音文件名是Y:/753951.wav

2011-04-13 14:18:51.697 7 Enter IVR\_SendVPPlayDrv() function, Param: MaxDigit=0, InterTime=0, EndDigit=0, MinDigit=0, CancelDigit=0, StartDigit=0, FirstTime=0, PlayType=2, PlaySentence=F:/beep.wav, PlayTimes=1, PlayObject=0, SSP=-1, LocationID=65535

2011-04-13 14:18:51.697 7 **IVR(1,7)->CCS(255,21): 放音收号:Resource:VP; 指定文件: 次数 = 1, 文件名 = F:/beep.wav**

2011-04-13 14:18:51.713 7 **CCS(255,21)->IVR(1,7): ICD录放音应答:放音,成功**,VPDsn=65535

录制前需要播放Beep音，IVR给CCS发送播放操作CCS返回放音应答成功，开始播放Beep音

2011-04-13 14:18:51.713 7 Enter IVR\_ProcessEvent() function, Param: UserDialing=0, UserHookOff=0,TimeOut=1, ResAvailable=0, DeviceInfo=1,Notify=0, UserAsr=0

2011-04-13 14:18:51.713 7 [事件处理]

2011-04-13 14:18:51.713 7 ProcessEvent receive msg type=23, mask=248

2011-04-13 14:18:55.728 7 **CCS(255,21)->IVR(1,7): 录放音结束**:MSG NOT SPECIFIED(21)

Beep音播放结束，CCS给IVR发送放音结束消息

2011-04-13 14:18:55.728 7 ProcessEvent receive msg type=21, mask=248

2011-04-13 14:18:55.728 7 设备消息出口

2011-04-13 14:18:55.728 7 Get Message form Queue! Event=设备消息出口(6)

2011-04-13 14:18:55.728 7 **IVR(1,7)->CCS(255,21)**: **录音:对呼叫方录音** 录指定文件: DsnSSP=21, Format=0, Mode=0, record\_len=10, **fileName=Y:\753951.wav**, vp\_dsn=65535

2011-04-13 14:18:55.728 7 **CCS(255,21)->IVR(1,7)**: **ICD录放音应答:录音,成功**,VPDsn=65535

IVR给CCS发送录音消息，录制的文件名是Y:\753951.wav，CCS返回录音开始成功

2011-04-13 14:18:55.728 7 **IVR(1,7)->CCS(255,21): 收号**: Max:1, Min:1, Inter:120, First:0, Start:255, Cancel:255, End:12;

IVR给CCS下发收号操作

2011-04-13 14:18:55.728 7 Enter IVR\_ProcessEvent() function, Param: UserDialing=1, UserHookOff=0,TimeOut=1, ResAvailable=0, DeviceInfo=1,Notify=0, UserAsr=0

2011-04-13 14:18:55.728 7 [事件处理]

2011-04-13 14:18:59.853 7 **CCS(255,21)->IVR(1,7): 收号应答**: Cause: Max number, Info: {\*BFFFFFFFFFFFFFFFFFFFFFFF\*}

用户按键，CCS给IVR回应收号结果是B，即”\*”键。

2011-04-13 14:18:59.853 7 ProcessEvent receive msg type=15, mask=249

2011-04-13 14:18:59.853 7 拨号出口

2011-04-13 14:18:59.853 7 Get Message form Queue! Event=拨号出口(0)

2011-04-13 14:18:59.853 7 Enter IVR\_StopVPRecord() function, Param: SSP=-1

2011-04-13 14:18:59.853 7 [停止VP录放音]

2011-04-13 14:18:59.853 7 Exec stop vp play!

2011-04-13 14:18:59.853 7 **IVR(1,7)->CCS(255,21): 停止录放音**:MSG NOT SPECIFIED(22)

2011-04-13 14:18:59.853 7 **CCS(255,21)->IVR(1,7): 停止录放音应答**:MSG NOT SPECIFIED(23)

IVR给CCS发送停止录音操作，CCS返回停止录音成功

2011-04-13 14:18:59.853 7 成功出口

2011-04-13 14:18:59.853 7 Call IVR\_VPRecord function, result=0

2011-04-13 14:18:59.853 7 Clear var list:PlayDRVOBJ

2011-04-13 14:18:59.853 7 Current dialog:form[18]

2011-04-13 14:18:59.853 7 Execute Object: PlayDRVOBJ

2011-04-13 14:18:59.853 7 Enter IVR\_VPPlayDrv() function, Param: MaxDigit=1, TimeOut=0, pUserInfo=5a1f18c.

2011-04-13 14:18:59.853 7 Enter IVR\_SendVPPlayDrv() function, Param: MaxDigit=1, InterTime=5, EndDigit=12, MinDigit=1, CancelDigit=11, StartDigit=255, FirstTime=0, PlayType=2, PlaySentence=Y:/753951.wav, PlayTimes=1, PlayObject=0, SSP=-1, LocationID=65535

2011-04-13 14:18:59.853 7 IVR(1,7)->CCS(255,21): 放音收号:Resource:VP; Max:1, Min:1, Inter:5, First:0, Start:255, Cancel:11, End:12;; 指定文件: 次数 = 1, 文件名 = Y:/753951.wav

2011-04-13 14:18:59.853 7 CCS(255,21)->IVR(1,7): ICD录放音应答:放音,成功,VPDsn=65535

2011-04-13 14:18:59.853 7 Enter IVR\_ProcessEvent() function, Param: UserDialing=1, UserHookOff=0,TimeOut=1, ResAvailable=0, DeviceInfo=1,Notify=0, UserAsr=0

2011-04-13 14:18:59.853 7 [事件处理]

2011-04-13 14:19:03.900 7 CCS(255,21)->IVR(1,7): 录放音结束:MSG NOT SPECIFIED(21)

2011-04-13 14:19:03.900 7 ProcessEvent receive msg type=21, mask=249

2011-04-13 14:19:03.900 7 设备消息出口

2011-04-13 14:19:03.900 7 Get Message form Queue! Event=设备消息出口(6)

2011-04-13 14:19:03.900 7 Enter IVR\_ProcessEvent() function, Param: UserDialing=1, UserHookOff=0,TimeOut=1, ResAvailable=0, DeviceInfo=0,Notify=0, UserAsr=0

2011-04-13 14:19:03.900 7 [事件处理]

2011-04-13 14:19:08.978 7 IVR(255,65535)->IVR(1,7): MSG\_SELF\_TIMER\_TRIGGERED(166):timerid=0

2011-04-13 14:19:08.978 7 ProcessEvent receive msg type=166, mask=9

2011-04-13 14:19:08.978 7 超时出口

2011-04-13 14:19:08.978 7 Get Message form Queue! Event=超时出口(4)

2011-04-13 14:19:08.978 7 Call IVR\_VPPlayDrv() function, result=4

2011-04-13 14:19:08.978 7 Current dialog:form[18]

2011-04-13 14:19:08.978 7 Form Collect throw event: Can't get next item!Dialog exit!

2011-04-13 14:19:08.978 7 Vxmlevent process, event type: exit , event message: Can't get next item!Dialog exit!

2011-04-13 14:19:08.978 7 Interpreter throw \_exit event: unhandled event: event=exit msg=Can't get next item!Dialog exit!

2011-04-13 14:19:08.978 7 Interpreter run, catch event : unhandled event: event=exit msg=Can't get next item!Dialog exit!

2011-04-13 14:19:08.978 7 Enter VXML\_Disconnect() function, Param: cause=0

2011-04-13 14:19:08.978 7 Enter CheckUserHookMsg() function, Param: ssp=5631

2011-04-13 14:19:08.978 7 Send disconnect/release msgtype=8 ssp=5631 cause=0

2011-04-13 14:19:08.978 7 IVR(1,7)->CCS(255,21): 拆线: 原因: 正常拆线

2011-04-13 14:19:08.978 7 CCS(255,21)->IVR(1,7): 释放呼叫: 原因: 正常拆线

流程执行完毕。IVR给CCS发送拆除呼叫，CCS返回呼叫释放完毕

2011-04-13 14:19:08.978 7 CATCH event=unhandled event: event=exit msg=Can't get next item!Dialog exit!,Dialog end.

2011-04-13 14:19:08.978 7 ParseTime=0 ParseCount=2 GetDocTime=0 GetDocCount=2 WaitTime=0 WaitCountCount=0

2011-04-13 14:19:08.978 7 delete::VxmlDocument 0x4ff5118, E:\flow\vxml\PlayRecord.vxml?

2011-04-13 14:19:08.978 7 Interpreter end delete last document tree URL=E:\flow\vxml\PlayRecord.vxml?

2011-04-13 14:19:08.978 7 ======================XML\_CALL\_END======================

## GSL流程相关呼叫日志解释

2011-11-15 11:21:17.447 1 ======================GSL\_CALL\_BEGIN======================

2011-11-15 11:21:17.447 1 GSL MAXID=13, MODIFIED TIME=2011-07-14 11:57:15, SCEVersion=ICDV300R005C15#

2011-11-15 11:21:17.447 1 CallInfo TrunkNO=0,CLI=4779798,CLD=223,CLOD=12345678,

2011-11-15 11:21:17.447 1 sender\_handle=0,sender\_dsn=1,receiver\_handle=255,receiver\_dsn=1,

2011-11-15 11:21:17.447 1 AccessCode=223 Exec flow E:/flow/playvoice2.gsl Version=ICDV300R005C15SPC004 OS=Windows

2011-11-15 11:21:17.447 1 Monitor info ICDDebug=0 icdinsider=0 DebugMode=0 TraceFlag=0

上面是到达IVR呼叫的基本信息，SCEVersion 表示的是生成当前gsl使用的SCE版本，CLI表示主叫，CLOD表示原始被叫（即排队机发送给平台的Setup中的org\_CLD字段，并不是IE信息中的原始被叫）

**注意CLD并不是被叫号码而是流程接入码**，ivr日志中显示的被叫号码不是用户拨入的被叫号码，而是流程接入码+后续接入码。比如本例中呼入的被叫号码是100212，匹配到平台配置的被叫1002，对应的流程接入码是6677020，则ivr打印的CLD字段就是667702012。如果流程想获得用户拨入的被叫号码，需要调用接口“获取呼叫详情cell”，具体方法请参看GSL流程开发指南。

AccessCode表示是流程接入码，Flow表示是当前接入码对应的流程

Version表示是当前IVR的版本号，OS表示是当前IVR所在的操作系统

另外上面的“ 1” 表示是当前呼叫在IVR中的ID，如果当前IVR有多个呼叫同时存在，则IVR日志中这些呼叫的日志就会互相交错，根据这个ID（注意要加上前后的空格）使用UltraEdit将相同ID的日志copy出来，就是单独的一个呼叫的日志了，方便分析。

2011-11-15 11:21:17.447 1 $4$[ANSWER3]

$4$表示当前执行的cell ID，通过这个ID可以使用SCE在流程中查找到唯一对应的cell（具体使用方法请参考SCE帮助中“查找业务构件”）。[ANSWER3]表示当前执行的cell类型是应答cell。

2011-11-15 11:21:17.447 1 Send alerting param: audio=255

2011-11-15 11:21:17.447 1 IVR(0,1)->CCS(255,1): 振铃:

IVR给CCS发送振铃消息即ACM消息，CCS会将此消息透传给排队机，因为IVR只能看到CCS看不到排队机，所以此处日志是打印不出CCS给排队机的透传操作。

2011-11-15 11:21:17.447 1 CCS(255,1)->IVR(0,1): 呼叫标识

CCS将呼叫标志发送给IVR

2011-11-15 11:21:17.447 1 RePlace CallID list: ssp=511 CallID=' 1321327277 1 0 0'

2011-11-15 11:21:17.447 1 Add CallID list: ssp=511 CallID=' 1321327277 1 0 0'

2011-11-15 11:21:17.447 1 CCS(255,1)->IVR(0,1): 通知:

CCS将排队机的Notify消息透传给IVR，同样因为IVR只能看到CCS看不到排队机，此处也是不会打印CCS接收到排队机Notify消息的步骤，

2011-11-15 11:21:17.447 1 Answer param: SSP=511, ChgInd=0, ExtraDataLen=0

2011-11-15 11:21:17.447 1 IVR(0,1)->CCS(255,1): 摘机:

IVR给CCS发送应答消息即ANM消息，CCS会将此消息透传给排队机

2011-11-15 11:21:17.447 1 CCS(255,1)->IVR(0,1): 摘机应答:

CCS将排队机的应答ACK消息透传给IVR。

2011-11-15 11:21:17.447 1 @0@成功出口

至此$4$[ANSWER3]这个cell才执行完毕，走成功出口。

2011-11-15 11:21:17.447 1 $6$[EX\_SIB\_ICDVPPLAY]

$4$[ANSWER3] cell走成功出口后执行的是放音cell，对应的cellID是6

2011-11-15 11:21:17.447 1 SetCallStatus ssp=511, DrvFlag=1

2011-11-15 11:21:17.447 1 IVR(0,1)->CCS(255,1): 放音收号:Resource:VP; Max:255, Min:1, Inter:255, First:255, Start:255, Cancel:11, End:12;; 指定文件: 次数 = 255, 文件名 = E:\wave\welcome.wav

IVR给CCS发送放音收号消息，最小收号长度是1，放音文件是E:\wave\welcome.wav，CCS会将此放音消息透传给排队机

2011-11-15 11:21:17.447 1 CCS(255,1)->IVR(0,1): ICD录放音应答:放音,成功,VPDsn=65535 ET\_ICD\_Global\_Reason\_IE: ErrCode=301989889

CCS将排队机返回的放音成功消息透传给IVR，注意，此处表示放音开始，从此用户就可以听到播放的语音了，并不是放音完毕

2011-11-15 11:21:17.447 1 @0@成功出口

$6$[EX\_SIB\_ICDVPPLAY] cell执行完毕，走成功出口。

2011-11-15 11:21:17.447 1 $2$[事件处理]

下一步执行的是ID为2的[事件处理] cell， 开始等待收号结果或放音结果。

2011-11-15 11:21:19.572 1 CCS(255,1)->IVR(0,1): 收号应答: Cause: First valid digit, Info: {\*1FFFFFFFFFFFFFFFFFFFFFFF\*}

CCS收到排队机的收号上报消息，将此消息透传给IVR，注意此处收号信息最多只能显示前两位，用于加密，如果收号超过两位，则后面号码都以E表示，例如如果收号信息是12345，则此处打印是{\*12EEEFFFFFFFFFFFFFFFFFFF\*}

2011-11-15 11:21:19.572 1 ProcessEvent receive msg type=15, mask=1179901

2011-11-15 11:21:19.572 1 @0@拨号出口

$2$[事件处理] cell执行完毕走拨号出口，下一个执行的cell是放音cell，ID是7，

2011-11-15 11:21:21.462 1 $7$[EX\_SIB\_ICDVPPLAY]

2011-11-15 11:21:21.462 1 CtrlCallStatus stop drv. MsgTye=19 DrvStatus=1

2011-11-15 11:21:21.462 1 Send stopAsrDrv message! StopType=1, SSP=511

2011-11-15 11:21:21.462 1 SetCallStatus ssp=511, DrvFlag=0

2011-11-15 11:21:21.462 1 IVR(0,1)->CCS(255,1): 停止放音识别收号的消息:停止的类型为1

因为前一个放音还未结束，则下发新的放音前需要停止老的放音

2011-11-15 11:21:21.462 1 CCS(255,1)->IVR(0,1): 停止识别应答消息:结果为1

CCS透传排队机的停止放音结果

2011-11-15 11:21:21.462 1 SetCallStatus ssp=511, DrvFlag=1

2011-11-15 11:21:21.462 1 IVR(0,1)->CCS(255,1): 放音收号:Resource:VP; Max:255, Min:255, Inter:255, First:255, Start:255, Cancel:11, End:12;; 指定文件: 次数 = 255, 文件名 = E:\wave\111.wav

2011-11-15 11:21:21.462 1 CCS(255,1)->IVR(0,1): ICD录放音应答:放音,成功,VPDsn=65535 ET\_ICD\_Global\_Reason\_IE: ErrCode=301989889

2011-11-15 11:21:21.462 1 @0@成功出口

新的放音开始了

2011-11-15 11:21:21.462 1 $8$[EX\_SIB\_WAIT\_EVENT]

执行等待事件cell，对应的cell ID是8

2011-11-15 11:21:25.510 1 CCS(255,1)->IVR(0,1): 录放音结束:MSG NOT SPECIFIED(21)

CCS收到排队机的放音结束消息，将消息透传给IVR。至此新的语音播放完毕。

2011-11-15 11:21:25.510 1 ProcessEvent receive msg type=21,eventmask=-1,waitobject=(0,4294967295),cureventtype=10,curobjecttype=1,reasoncode=0

2011-11-15 11:21:25.510 1 CurrentSib=8: wait event result=0 event=10 object=1

2011-11-15 11:21:25.510 1 @0@成功出口

$8$[EX\_SIB\_WAIT\_EVENT] cell等待到了放音结果，走成功出口

2011-11-15 11:21:25.510 1 $3$[类型2呼叫释放]

IVR执行流程的呼叫释放cell，对应的cellID是3

2011-11-15 11:21:25.510 1 Enter CheckUserHookMsg() function, Param: ssp=511

2011-11-15 11:21:25.510 1 Send disconnect/release msgtype=8 ssp=511 cause=0

2011-11-15 11:21:25.510 1 IVR(0,1)->CCS(255,1): 拆线: 原因: 正常拆线

IVR给CCS发送呼叫拆线，CCS会将此拆线消息透传给排队机

2011-11-15 11:21:25.525 1 CCS(255,1)->IVR(0,1): 释放呼叫: 原因: 正常拆线

CCS收到排队机的呼叫释放完毕消息，将此消息透传给IVR

2011-11-15 11:21:25.525 1 @0@成功出口

2011-11-15 11:21:25.525 1 流程结束

呼叫释放cell走成功出口，流程执行完毕。

2011-11-15 11:21:25.525 1 ======================GSL\_CALL\_END======================

# IVR Trace日志解释

IVR Trace日志的格式和IVR呼叫日志格式相同，不再赘述。

# IVR日志问题分析以及疑问解答

## Ivr运行相关

### IVR启动失败

1. 日志中有打印 “Cannot create thread XXX”

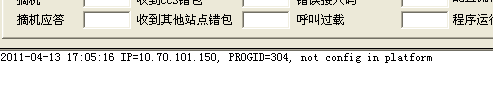
系统中允许每个进程中的最大线程数太小了，需要配置为1200

### IVR无法连接CCS

IVR运行日志中打印有如下错误：

Login CCS call CccLogon rc rc=132

同时IVR监控界面会打印如下信息



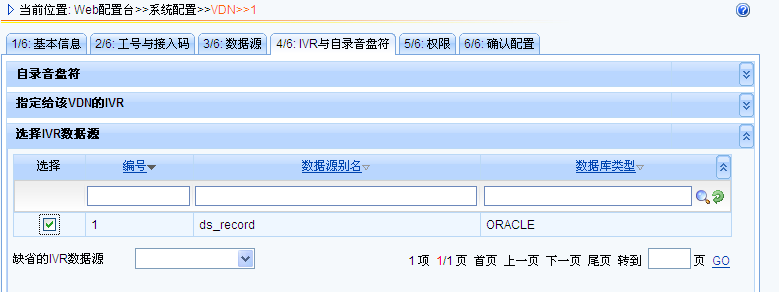
需要检查平台系统配置中IVR相关配置，是否配置了当前的这个IVR。

IVR运行日志中打印如下错误

Login CCS call CccLogon rc rc=104

需要检查一下CCS是否运行正常，如果CCS运行正常，则需要检查一下ivr所在服务器，如果是win操作系统，则需要检查注册表如下目录HKEY\_LOCAL\_MACHINE\SOFTWARE\Huawei\CCS中的MainIP和BACKIP项配置的是否是CCS的IP地址

UNIX&Linux操作系统，则需要检查平台用户下icddir/config/icdpub.ini文件中[CCS]项下的主备CCS IP地址是否正确。



## 放音收号相关

### TTS放音失败。

现场版本是B16X之前

检查hosttype配置是否正确，win在注册表HKEY\_LOCAL\_MACHINE\SOFTWARE\Huawei\IVR中，UNIX/Linux在icddir/config/icdpub.ini中。该值为0表示ACD主机，1表示为AIP或URP, 2表示为UAP主机。

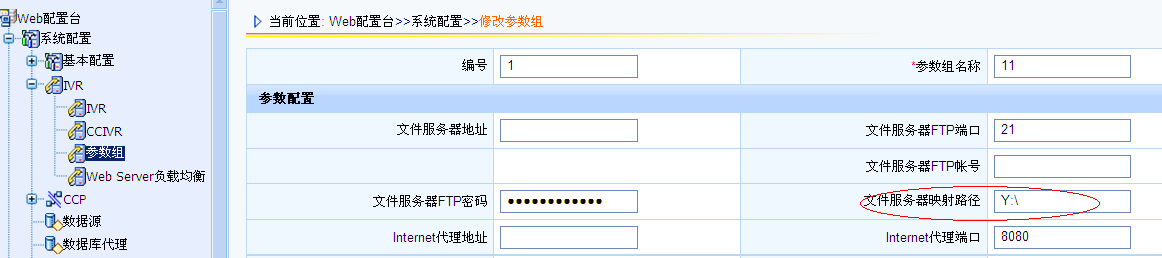
现场版本是B16X之后

检查ctiserver版本是否和排队机版本匹配（for AIP、for UAP）。

另外如果排队机版本为AIP版本或URP8100版本，则 “系统配置－CCS/媒体服务器－CTI相关配置”页面中“是否具有主机版本协商”配置项必须选择为“是”。

如果要播放的TTS字符串长度大于128字节

则IVR会将IVR会将放音内容在文件服务器上生成txt文档，然后通知排队机去播放对应的txt文件。需要在系统配置台中检查此IVR是否配置了参数组，



而且此参数组还要分配给对应的ivr



如果IVR没有分配参数组，则呼叫日志中会打印如下类似信息，可以看到文件路径缺了个盘符。

Open tts file /tts/10/CI25.txt

IVR能够使用参数组中“文件服务器映射路径”配置的盘符读写文件服务器。

如下日志就表示IVR按照配置的路径无法访问。

Open tts file F:/tts/10/CI25.txt failed

fwrite TTS file F:/tts/10/CI25.txt error.

修改完“文件服务器映射路径”需要重启IVR，因为此配置是在IVR启动之初读取得。

如果文件服务器对应路径下已经有对应的文件生成，则需要看一下排队机侧配置的访问文件服务器的盘符是否和IVR相同，排队机是否能按照此共享盘符路径访问txt文件。

如果排队机按照此盘符路径访问文件服务器正常，就需要联系排队机侧看一下为什么放音失败了。

### Menu音失败。

本来要进行直接放音的，结果IVR申请了一个VP，将VP和呼叫资源连接，然后再对VP放音。IVR的一个问题CTIPLATFORM-554

### 会场通道收号，排队返回了收号结果，但是业务获取不到

GSL流程中，会场通道收号cell成功后，后面的事件处理cell必须将通知接口连上，才能会理会场相关事件。3.5版本中可以使用“事件等待cell”和“事件获取cell”来实现。

### 收任意号码

如果想收任意键，则“EndDigit”、“CancelDigit”参数设置为0～9、\*、#之外的其他值即可

标准vxml中，可以设置最大收号数为1即可，若不能设置为1则在和ivr同级目录中有一个vxml目录，修改此vxml目录中的digits.gram

,添加信息如下

<digit> = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | "#" | "\*";

然后修改流程脚本，在对应的放音<field>中，增加属性“termchar”，其值不能是数字或者#,\*如下例子，

<field name="test" type="digits">

<property name="termchar" value="r">

<property name="inputmodes" value="dtmf"/>

<property name="maxdigit" value="2"/>

<prompt> Please say the name of a state. </prompt>

<filled>

<log>input value <value expr="test"/>?</log>

<if cond="test == '#'">

<prompt> 111 </prompt>

</if>

</filled>

</field>

然后重启IVR

## ASR相关问题

ASR实现，基本ASR语法文件是排队机自动加载的业务不用关注，但是业务生成的语法文件，必须要在业务中指定。

IVR解析业务时，发现业务指定了语法文件，则去查询此语法文件（包括全路径）是否曾被加载过，如果没有被加载过，则会将此文件复制到文件服务器的/grammar目录，重命名为dgxxxx，并通知排队机去文件服务器对应路径加载。排队机加载成功后才能进行正常的ASR收号。

### ASR功能失败

查看日志中有如下记录，说明

2011-01-06 14:05:15.000 262 Begin Add Dynamic Grammar to VP.

2011-01-06 14:05:15.000 262 IVR(8,262)->CCS(255,727): 加载动态语法消息

2011-01-06 14:05:15.125 262 CCS(255,727)->IVR(8,262): 加载动态语法的应答消息:加载动态语法的结果为0

2011-01-06 14:05:15.125 262 Interpreter send new DGid, 1 times load dynamic grammar failed!

2011-03-31 10:23:44.019 898 $2$[EX\_SIB\_DRV\_ASR]

2011-03-31 10:23:44.019 898 Copy file=c:/asrgrammer.gsl to destination file=Y:/grammar/dg1017540 fail!

2011-03-31 10:23:44.019 898 Get and Load the 0 GSL voice grammar fail, fileName=c:\asrgrammer.gsl !

2011-03-31 10:23:44.019 898 @1@失败出口

### VXML业务识别和收号同时启动，收号失败

VXML业务使用<field>元素同时启动识别和收号的时候，收号结果失败

业务1：

<property name="inputmodes" value="dtmf voice"/>

<field name="secondmain" type="digits">

业务2：

<property name="inputmodes" value="dtmf voice"/>

<field name="secondmain">

**业务1对应的呼叫日志：**

2009-11-23 09:51:13.243 3 Interpreter gather active DGid, active grammar id: 16842747, grammar file name: dg100number0fffb

2009-11-23 09:51:13.243 3 Interpreter gather active DGid, active grammar id: 16842752, grammar file name: dg1010000

2009-11-23 09:51:17.306 3 [事件处理]

2009-11-23 09:51:20.806 3 CCS(255,191)->IVR(0,3): 录放音结束:

2009-11-23 09:51:20.806 3 ProcessEvent receive msg type=21, mask=1048827

2009-11-23 09:51:20.806 3 Get MSG\_SSP\_SCP\_PLAYRECORD\_RES, but the result is NOT success! msg.result=36

2009-11-23 09:51:20.806 3 设备消息出口

2009-11-23 09:51:20.806 3 Get Message form Queue! Event=设备消息出口(9)

2009-11-23 09:51:20.806 3 Play\_Drv\_Asr return value = 9,Left <prompt> count= 0

2009-11-23 09:51:20.806 3 CI Speak&Input break for now get some input, so break to stop play

2009-11-23 09:51:20.806 3 <===SpeakInputResult=9 input:

2009-11-23 09:51:20.806 3 Platform speak Input result wrong, throw event: ERROR: Error occurs on VP when prompting, error.platform.vperror!

**业务2对应的呼叫日志：**

2009-11-24 16:59:40.891 10 IVR(0,10)->CCS(255,37): 放音收号消息:放音识别收号的命令类型为15; VP 放音 收号 语音识别, FileName=x:\voice\input\2.wav; Max:1, Min:1, Inter:5, First:0, Start:255, Cancel:255, End:255;

2009-11-24 16:59:41.032 10 CCS(255,37)->IVR(0,10): SP\_START\_DTMF\_ASR\_ACK result: 成功, vp dsn: 1024

2009-11-24 16:59:41.032 10 Enter IVR\_ProcessEvent() function, Param: UserDialing=1, UserHookOff=1,TimeOut=1, ResAvailable=0, DeviceInfo=1,Notify=0, UserAsr=1

2009-11-24 16:59:41.032 10 [事件处理]

2009-11-24 16:59:44.782 10 CCS(255,37)->IVR(0,10): 收号应答: Cause: Max number, Info: {\*1FEEEEEE0000000000000000\*}

2009-11-24 16:59:44.782 10 ProcessEvent receive msg type=15, mask=1048827

2009-11-24 16:59:44.813 10 CI Speak&Input break for now get some input, so break to stop play

2009-11-24 16:59:44.813 10 <===SpeakInputResult=0 input: 1\*\*\*0 InputType=1

2009-11-24 16:59:44.813 10 Interpreter process input result: Nomatch user input:1

2009-11-24 16:59:44.813 10 Vxmlevent process, event type: nomatch , event message: 1

对于业务1：

这样的写法表示启动识别收号，且识别收号的语法都使用builtin::digits的静态语法。而实际用户的需求并没不需要识别的digits语法。再则目前对于bultin静态语法，UAP暂无解决方案

对于业务2：

这样的写法表示启动识别收号，但识别收号都需要使用动态语法。一般用户都会指定识别的动态库语法，而忘记指定收号的动态语法，所以造成IVR做结果匹配的时候失败。

附：

VXML静态语法ID编码规则

基值包括（低四位）

boolean 0xffff

date 0xfffe

digits 0xfffd

currency 0xfffc

number 0xfffb

phone 0xfffa

time 0xfff9

help 0xfff8

动态值（高四位）

0x01xx(xx:表示语种ID值)

例如中文的number：0x01000fffb

对于业务1：待UAP支持静态语法后才能解决

对于业务2：增加<grammar>指定一个收号的动态语法

附：

收号动态语法文件

### 语法更新

流程脚本中写明语法文件的访问地址，在文件名后写一个动态的字符串，此字符串和语法文件名一起拼成ivr访问的语法文件名。

因为流程可以根据业务设置动态字符串，而动态字符串变化后，ivr访问的语法文件名便掉了，所以IVR就会重新加载语法文件了。

Vxml脚本必须从webserver上获取。

如果语法文件名为宝山.grxml，则脚本中写的文件名是<http://...(对应的http>访问路径)/ 宝山.grxml?XXXXX.（注意必须使用http模式路径，共享盘符模式无法实现动态语法）

XXX即为文件的附加属性参数，就象我们访问普通的动态页面一样。

然后脚本中需要有方法修改这个文件的动态属性。比如从数据库获取某值，或根据当前系统日期等等，根据现场业务而定。

这样就可以不用修改vxml脚本，不用修改语法文件名而实现动态加载。

### VXML业务不能获取到识别结果（可信度）

VXML业务不能获取到识别结果，但排队机已经上报识别结果

IVR呼叫日志有以下信息

“Platform Wait For Input, throw event: Condidence below this threshold”

当UAP上报的识别结果里携带的可信度信息低于业务设置的值（平台默认为0.3/30%），则IVR就会抛弃掉这个识别结果返回用户"nomatch"事件

（识别可信度是由ASR服务器决定的，只是在Nuance Asr服务器9.0.1和9.0.5版本中识别结果中带的可信度偏低，造成结果被平台拒绝）

在VXML业务脚本里指定识别可信度的最低阈值

<property name="confidence" value="0.05"/> 范围 0～1 默认为0.3即可信度为30%

附：



## 呼叫信息相关

### ivr日志中显示的被叫号码

ivr日志中显示的被叫号码不是用户拨入的被叫号码，而是流程接入码+后续接入码。

比如呼入的被叫号码是123456，匹配的被叫是123，对应的流程接入码是789，则给ivr的被叫是789456

### 获取被叫号码的方法

IVR从CTIServer的Setup消息中能获得一个被叫号码VXML在session.telephone.dnis

但是此被叫号码并不是真正的排队机发过来的被叫号码，而是流程接入码+后续接入码。必须调用接口去ctiserver查询出真正的被叫。方法如下：

VXML在 QueryCallInfo

SCE在 获取呼叫详情cell

### 获取原始被叫方法

VXML从session.telephone.odnis中获取

### IVR获取或设置IE时，无法操作整数类型的IE（GSL流程）

IVR处理的一个问题，流程中绑定的整数类型IE时，只能使用一字节整数，如果是非一字节整数，则在UNIX环境下是无法操作对应的IE信息。在C06SPC005版本中解决。

### 人工台与自动业务进行数据交互

1、人工台不能正确获取到自动业务的数据

2、IVR与人工台进行数据交互，SCE业务下正常，业务修改为VXML后不正常；或VXML业务下正常，替换为SCE业务后不正常

1、第三方解包软件，对于字符串类型的变量一定要按照C语言的"0"结束符做解析。

2、注意VXML和SCE打包的时候，SCE的字符串类型定义比VXML的要少1字节。例如VXML里定义了一个字符串变量长度为8字节，则SCE里定义为7字节即可

1、修改第三方解包软件（人工台）必须对于字符串类型的数据，必须按照C语言的"0"结束符做解析

2、调整SCE/VXML业务，统一按照一种方式做处理

## 呼叫接续相关

### IVR流程发起外呼，收不到检测结果

平台处理存在问题，B163SPC001版本中解决。

### IVR跟踪日志中显示拆线原因为Disconnection due to error

当ctiserver给ivr的呼叫拆线消息中原因码ucCause字段是3：SP\_Fault\_Cause，ivr呼叫日志就会打印这个原因。

而ctiserver发出这个拆线原因码一般都是因为“是否转换排队机拆线原因码”选择为是引起。

### 呼叫释放cell中传入的拆线原因码的透传方法。

如果原因码<4则直接透传

如果原因码>4且<125则将对应的原因码减1再传给排队机。

如果原因码>125 且<254则直接透传

### 调用callout接口时，排队机返回拆线消息中时被叫忙，但是没有走被叫忙出口。

2010-04-23 10:49:11.359 268 CCS(57,20811)->IVR(232,268): 振铃:

2010-04-23 10:49:11.359 268 Received msg type=3 sender\_dsn=20811 sender\_handle=57

2010-04-23 10:49:11.359 268 Received Alerting msg ssp=5327673 audio=255

上面日志表示收到了排队机的振铃消息。

只有当收到alerting消息之前，收到排队机的拆线消息，原因码为被叫忙，才会走忙出口。

### vxml脚本，IVR呼叫日志中存在有乱码

现场安装的平台时选择语言是英文，IVR使用编码为UTF-8编码，但是运行的vxml脚本中存在有中文，中文ivr需要使用GB2312编码，造成编解码异常，中文解析为乱码。

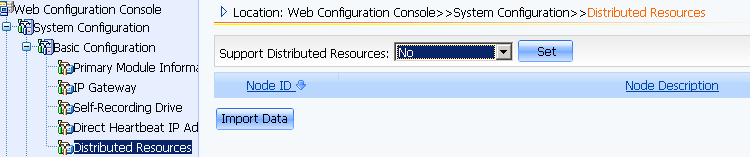
若是修改环境变量中的语言标志，则影响较大，可以在 ivr所在服务器平台用户的icddir/config目录下icdpub.ini文件中[IVR]项目下面增加  
         LocaleCode=GB2312

注意需要区分大小写，之后重启ivr

### IVR不断向排队机进行VP资源申请

现场IVR没有加载任何流程，也没有进行任何呼叫，但平台还是向UAP发送SP\_VPTsRequest消息。

1. 既然没有配置分布式系统，请检查系统配置台是否启动了支持分布式系统，如果当前没有使用分布式环境，则配置为不支持分布式，重启IVR即可。



2、从IVR系统日志也可检查

2009-11-11 10:32:27 [3520] [RUN] [INFO] [0] [DistNode].EnableFlag=0

===>0-没有启动分布式，1－启动分布式

### GSL业务不能获取CP会场放音结束事件

CP会场事件处理必须连接notify出口的问题

## 外接接口相关

### IVR调用存储过程超时

2011-09-13 10:07:28.765 152 $24$[ICD stored procedure 1]

2011-09-13 10:07:28.765 152 name={\*I\_SCE\_SetBlackWhiteList\*},num=5,dataSource=icdboss

2011-09-13 10:07:28.765 152 IVR(0,152)->Database agent:12(255,65535): Execute stored procedureExecuting stored procedure: stored procedure name {\*I\_SCE\_SetBlackWhiteList\*}, Agent node No. 10Proxy(IP=172.16.19.36:12)

2011-09-13 10:07:34.765 152 AccessCode=5758, CLD=5758733361305, name={\*I\_SCE\_SetBlackWhiteList\*},num=5,dataSource=icdboss timeout=6000 Proxy(IP=172.16.19.36 ProgID=12) TimeOut.

2011-09-13 10:07:34.765 152 Proc timeout Params: [P0]:type=21 value=7\*\*\*305(9) [P1]:type=21 value=7\*\*\*807(9) [P2]:type=50201 value=0 [P3]:type=50201 value=2 [P4]:type=50201 value=255

2011-09-13 10:07:34.765 152 AccessCode=5758, CLD=5758733361305, name={\*I\_SCE\_SetBlackWhiteList\*},num=5,dataSource=icdboss.

2011-09-13 10:07:34.765 152 Executing Procedure I\_SCE\_SetBlackWhiteList failed.

2011-09-13 10:07:34.765 152 @1@Failure outlet

### IVR无法访问对应的数据源

IVR呼叫日志上显示有如下显示

2011-04-13 16:32:31.202 2 [ICD存储过程]

2011-04-13 16:32:31.202 2 SetDBDelayTime DelaySeconds=6.

2011-04-13 16:32:31.202 2 存储过程名：{\*GetRecordFileName\*}

2011-04-13 16:32:31.202 2 name={\*GetRecordFileName\*},num=2,dataSource=ds\_record

2011-04-13 16:32:31.202 2 AccessCode=6677020, CLD=66770200, name={\*GetRecordFileName\*},num=2,dataSource=ds\_record , **no proxy loaded datasource**.

2011-04-13 16:32:31.202 2 call VXML\_DBStoreProc() function, result=0

或

2011-04-13 16:30:55.624 0 [ICD存储过程]

2011-04-13 16:30:55.624 0 SetDBDelayTime DelaySeconds=6.

2011-04-13 16:30:55.624 0 存储过程名：{\*GetRecordFileName\*}

2011-04-13 16:30:55.624 0 name={\*GetRecordFileName\*},num=2,dataSource=ds\_record

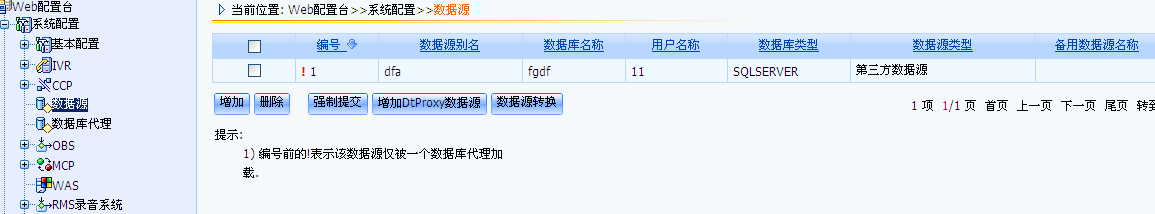
2011-04-13 16:30:55.624 0 AccessCode=6677020, CLD=66770200, name={\*GetRecordFileName\*},num=2,dataSource=ds\_record.

2011-04-13 16:30:55.624 0 **can not find datasource** ds\_record **from vdn.config .**

2011-04-13 16:30:55.624 0 call VXML\_DBStoreProc() function, result=0

或者IVR日志中显示加载的数据源没有业务需要的数据源。

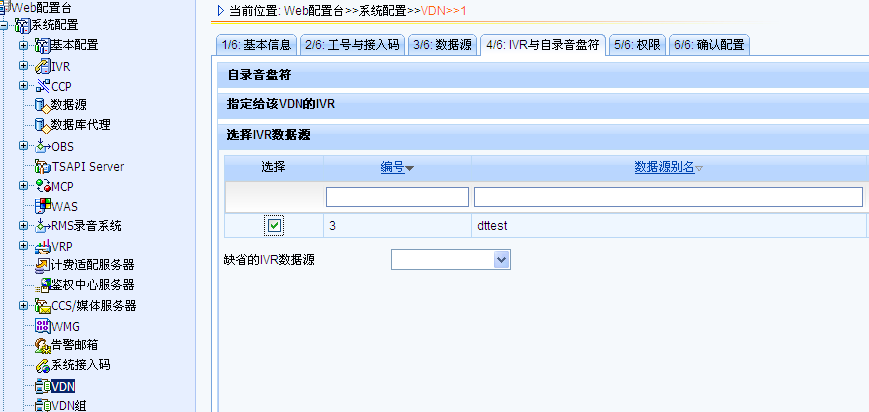
1、请在平台配置台““系统配置－数据源”配置对应的数据源，并且检查加载此数据源的aplogic是否已经运行，使用系统监控台检查对应的Aplogic是否加载此数据源成功。



2、请在平台配置台“系统配置－VDN－数据源－选择第三方数据源”，将对应的数据源选上



3、请在平台配置台“系统配置－VDN－IVR与自录音盘符－选择IVR数据源”，将对应的数据源选上。



### 加载第三方动态库失败

1. 打印Error message:/home/icd/icddir/lib/XXXX.so: wrong ELF class: ELFCLASS64

动态库必须使用二级制模式进行下载和上传。

64位操作系统下，动态库编译时必须增加选项-m32。

### SCE业务连接邮件服务器走失败出口

SCE业务使用【连接邮件服务器】Cell，但走失败出口（用outlook对账户进行测试，收发邮件正常，说明邮件账户不存在问题）

从debug日志可以看到

2009-10-12 09:28:56.797 138 $87$[连接邮件服务器]

2009-10-12 09:28:57.000 138 Connect mail server 10.80.19.3:25, return 50.

2009-10-12 09:28:57.016 138 @1@失败出口

1. 对于邮件操作，出现问题的时候建议使用telenet命令进行远程测试，确认邮件服务器是处于正常工作状态，如有问题，telenet也会给出原因码的解释

telnet 10.70.101.196 110

+OK ArGoSoft Mail Server Pro for WinNT/2000/XP, Version 1.8 (1.8.4.3)

1. 注意常用的邮件服务端口

发送邮件端口：25(SMTP)

接收邮件端口：110（POP3）

3、回到这个业务

（1）连接邮件服务器给出的地址为10.80.19.3:25，从端口看应该是发送邮件。但【连接邮件服务器】使用的是接收邮件POP3接口，业务给出的地址和cell功能是冲突的

（2）如果是发送邮件无需调用【连接邮件服务器】Cell

### IVR不能携带cookie的问题

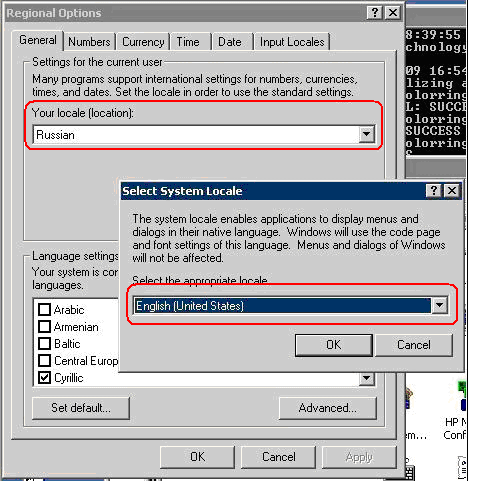
IVR每次向HTTP发送请求的时候，都不能携带上次HTTP请求传回的cookie信息，造成本次HTTP获取信息不正常

2009-09-18 10:25:55.498 0 Receive cookie:JSESSIONID=a4oGGZjSQ5bd; path=/,cookie length:31.

2009-09-18 10:25:55.498 0 The cookie time September-18-2009 03:25:55 parse failed.

由于IVR不能解析cookie time， 因此IVR不能缓存下本次HTTP的头信息，造成下次操作时不能将那些信息携带出去(该情况只在WIN操作系统下出现)

检查WIN操作系统的区域语种设置，如果为非英语设置，请务必修改为英语语种的区域设置，修改后无需重启IVR，问题即可得到解决



### IVR调用jave接口失败

IVR只能调用静态的java接口

### IVR缓存的超时时间

Software\\HUAWEI\\IVR\\ DefExpireTime单位为分钟，默认为10分钟。

若是本地文件，默认超时时间为1分钟，无法配置。

## 其它

# IVR自动退出机制日志分析

## 大量呼叫和CTIServer握手超时的日志

### 触发原因

当呼叫在IVR上处理时，IVR和CTIServer之间存在有此呼叫的握手，当IVR 90s未收到CTIServer发送过来的关于此呼叫的握手，则IVR会标记此呼叫为异常呼叫。此时IVR监控界面上，此呼叫的控制块显示为红色。

当异常的呼叫数超过当前IVR支持的最大呼叫数之30％，IVR会自动退出。

当异常的呼叫数超过当前IVR支持的最大呼叫数之10％，且超过当前活动呼叫数的90％，IVR会自动退出。

IVR和CTIServer之间呼叫握手超时，一般情况下都是因为ctiserver收到了排队机的呼叫拆线消息后，通知ivr呼叫释放，但是ivr在进行某些耗时较长之类的操作，或内部存在有死循环，没有处理呼叫拆线事件，导致呼叫一直在ivr业务中存在。

### 检查方法

**IVR运行日志检查**

检查出现问题时间附近IVR的运行日志IVRXXX.log(XXX表示IVR当前的ProgId，例如301、302等)观察如下IVR状态日志，此日志每隔5分钟打印一次，记录IVR的运行状态。其中ExceedMaxTalkTime表示因为呼叫存在的时间超过了最大通话时间标志为吊死的呼叫数。ShakehandOvertime表示因为和ctiserver握手断连标志为吊死的呼叫数。

ActiveCall=6 ,ShakehandOvertime=363, ExceedMaxTalkTime= 0 CallIn=288 DisconnectNoResource=0

如果吊死的呼叫数过多，ivr会自动退出，同时会有日志打印出呼叫是在那个流程中吊死。如下日志显示在ICD\_Fast\_YH\_V3.0Ecmtj.GSL流程中吊死的呼叫有3个，在ICD\_Transfer\_nolimit\_V3.0Ecmtj.GSL流程中吊死呼叫有241个。

2010-04-29 21:34:51 [70076] [RUN] [INFO] [0] The Dead CI Info: IVR302, TotalDeadCount = 245. F:/1/icdmmp/config/flow/ICD\_Fast\_YH\_V3.0Ecmtj.GSL, num=3; F:/1/icdmmp/config/flow/ICD\_Transfer\_nolimit\_V3.0Ecmtj.GSL, num=241;

**IVR呼叫日志检查**

当ivr发现有呼叫吊死时，ivr会主动在呼叫日志ivrXXX\_call.log中将出现问题呼叫的流程轨迹打印出来。

如下日志表示，一个呼叫因为和ctiserver握手超时吊死。109是当前这个呼叫的标志，DelayTickCount表示这个呼叫从最近一次收到ctiserver的握手至今的时间，毫秒为单位。

2010-04-29 21:33:56.895 109 Send shakehands timeout msg time=158885652 LastReceiveReset=158795135 GetTickCount=158885652 DelayTickCount=90517

在ivr呼叫日志中查询出呼叫标志为109的呼叫轨迹如下：

2010-04-29 21:33:56.895 109 Send shakehands timeout msg time=158885652 LastReceiveReset=158795135 GetTickCount=158885652 DelayTickCount=90517

2010-04-29 21:33:56.902 109 AccessCode:10086

2010-04-29 21:33:56.902 109 Service:F:/1/icdmmp/config/flow/ICD\_Transfer\_nolimit\_V3.0Ecmtj.GSL

2010-04-29 21:33:56.903 109 TrunkNO=2243,CLI=15922009301,CLD=10086,CLOD=

2010-04-29 21:33:56.903 109 $14109$[运算] 21:32:21

2010-04-29 21:33:56.903 109 $14110$[运算] 21:32:21

2010-04-29 21:33:56.903 109 $14124$[运算] 21:32:21

2010-04-29 21:33:56.908 109 $14258$[运算] 21:32:22

… ….

2010-04-29 21:33:56.908 109 $14272$[运算] 21:32:22

2010-04-29 21:33:56.908 109 $14262$[参数入堆栈] 21:32:22

**2010-04-29 21:33:56.908 109 $14264$[资源控制] 21:32:22**

**2010-04-29 21:33:56.910 109 @2@失败出口**

2010-04-29 21:33:57.006 109 $14283$[参数入堆栈]

2010-04-29 21:33:57.006 109 @0@成功出口

2010-04-29 21:33:57.006 109 $14288$[比较]

**… …**

日志格式说明：

打印日志时间＋呼叫的标志id＋$呼叫执行的Cell索引$[cell类型名]＋执行这个cell时间

2010-04-29 21:33:56.908 109 $14255$[参数入堆栈] 21:32:22

例如上面这条日志，打印日志的时间是2010-04-29 21:33:56.908，呼叫标志id是19，呼叫执行的cell索引是14255，呼叫执行的cell类型名是“参数入堆栈”，执行这个cell的时间是21:32:22。

从上面搜索出的标志为109的呼叫轨迹可以看出打印时间在2010-04-29 21:33:56.910的日志的右侧是没有显示cell执行时间的，表示cell执行时间和日志打印时间相同。因此呼叫吊死的地方可以确认为标志红色处。

**2010-04-29 21:33:56.908 109 $14264$[资源控制] 21:32:22**

**2010-04-29 21:33:56.910 109 @2@失败出口**

21:32:22执行了cell索引为14264的cell，然后直到2010-04-29 21:33:56.910才从这个cell中出来。如果在这段时间内用户拆线，ctiserver内部会清除对应的呼叫信息，并通知ivr呼叫释放，但是ivr因为吊在了这个cell内无法处理，造成呼叫在ivr中继续存在，和ctiserver关于此呼叫握手超时。

### 解决方案

根据日志分析可以看出，当前呼叫吊死的流程是F:/1/icdmmp/config/flow/ICD\_Transfer\_nolimit\_V3.0Ecmtj.GSL，在流程中ID为14264的cell吊住引起。需要找这个流程到对应的sce文件，检查一下对应cell在进行什么操作，检查前后cell的业务逻辑。联系这个流程的开发人员确认一下对应的cell使用是否存在问题。

## 大量呼叫持续时间超长的日志

### 触发原因

平台VDN配置中，流程配置页面中有“限制最大通话时间”配置项，可以通过此配置项配置每个呼叫在当前流程上允许运行的最大时间。如果这个时间配置为0，则IVR会默认最大超时时间为3600s

当呼叫在IVR上执行的时间超过允许的最大超时时间，则IVR会标记此呼叫为异常呼叫。此时IVR监控界面上，此呼叫的控制块显示为黄色。

当异常的呼叫数超过当前IVR支持的最大呼叫数之30％，IVR会自动退出。

当异常的呼叫数超过当前IVR支持的最大呼叫数之10％，且超过当前活动呼叫数的90％，IVR会自动退出。

一般情况下呼叫在流程中持续时间超常都是因为流程中存在有死循环引起。

### 检查方法

和2.1大量呼叫和ctiserver握手超时相同。

IVR发现当前有呼叫查过了流程的最大通话时间，就打出1M左右此呼叫的debug级别的流程日志。从下面的ivr\_call.log日志可以看到，呼叫在流程/home/ivr/icddir/flow/MainSCE\_Flow.GSL中吊死，在流程中从ID为$511$的cell执行到ID为$265$的call后，又从$511$开始执行，如此循环。

2011-11-11 12:06:17.182 173 Set ExceedMaxTalkTime flag BeginTime=15606505 CurTime=19208199 MaxTime=3600000

2011-11-11 12:06:17.182 173 Set CallTimeTooLong flag CallBeginTime=15606505 CurTime=19208200 CurSibIndex=533

2011-11-11 12:06:17.182 173 AccessCode:8828

2011-11-11 12:06:17.182 173 **Service:/home/ivr/icddir/flow/MainSCE\_Flow.GSL**

2011-11-11 12:06:17.182 173 TrunkNO=16383,CLI=3347907778,CLD=8828,CLOD=3367978692

2011-11-11 12:06:17.182 173 **$511$**[EXTENDED\_SIB\_ALGORITHUM] 12:6:16

2011-11-11 12:06:17.182 173 $277$[Parameter incoming stack] 12:6:16

2011-11-11 12:06:17.182 173 $285$[Operation] 12:6:16

2011-11-11 12:06:17.182 173 $278$[Operation] 12:6:16

2011-11-11 12:06:17.182 173 $279$[Operation] 12:6:16

2011-11-11 12:06:17.182 173 $280$[Operation] 12:6:16

2011-11-11 12:06:17.182 173 $281$[Operation] 12:6:16

2011-11-11 12:06:17.182 173 $283$[Operation] 12:6:16

2011-11-11 12:06:17.182 173 $284$[Operation] 12:6:16

2011-11-11 12:06:17.182 173 $269$[Parameter incoming stack] 12:6:16

2011-11-11 12:06:17.182 173 $270$[Playing and number receiving] 12:6:16

2011-11-11 12:06:17.182 173 $271$[Event processing] 12:6:16

2011-11-11 12:06:17.182 173 $531$[Parameter incoming stack] 12:6:16

2011-11-11 12:06:17.182 173 $532$[Data packet operation] 12:6:16

2011-11-11 12:06:17.182 173 $533$[Access manual station data] 12:6:16

2011-11-11 12:06:17.182 173 $534$[Request for route] 12:6:17

2011-11-11 12:06:17.182 173 **$265$**[Compare] 12:6:17

2011-11-11 12:06:17.182 173 **$511$**[EXTENDED\_SIB\_ALGORITHUM] 12:6:17

2011-11-11 12:06:17.182 173 $277$[Parameter incoming stack] 12:6:17

2011-11-11 12:06:17.182 173 $285$[Operation] 12:6:17

2011-11-11 12:06:17.182 173 $278$[Operation] 12:6:17

2011-11-11 12:06:17.182 173 $279$[Operation] 12:6:17

2011-11-11 12:06:17.182 173 $280$[Operation] 12:6:17

2011-11-11 12:06:17.182 173 $281$[Operation] 12:6:17

2011-11-11 12:06:17.182 173 $283$[Operation] 12:6:17

2011-11-11 12:06:17.182 173 $284$[Operation] 12:6:17

2011-11-11 12:06:17.182 173 $269$[Parameter incoming stack] 12:6:17

2011-11-11 12:06:17.182 173 $270$[Playing and number receiving] 12:6:17

2011-11-11 12:06:17.182 173 $271$[Event processing] 12:6:17

2011-11-11 12:06:17.182 173 $531$[Parameter incoming stack] 12:6:17

2011-11-11 12:06:17.182 173 $532$[Data packet operation] 12:6:17

2011-11-11 12:06:17.182 173 $533$[Access manual station data] 12:6:17

2011-11-11 12:06:17.182 173 $534$[Request for route] 12:6:17

2011-11-11 12:06:17.182 173 **$265$**[Compare] 12:6:17

2011-11-11 12:06:17.182 173 **$511$**[EXTENDED\_SIB\_ALGORITHUM] 12:6:17

2011-11-11 12:06:17.182 173 $277$[Parameter incoming stack] 12:6:17

2011-11-11 12:06:17.182 173 $285$[Operation] 12:6:17

2011-11-11 12:06:17.182 173 $278$[Operation] 12:6:17

2011-11-11 12:06:17.182 173 $279$[Operation] 12:6:17

2011-11-11 12:06:17.182 173 $280$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $281$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $283$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $284$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $269$[Parameter incoming stack] 12:6:17

2011-11-11 12:06:17.183 173 $270$[Playing and number receiving] 12:6:17

2011-11-11 12:06:17.183 173 $271$[Event processing] 12:6:17

2011-11-11 12:06:17.183 173 $531$[Parameter incoming stack] 12:6:17

2011-11-11 12:06:17.183 173 $532$[Data packet operation] 12:6:17

2011-11-11 12:06:17.183 173 $533$[Access manual station data] 12:6:17

2011-11-11 12:06:17.183 173 $534$[Request for route] 12:6:17

2011-11-11 12:06:17.183 173 **$265$**[Compare] 12:6:17

2011-11-11 12:06:17.183 173 **$511$**[EXTENDED\_SIB\_ALGORITHUM] 12:6:17

2011-11-11 12:06:17.183 173 $277$[Parameter incoming stack] 12:6:17

2011-11-11 12:06:17.183 173 $285$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $278$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $279$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $280$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $281$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $283$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $284$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $269$[Parameter incoming stack] 12:6:17

2011-11-11 12:06:17.183 173 $270$[Playing and number receiving] 12:6:17

2011-11-11 12:06:17.183 173 $271$[Event processing] 12:6:17

2011-11-11 12:06:17.183 173 $531$[Parameter incoming stack] 12:6:17

2011-11-11 12:06:17.183 173 $532$[Data packet operation] 12:6:17

2011-11-11 12:06:17.183 173 $533$[Access manual station data] 12:6:17

2011-11-11 12:06:17.183 173 $534$[Request for route] 12:6:17

2011-11-11 12:06:17.183 173 **$265$[**Compare] 12:6:17

2011-11-11 12:06:17.183 173 **$511$**[EXTENDED\_SIB\_ALGORITHUM] 12:6:17

2011-11-11 12:06:17.183 173 $277$[Parameter incoming stack] 12:6:17

2011-11-11 12:06:17.183 173 $285$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $278$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $279$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $280$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $281$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $283$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $284$[Operation] 12:6:17

2011-11-11 12:06:17.183 173 $269$[Parameter incoming stack] 12:6:17

2011-11-11 12:06:17.183 173 $270$[Playing and number receiving] 12:6:17

2011-11-11 12:06:17.183 173 $271$[Event processing] 12:6:17

2011-11-11 12:06:17.183 173 $531$[Parameter incoming stack] 12:6:17

2011-11-11 12:06:17.183 173 $532$[Data packet operation] 12:6:17

2011-11-11 12:06:17.183 173 $533$[Access manual station data] 12:6:17

2011-11-11 12:06:17.183 173 $534$[Request for route] 12:6:17

2011-11-11 12:06:17.183 173 $265$[Compare] 12:6:17

2011-11-11 12:06:17.183 173 **$511$**[EXTENDED\_SIB\_ALGORITHUM] 12:6:17

2011-11-11 12:06:17.183 173 $277$[Parameter incoming stack] 12:6:17

2011-11-11 12:06:17.183 173 $285$[Operation] 12:6:17

2011-11-11 12:06:17.204 173 Setting manual station data fails,Ret:1.

2011-11-11 12:06:17.204 173 @1@Failure outlet

### 解决方案

和2.1大量呼叫和ctiserver握手超时相同

### 注意事项

如果“限制最大通话时间”配置项不为0，则当呼叫持续时间超过最大允许的通话时间后，CTIServer会主动给IVR发送拆线消息终止呼叫。

## 动态库内存泄漏导致ivr退出日志

### 触发原因

IVR加载的流程中调用了第三方动态库，而此动态库中如果存在内存泄漏的话，会出现大量内存被占用，因为此动态库是由IVR加载的，所以就表现出了IVR占用的内存大量增长，当mds检测到IVR占用的内存超过规定的阀值1000M后，mds就会重启IVR，并生成core文件。

### 检查方法

从icddir/log/mds.log日志中可以看到如下记录，

2011-11-09 16:21:58 [1127] [RUN] [INFO] [5115] used memory(M) exceed limit ServerName=ui,ProgID=304, MemUsed:1479, MemLimit:1000, Frequency:3

2011-11-09 16:31:58 [1127] [RUN] [INFO] [5093] Restart service for occupied memory(M) exceeding the limit(M) ServerName=ui,ProgID=304, MemUsed:1479, MemLimit:1000

2011-11-09 16:31:58 [1127] [RUN] [INFO] [5078] MDS restart server. ServerName=ui, ProgID=304

此处表明是mds检测到IVR内存超过阀值1000M，将IVR重启的。但是mds是无法检测出IVR内存超过阀值的原因。

因为IVR加载动态库后，很难区分是动态库的内存泄漏，还是IVR的内存泄漏了。所以平台提供了IVRAgent模块，将于动态库和IVR隔离，使用IVRAgent加载动态库后，IVR只是向IVRAgent发送执行消息，不再加载动态库。

如果IVRAgent加载动态库后，IVR的内存不再增张了，但是IVRAgent内存不断增长超过了阀值，则说明此处内存泄漏一定是存在动态库中。

具体IVRAgent的使用方法可以参考附录文档

### 解决方案

因为IVR是不知道动态库中内部的实现，所以对于内存泄漏问题，如果使用IVRAgent证明是第三方动态库的问题，必需要第三方动态库的开发人员分析确认。

## 动态库自身coredump的日志

### 触发原因

IVR加载的流程中调用了第三方动态库，如果第三方动态库内部操作异常，比如内存地址非法访问等操作，此时动态库就会出现coredump，因为当前动态库是由IVR加载执行的，所以表现就是IVR异常退出，且有core文件生成。

### 检查方法

1. 打开icddr/log/mds.log，可以看到出现问题时间点有如下日志：

**2011-11-15 00:16:20 [26335] [RUN] [INFO] [5163] 进程未启动或者进程意外退出. ServerName=ivr, ProgID=301, EventID=NTF\_STOPPED, CurCMD=RUNNING\_VCMD, CurRunState=RUNNING**

**2011-11-15 00:16:20 [26335] [RUN] [INFO] [5069] 发送""启动服务""指令. ServerName=ivr, ProgID=301,**

**… …**

**2011-11-15 00:16:27 [26335] [RUN] [INFO] [5047] /usr/bin/file /home/c15/icddir/bin/core.15227**

**2011-11-15 00:16:27 [26335] [RUN] [INFO] [5003] Found a coredump file. CoreFile=/home/c15/icddir/bin/core.15227, ProcName=ivr**

**2011-11-15 00:16:27 [26335] [RUN] [INFO] [5004] Move Coredump File Success. SourceFile=/home/c15/icddir/bin/core.15227, DestFile=/home/c15/icddir/corebak/core\_20111115\_001620\_ivr**

Mds检测到IVR异常退出，重启IVR，同时发现有core文件生成。根据mds日志可知异常退出时此处的IVR的ProgID是301

1. 打开icddir/log/301目录下出问题时的ivr日志。可以看到如下信息：

**callinginstance[1231] accesscode:6615 flow:calldll.gsl get sig[6] in 21**

2011-11-15 00:12:37 [11778] [RUN] [INFO] [0] \*\*\*\*\*\*\*\*\*\*ivr start\*\*\*\*\*\*\*\*\*\*

从上面日志可以看到，在IVR执行到脚本calldll.gsl中 ID为21的cell时，收到了操作系统的信号6，将IVR异常退出。

打开calldll.gsl对应的sce文件，查找ID为21的cell，若cell为调用第三方动态库cell，则说明第三方动态库异常需要动态库的开发人员分析，若是其他cell请发回给平台开发人员分析。

注意：此处只使用于gsl流程，对于vxml流程因为没有cell的概念，所以此处是无法看到执行到流程那一步的。而且即使是gsl，如果IVR退出较快也会出现没有打印这种异常日志的情况。此时就必须按照下面步骤打开core文件分析了。

3、检查平台用户下icddir/corebak目录中有ivr生成的core文件。

4、打开core文件，检查出现问题的函数。打开方法如下：

* 使用平台用户登陆
* 进入icddir/corebak目录
* Linux&HP环境下输入如下命令后回车

gdb ../bin/ivr core文件名

AIX环境下输入如下命令后回车

dbx ../bin/ivr core文件名

* 在gdb或dbx提示符环境下，输入where, 之后回车
* 观察打印出的信息，是否有如下类似的信息

</home/c15/icddir/corebak>gdb ../bin/ivr core\_20111115\_001620\_ivr

GNU gdb 6.6

Copyright (C) 2006 Free Software Foundation, Inc.

GDB is free software, covered by the GNU General Public License, and you are

welcome to change it and/or distribute copies of it under certain conditions.

Type "show copying" to see the conditions.

There is absolutely no warranty for GDB. Type "show warranty" for details.

This GDB was configured as "x86\_64-suse-linux"...

Using host libthread\_db library "/lib64/libthread\_db.so.1".

warning: Can't read pathname for load map: Input/output error.

Reading symbols from /home/c15/icdcomm/lib/libhtcpapi.so...done.

Loaded symbols for /home/c15/icdcomm/lib/libhtcpapi.so

… …

Reading symbols from /usr/lib/gconv/EUC-CN.so...done.

Loaded symbols for /usr/lib/gconv/EUC-CN.so

Reading symbols from /usr/lib/gconv/libGB.so...done.

Loaded symbols for /usr/lib/gconv/libGB.so

warning: Lowest section in system-supplied DSO at 0xffffe000 is .hash at ffffe0b4

Core was generated by `**/home/c15/icddir/bin/ivr -p 301'**.

Program terminated with signal 6, Aborted.

#0 0xffffe410 in \_\_kernel\_vsyscall ()

(gdb) where

#0 0xffffe410 in \_\_kernel\_vsyscall ()

#1 0xf73e98d0 in raise () from /lib/libc.so.6

#2 0xf73eaff3 in abort () from /lib/libc.so.6

#3 0x08201b83 in ProgramExitSIGDeal (sig=11) at ../scp/ntscp.cpp:274

#4 <signal handler called>

**#5 0xef10b708 in func2 () at testdll2.cpp:6**

#6 0x08182fc2 in **CFunctionSib::ExecFunction** (this=0xf6e755b8, pCI=0xeed82300) at ../scp/sib/testsib.cpp:9095

#7 0x08182c3b in CFunctionSib::DoVerb (this=0xf6e755b8, pCI=0xeed82300) at ../scp/sib/testsib.cpp:9009

#8 0x0809b6b7 in CCallingInstance::Run (this=0xeed82300) at ../icd\_files/icd\_callapp.cpp:2142

#9 0x08099ee7 in CCallingInstance::ThreadProc (lpParam=0xeed82300) at ../icd\_files/icd\_callapp.cpp:1464

#10 0xf74f22ab in start\_thread () from /lib/libpthread.so.0

#11 0xf7480b9e in clone () from /lib/libc.so.6

从core文件中可以明显看出是301 IVR生成的core，其中的CFunctionSib::ExecFunction

即为IVR执行动态库的调用，CFunctionSib::ExecFunction之上的就是动态库的堆栈信息了，所以可以确认，此处是执行动态库的testdll2.cpp代码中的func2时异常了。需要动态库的开发人员分析。

若堆栈信息中没有看到CFunctionSib::ExecFunction，则请将此堆栈信息发送给平台开发人员分析。

### 解决方案

若通过上面的步骤分析出是业务动态库异常，则请给对应的业务开发人员分析，否则请将对应的信息发送给平台开发人员分析。