



**Cisco Networkers**  
**2009**

January 26-29 Barcelona, Spain

# Advanced Tips and Tricks for Self-Service Application Development using Cisco Unified Call Studio

BRKUCT-3015



**Paul Tindall**

# Housekeeping

- We value your feedback- don't forget to complete your online session evaluations after each session & complete the Overall Conference Evaluation which will be available online from Thursday
- Visit the World of Solutions
- Please remember this is a 'non-smoking' venue!
- Please switch off your mobile phones
- Please make use of the recycling bins provided
- Please remember to wear your badge at all times including the Party

# Abstract

This is an advanced session which presents a collection of tips and tricks for use by Call Studio application designers and developers in performing many interesting tasks for which the solution is often not immediately apparent or may even be considered not possible. Examples include overriding ANI on transfers, accessing in the script and sending call signalling information such as UUI, and how to send Display-Name on the transferred call leg. The session also examines the internals of how the recently published CVP Standalone Outbound MakeCall capability works, and how it can be used in applications that need to deliver automated reminder and notification calls.

This session is for contact center specialists involved in service creation for Cisco Unified CVP who want to learn more about scripting using Call Studio in the Standalone deployment model. Attendees should be familiar with Cisco Unified CVP and Call Studio with either a keen interest or experience in developing applications that make use of advanced techniques.

# Summary

As just two examples, how to make outbound notification/reminder style calls and sending/receiving UUI have become frequently asked questions with regard to the CVP Standalone VoiceXML deployment model.

This session aims to demystify and explain how these and many other advanced requirements can be addressed within a solution comprising only the VoiceXML gateway/browser and CVP VoiceXML Server.

Attendees will benefit from greater understanding of how the CVP Standalone Model works and how a number of advanced techniques may be used to address the examples presented here as well as stimulating ideas for handling similar challenges that may be encountered in designing and developing self-service applications as part of an overall UC or Contact Centre solution.

The presenter is a participant in the Meet The Expert program.

# Agenda

- Brief Overview of CVP Standalone VoiceXML
- Incoming Calls and Data Flow
- How-Is-It-Done Tips and Tricks
- Optimal Prompt Caching
- Custom Script Elements

# How Do You ... ?

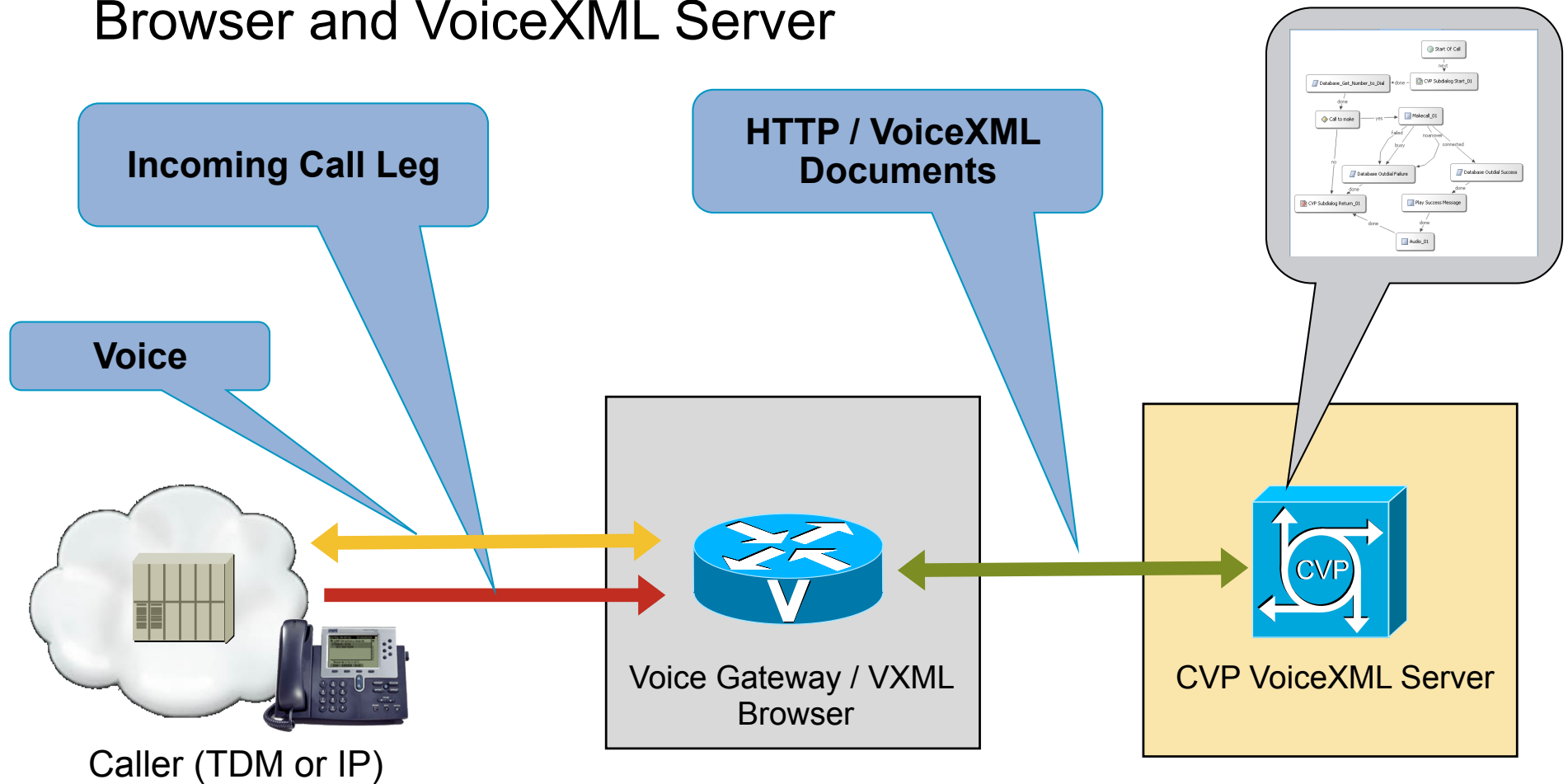
- Receive/Send Signalling Information, such as:
  - User-User Information (UUI)
  - Display-Name
  - Modified ANI
- Perform Call Control from the CVP Application
  - Accessing Gateway TCL Functionality from the CVP Application
- Make Outbound Calls
- Use Live Audio Feeds as Prompts
- Perform Fax Detection

# Agenda

- **Brief Overview of CVP Standalone VoiceXML**
- Incoming Calls and Data Flow
- How-Is-It-Done Tips and Tricks
- Optimal Prompt Caching
- Custom Script Elements

# In Very Simple Terms

IVR platform comprising IOS Voice Gateway / VoiceXML Browser and VoiceXML Server



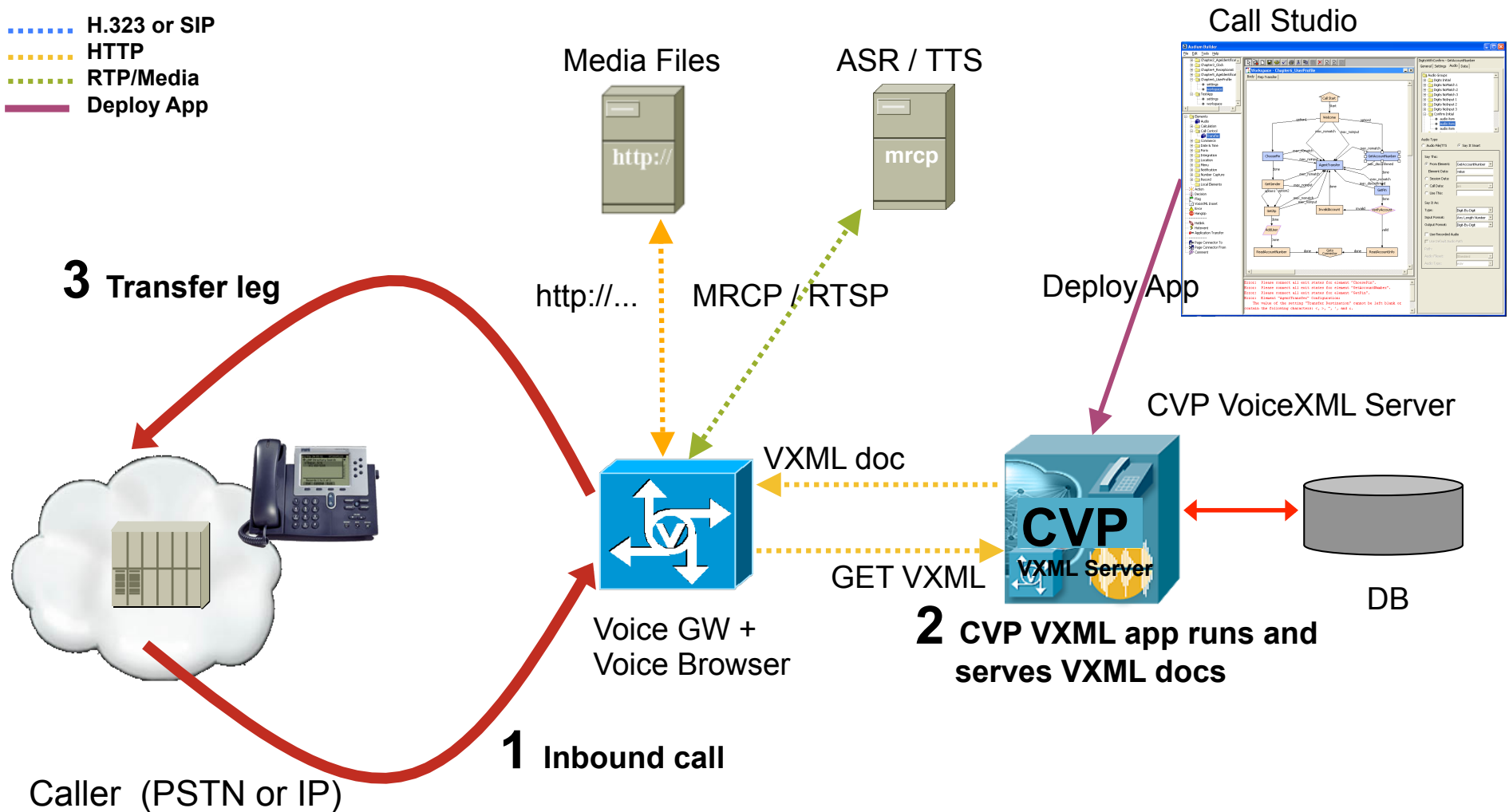


# CVP VoiceXML Standalone

- Terminates calls independently of UC Manager
- Remotely distributed call delivery and IVR served by centralised applications, avoiding media across WAN
- For advanced speech applications
- Offers back-end integration flexibility
- Customisation and element reusability
- Scalability using VXML Gateway and Server farms
- Use with TDM, SIP or H.323
- Transfers via VXML Gateway for TDM / VoIP combinations including TBCT & SIP REFER redirects

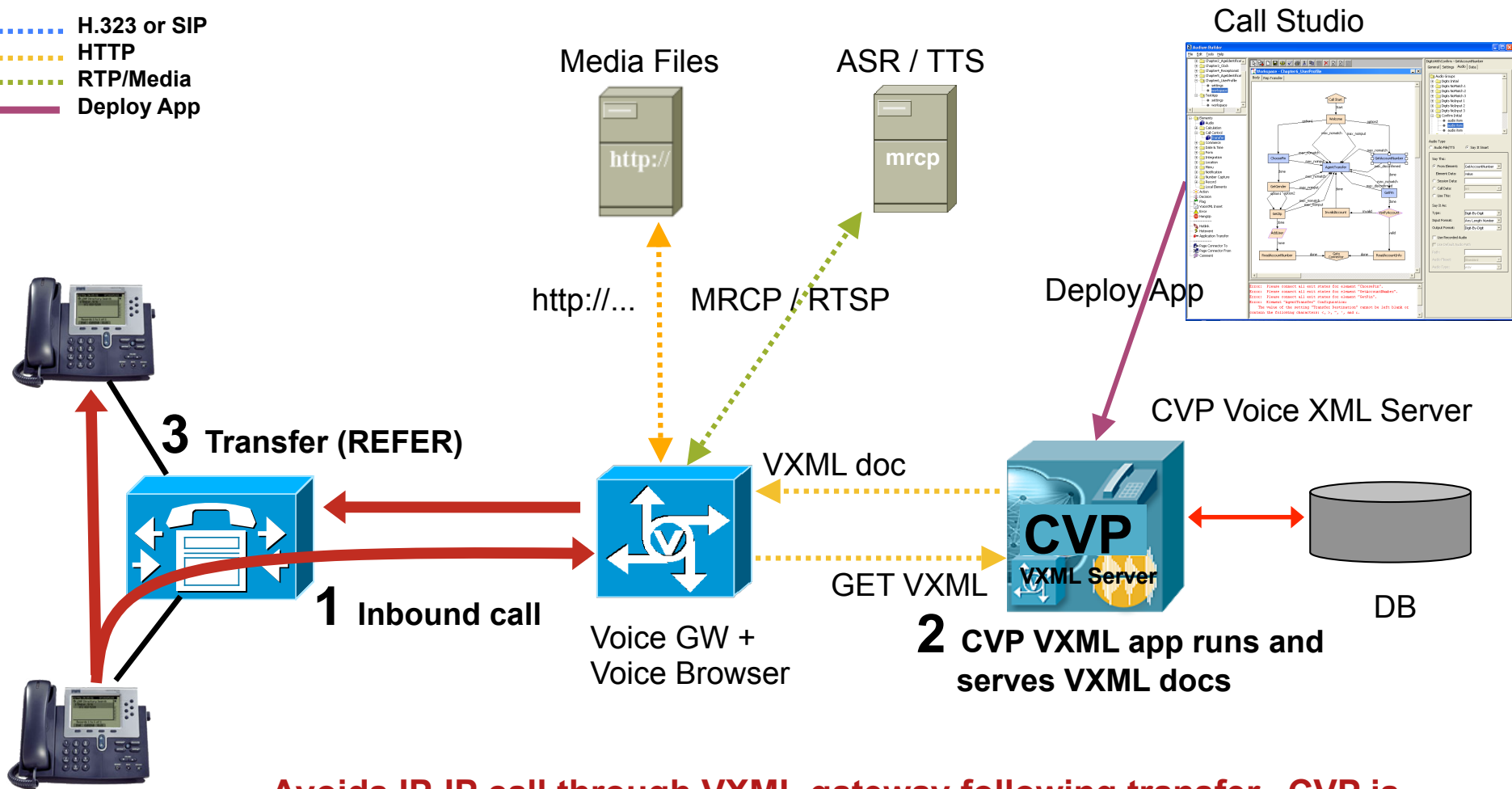
# Standalone VoiceXML Transfers

- ..... H.323 or SIP
- ..... HTTP
- ..... RTP/Media
- Deploy App



# Standalone VoiceXML Transfers (SIP REFER)

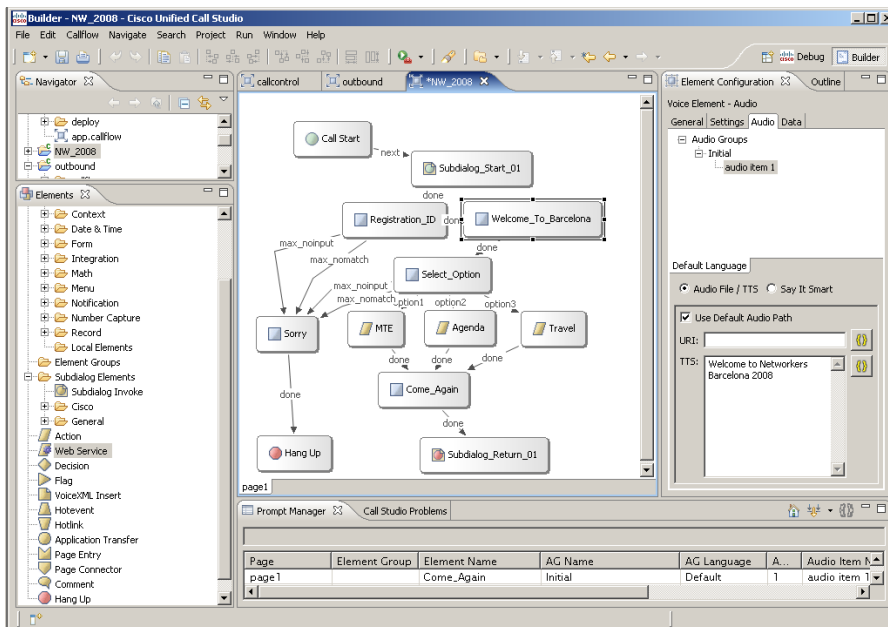
- ..... H.323 or SIP
- ..... HTTP
- ..... RTP/Media
- Deploy App



**Avoids IP-IP call through VXML gateway following transfer. CVP is dropped out of the call but note that this transfer method still needs an IOS feature set which includes IP-IP Gateway.**

# Cisco Unified Call Studio For Scripting

- Eclipse based development studio for CVP VoiceXML applications
- Use for full self-service interaction
- Dynamic VoiceXML at run-time
- ASR/TTS, n-best results



- Backend integration
  - Web Services element
  - Databases
  - HTTP / XML
  - Custom Java classes
- Debugger to simulate calls
- Customisation / reusability

# Call Studio Development Environment

Projects and files explorer

Application call flow

Script element properties and configuration

Debug / Builder switch

Script element palette

Prompt manager

The screenshot displays the Cisco Call Studio Builder interface. The main workspace shows a call flow diagram with elements like 'Call Start', 'Subdialog\_Start\_01', 'Registration\_ID', 'Welcome\_To\_Barcelona', 'Select\_Option', 'Sorry', 'MTE', 'Agenda', 'Travel', 'Come\_Again', and 'Hang Up'. The interface includes several panels: a 'Navigator' on the left for project files, an 'Elements' palette for script elements, a 'Voice Element - Digits' configuration panel on the right, and a 'Prompt Manager' at the bottom. A 'Debug / Builder' switch is located in the top right corner of the application window.

Name	Value
* Input Mode	both
* Noinput Timeout	5s
* Digits Max NoInput ...	3
* Digits Max NoMatch ...	3
* Digits Confidence Le...	0.40
* Min Digits	6
* Max Digits	6
* Disable Hotlinks	false
* Secure Logging	false
* Maxnbest	1

Page	Element Group	Element Name	AG Name	AG Language	A...	Audio Item
page1		Come_Again	Initial	Default	1	audio item

# Getting Started – Build an Application

The screenshot shows the Cisco Unified Call Studio Builder interface. The main workspace displays a call flow diagram with elements: 'Start Of Call', 'CVP Subdialog Start\_01', and 'Welcome To Barcelona'. A context menu is open over the 'Welcome To Barcelona' element, showing options like 'Exit States', 'Skip and >', 'Error Element', 'Group Elements', 'Undo Rename Element', 'Redo', 'Cut', 'Copy', 'Paste', 'Edit Comment', 'Rename', and 'Delete'. The 'Exit States' sub-menu is expanded, showing 'done'. The 'Elements' palette on the left includes 'Audio', 'Call Control', 'Cisco', 'Commerce', 'Context', 'Date & Time', 'Form', 'Integration', 'Math', 'Menu', 'Notification', 'Number Capture', 'Record', 'Local Elements', 'Element Groups', 'Subdialog Elements', 'Subdialog Invoke', 'Cisco', 'CVP Subdialog Start', 'CVP Subdialog Return', and 'General'. The 'Element Configuration' pane on the right shows the configuration for 'Voice Element - Audio', including 'Audio Groups' (Initial, audio item 1), 'Default Language', and 'TTS' (Welcome to Networkers 2008 in sunny Barcelona). A table at the bottom shows the current element configuration:

Page	Element Group	Element Name	AG Name	AG Language	A...	Audio Item N...
page 1		Welcome To Barcelo...	Initial	Default	1	audio item 1

Drag and drop audio element from palette, rename ...

Join up element exit paths

Configure element properties

# Possibly Include Database Lookup

Drag and drop Database element from palette

The screenshot shows the Cisco Unified Call Studio Builder interface. On the left, the 'Elements' palette is visible, with the 'Database' element highlighted. A red arrow points from this element to a 'DB Lookup ID' element in the call flow diagram. Another red arrow points from the 'DB Lookup ID' element to the 'Element Configuration' window. A callout box with a lightning bolt icon points to the 'SQL Query' field in the configuration window, containing the text: 'Bring up query editor window'.

Bring up query editor window

Name	Value
* Type	single
* JNDI Name	nw08db
* SQL Query	select * from attendees where RegID = {Data.Element.Registration Number .valu...

# Use Tag Substitution to Build Query

Setting - SQL Query

```
select * from attendees where RegID = {Data.Element.Registration Number.value}
```

**Substitution Tag Builder**

Create a substitution tag.

Element Data | Session Data | Call Data | Caller Activity | Demographics | Account Info | Phone Number | Date/Time

Element: Registration Number

Element Data: value

Value: select \* from attendees where RegID = {Data.Element.Registration Number.value}

Insert selected object.data

Bring up Substitution Tag Builder to select and insert variables into the query string

OK Cancel



# Play Data From Database

Drag and drop Audio element from palette

The screenshot displays the Cisco Unified Call Studio Builder interface. On the left, the 'Elements' palette is open, showing a tree view of various audio and call control elements. A red arrow points from the 'Audio' element in the palette to a 'Play Attendee Info' element in the call flow diagram. The call flow diagram shows a sequence of steps: 'Start Of Call' leads to 'CVP Subdialog Start\_01', which leads to 'Welcome To Barcelona', then 'Registration Number', 'DB Lookup ID', and finally 'Play Attendee Info'. A 'Sorry' element is also present, connected to 'DB Lookup ID' and 'CVP Subdialog Return\_01'. The 'Element Configuration' window for the 'Audio' element is open on the right, showing the 'Audio' tab. Under 'Audio Groups', the 'Initial' group is expanded, showing a list of audio prompts: 'Hello', 'Registered', 'Session-count', and 'Sessions'. The 'Default Language' section has 'Audio File / TTS' selected. The 'Use Default Audio Path' checkbox is checked. The 'URI' field is empty, and the 'TTS' field contains the text 'Hello {Data.Element.DB Lookup ID.Firstname}'. A callout box points to the TTS field with the text: 'Substitution using data retrieved from database lookup (referenced by DB table column name)'. The call flow diagram also shows transitions labeled 'next', 'done', 'max\_nomatch', and 'max\_noinput'.

Substitution using data retrieved from database lookup (referenced by DB table column name)

# Use the Debugger for Initial Testing

Call input and output

Invoke debugger to simulate call activity

Name	Value
Element Data	
confidence	1.0
value	145622
nbestUtterance1	145622
nbestConfidence1	1.0
nbestInputmode1	voice
nbestInterpretation1	145622

```
07-Jan-2008 01:20:04 org.apache.coyote.http11.Http11BaseProtocol st
INFO: Starting Coyote HTTP/1.1 on http-57485
07-Jan-2008 01:20:04 org.apache.catalina.startup.Catalina start
INFO: Server startup in 5062 ms
```

Name	Value	Create

Connect		Clear	
No Input	No Match	Hang Up	
1	2	3	
4	5	6	

# Deploy to VoiceXML Server

The screenshot displays the Cisco Unified Call Studio Builder interface. The main workspace shows a call flow diagram with the following elements and transitions:

- Start Of Call** (Start) → **next** → **CVP Subdialog Start\_01**
- CVP Subdialog Start\_01** → **done** → **Welcome To Barcelona**
- Welcome To Barcelona** → **done** → **Registration Number**
- Registration Number** has two outgoing paths:
  - max\_nomatch** → **Sorry**
  - max\_noinput** → **DB Lookup ID**
- DB Lookup ID** → **done** → **Play Attendee Info**
- Play Attendee Info** → **done** → **Delegate Type**
- Delegate Type** branches into three paths:
  - customer** → **customer** (Speaker)
  - partner** → **partner** (Speaker)
  - speaker** → **speaker** (Speaker)
- Sorry** → **done** → **CVP Subdialog Return\_01**

The **Deploy Call Studio Project(s)** dialog box is open, showing the following configuration:

- Deploy Call Studio Project(s)**: Deploy one or more Call Studio Projects.
- Projects List**:
  - Cannes Demo
  - NW\_2008
  - callcontrol
  - outbound
- Deploy Destination**:
  - Archive File: \\Nas1\disk 2\CVP\_VXML Apps\outbound.zip (Browse...)
  - Audio Home: Z:\Cisco\CVP\VXMLServer (Browse...)
- Deploy Options**:
  - Deploy Remotely
  - FTP Server: \_\_\_\_\_
  - User ID: \_\_\_\_\_
  - Password: \_\_\_\_\_
- Buttons**: Finish, Cancel

A red arrow points from the **Deploy** menu item in the left-hand pane to the **Audio Home** field in the dialog box.

**Deploy to server or location for OAMP deployment**

# Configure the VoiceXML Gateway to Invoke the Application

```
ip host bristol 10.52.202.38
!
application
!
service networkers_2009 flash:CVPSelfService.tcl
  param CVPPrimaryVXMLServer bristol
  param CVPSelfService-port 7000
  param CVPSelfService-app NW_2009
!
!
dial-peer voice 8801 voip
  service networkers_2009
  incoming called-number 8801
dtmf-relay rtp-nte h245-signal h245-alphanumeric
codec g711ulaw
```

3. CVP VXML Server host / address

2. Service invokes CVPSelfService.tcl and points to CVP VXML Server and application "NW\_2009"

1. Handle incoming call and invoke service "networkers\_2009"

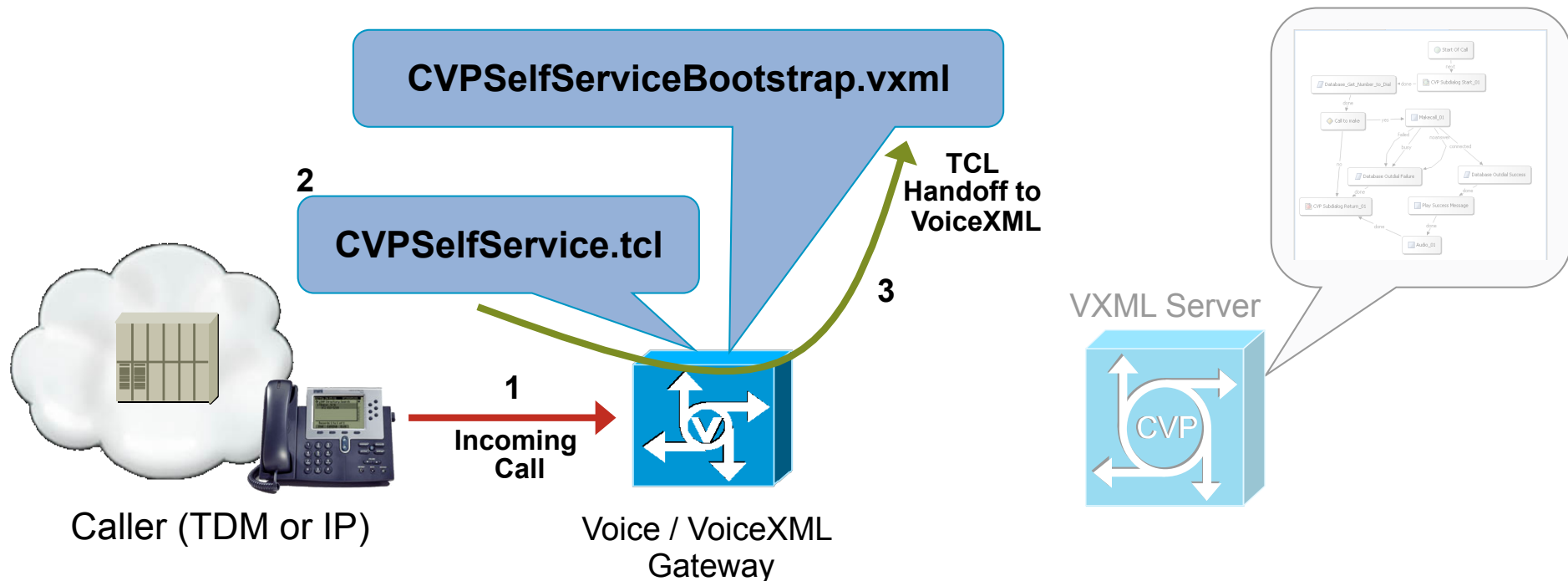
## What happens next?

# Agenda

- Brief Overview of CVP Standalone VoiceXML
- Incoming Calls and Data Flow
- How-Is-It-Done Tips and Tricks
- Optimal Prompt Caching
- Custom Script Elements

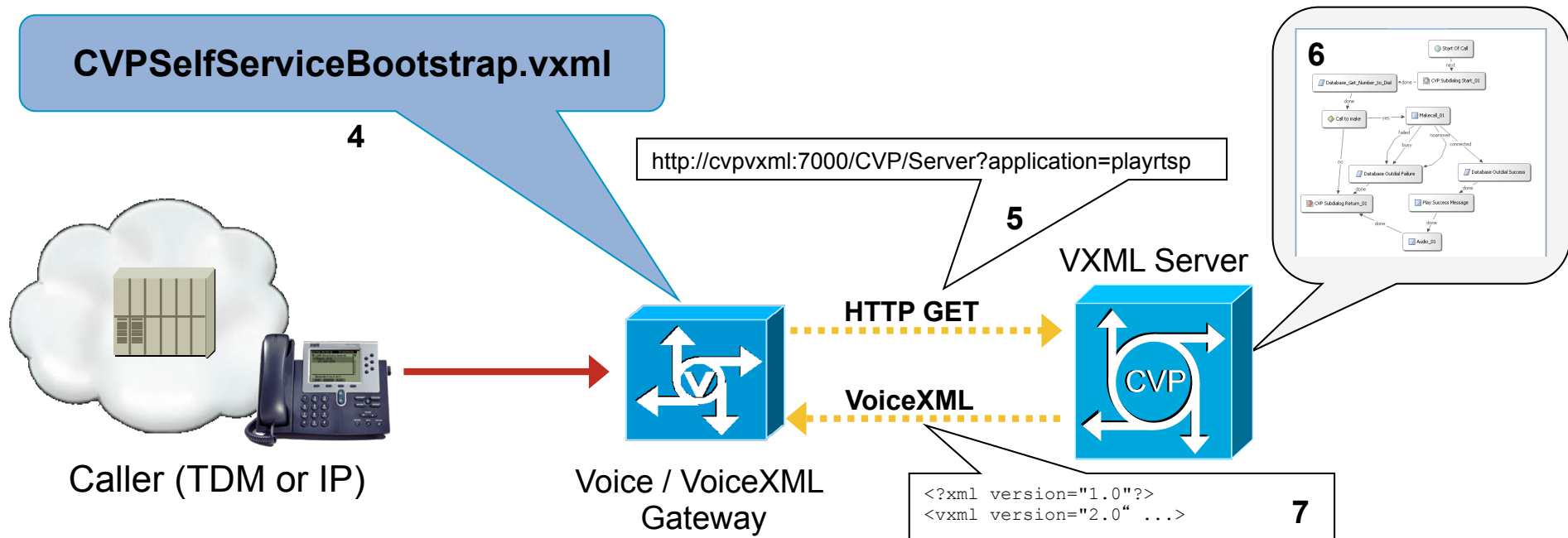
# What Does Happen Next?

1. Incoming call matches dial-peer and triggers the service that invokes CVPSelfService.tcl
2. CVPSelfService.tcl reads the configuration parameters for the service, **immediately answers the call** and ...
3. Performs handoff to CVPSelfServiceBootstrap.vxml passing an argument string with data to build the initial invocation URL



# What Does Happen Next?

4. CVPSelfServiceBootstrap.vxml builds the application invocation URL and executes a subdialog element using this URL plus parameters including required application name, called/calling numbers, callid, user defined variables configured on the service
5. An HTTP GET request is sent to the VoiceXML server
6. The VoiceXML server processes this request and invokes the application defined in the URL parameter "application"
7. The VoiceXML server returns a VoiceXML document



# Agenda

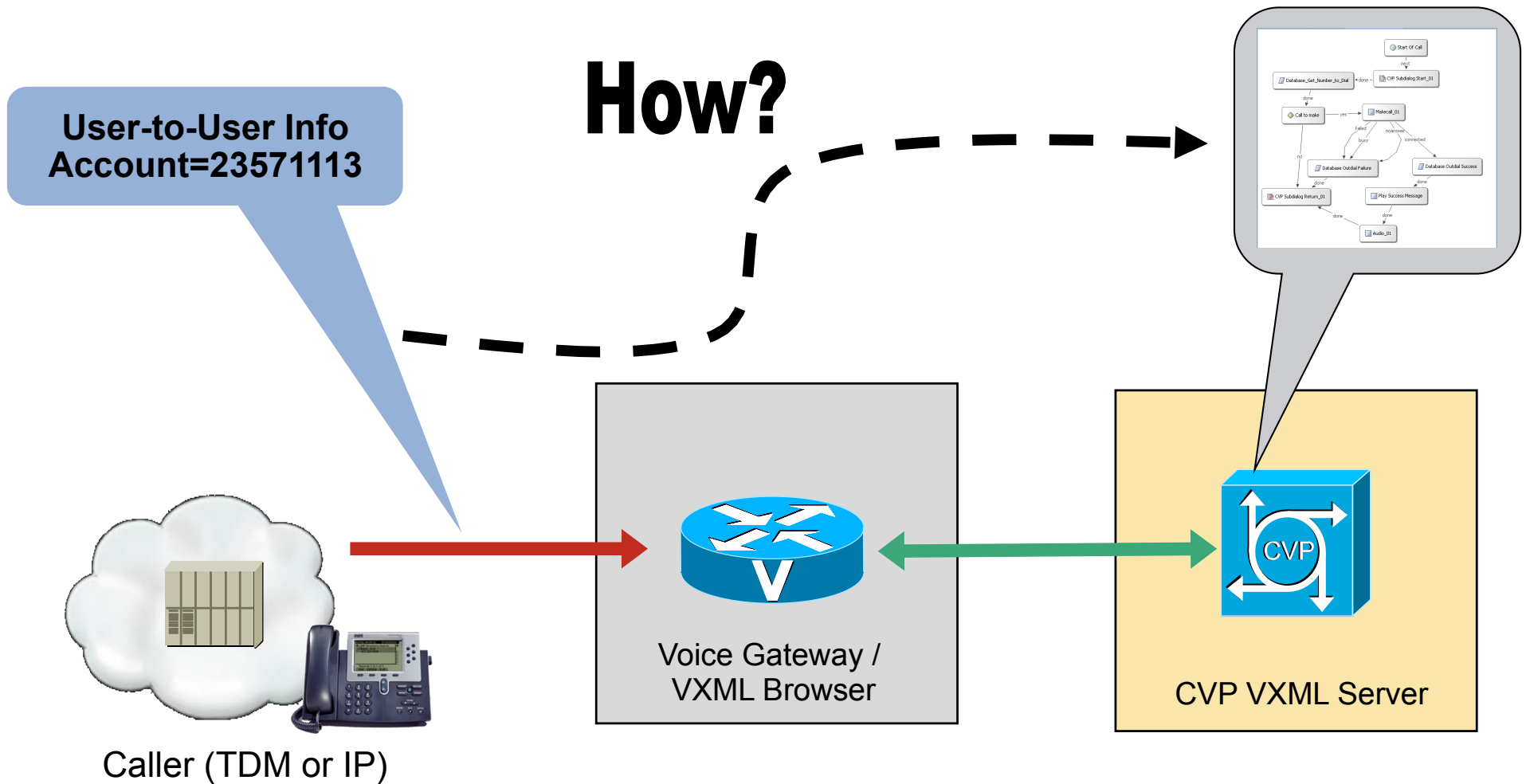
- Brief Overview of CVP Standalone VoiceXML
- Incoming Calls and Data Flow
- **How-Is-It-Done Tips and Tricks**
- Optimal Prompt Caching
- Custom Script Elements



# Call Signaling – Accessing Incoming UUI



# Script Access To Incoming User-User Info



# Accessing Incoming User-User Info

- GTD (Generic Transparency Descriptor) used to transport call signalling information between VoIP endpoints
- More information available than just ANI and DNIS, for example (but network dependent)
  - Calling Party information including presentation/screening indicators
  - User-User information
  - Redirection information
- Both TCL and VoiceXML can read and modify GTD content
- UUI and other incoming signalling information can readily be accessed on the VoiceXML gateway
- **How is it done? ...**

# Referencing GTD content

- Access in TCL or VoiceXML
- VoiceXML uses format *attribute[instance].field*
- For example: UUS[0].dat, CGN[0].pi
- UUS,pd,dat
  - pd – protocol discriminator
  - dat – user-to-user info
- CGN,noa,cni,npi,pi,si,#
  - noa – nature of address
  - cni – complete number indicator
  - npi – numbering plan indicator
  - pi – presentation indicator
  - si – screening indicator
  - # – address
- Useful reference guide ITU-T Recommendation Q.1980.1

# Accessing Incoming User-User Info

- Could extract from GTD in CVPSelfService.tcl and include as parameter on the initial URL
  - Requires modification to standard CVP TCL script
  - Better to use alternative approach driven from application
- Serve a custom VoiceXML document from the CVP VoiceXML application that assigns GTD content to element or session variables
  1. VoiceXML Insert element to deliver custom VoiceXML
  2. Custom element that uses VFC's and specify GTD item via configuration settings

(More flexible although bit more difficult approach than VoiceXML Insert)

# Retrieving UUS from GTD content (VoiceXML Insert Example)

```
<vxml version="1.0" application="/CVP/Server?
audium_vxml_root=true&calling_into=getgtduus&namelist=element_log_uusdat">

  <form id="audium_start_form">
    <block>
      <assign name="audium_vxmlLog" expr="" />
      <assign name="audium_element_start_time_millisecs" expr="new Date().getTime()" />
      <goto next="#start" />
    </block>
  </form>

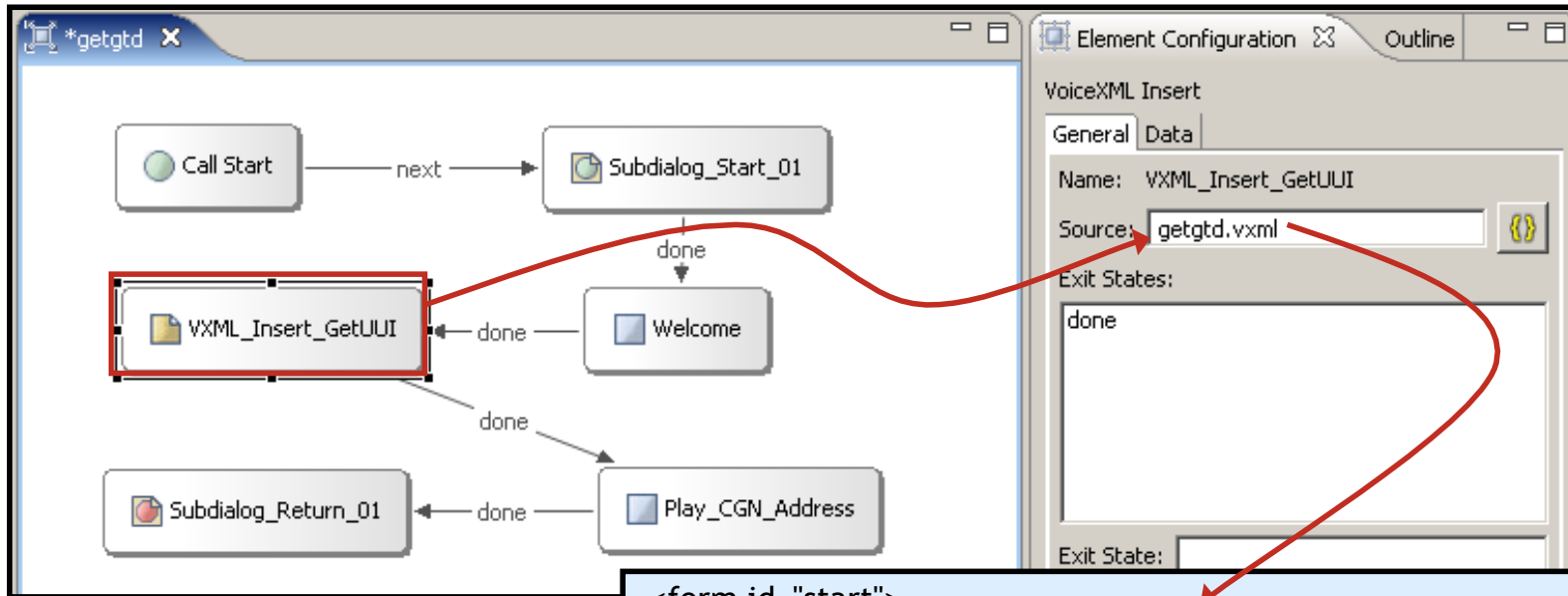
  <form id="start">
    <var name="setup_gtd" expr="com.cisco.signal.gtdlist['setup_indication']"/>
    <var name="element_log_uusdat" expr="setup_gtd.UUS[0].dat"/>
    <block>
      <assign name="audium_exit_state" expr="done" />
      <return namelist="audium_exit_state element_log_uusdat audium_vxmlLog audium_hotlink audium_hotevent
audium_error audium_action" />
    </block>
  </form>
</vxml>
```

Assign required GTD field to variable

Return to VoiceXML server and  
populate element data

- VoiceXML Insert element references subdialog document above
- VoiceXML browser assigns variable(s) with the required GTD content and CVP element variables are populated when subdialog returns

# GetGTD Example – Calling Party Number



```
<form id="start">
  <var name="setup_gtd" expr="com.cisco.signal.gtdlist['setup_indication']"/>
  <var name="element_log_cgntp" expr="setup_gtd.CGN[0].pi"/>
  <var name="element_log_cgnsi" expr="setup_gtd.CGN[0].si"/>
  <var name="element_log_cgnavdr" expr="setup_gtd.CGN[0]['#']"/>

```

```
*Jan 13 22:41:33.922: ISDN Se0/0/0:15
Q931: RX <- SETUP pd = 8 callref = 0x00f
  Bearer Capability i = 0x8090A3
    Standard = CCITT
    Transfer Capability = Speech
    Transfer Mode = Circuit
    Transfer Rate = 64 kbit/s
  Channel ID i = 0xA98382
    Exclusive, Channel 2
  Calling Party Number i = 0x0081, '1000'
    Plan:Unknown, Type:Unknown
  Called Party Number i = 0x80, '7784423'
    Plan:Unknown, Type:Unknown
  User-User i = 0x00, 'Networkers'
```

```
Jan 13 22:41:33.926: gtd msg = IAM,
PRN,isdn*,,QSIG*,
USI,rate,c,s,c,1
USI,lay1,alaw
TMR,00
CPN,00,,u,7784423
CGN,00,,u,y,2,1000
UUS,0,4e6574776f726b657273
CPC,09
FCI,,,,,,y,
GCI,29d51fd7e0fa11dd807400169db58e40"
```

```
2:33.984,VXML_Insert_GetUII,data,cgntp,y
2:33.984,VXML_Insert_GetUII,data,cgnsi,2
2:33.984,VXML_Insert_GetUII,data,cgnavdr,1000
```

```
192.168.1.22.1231882953750.42.getgtd,01/13/2009 13:42:33.984,VXML_Insert_GetUII.exit.done
```

# Call Signaling – Sending UUI





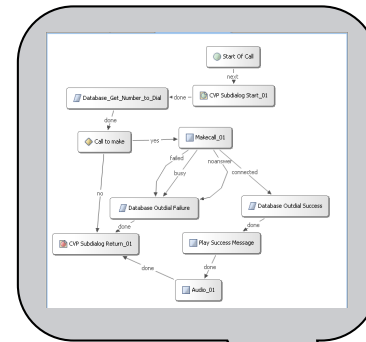
# Sending User-User Info on Transfers

Transfer Destination  
(TDM or IP)



User-to-User Info  
Cust\_ID=003762103

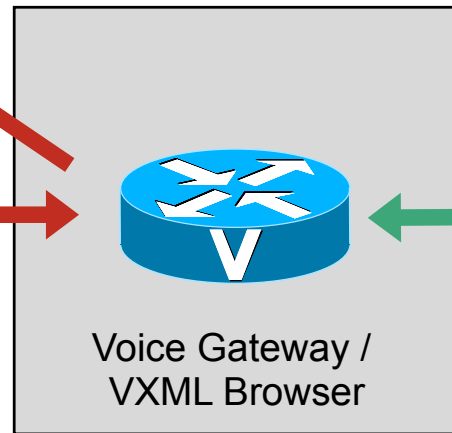
How?



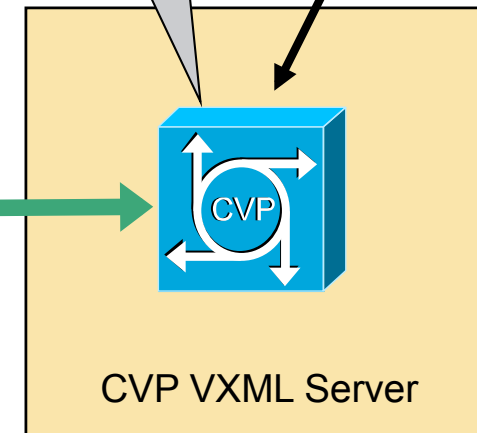
DB



Caller (TDM or IP)



Voice Gateway /  
VXML Browser



CVP VXML Server

# Sending User-User Info on Transfers

- Cisco VoiceXML Transfer Element supports additional proprietary attributes including “cisco-gtd”
- Allows application-created GTD containing UUI and other signalling data to be sent on the transferred call leg
- Not exposed as standard in CVP Call Studio transfer script element
- However, it can be included via straightforward customisation
- How is it done? ...

# Sending User-User Info

- Serve a custom VoiceXML document from the CVP VoiceXML application
- Builds GTD containing User-to-User attribute UUS[0].dat
- Performs the transfer including Cisco proprietary attributes
- Use either –
  1. VoiceXML Insert element to deliver custom VoiceXML as a subdialog
  2. Custom transfer element
    - Use VFC's to build the transfer VoiceXML
    - Use `addProprietaryAttribute("cisco-gtd", ...)` method to include GTD on the transfer element
    - Specify this and other Cisco attributes via element configuration settings
- 2 is more flexible although bit more difficult approach than VoiceXML Insert

# Extending Transfer Element Attributes (VoiceXML Insert Example)

```
<form id="start">
```

```
<var name="new_gtd" expr="new com.cisco.objclass.gtd()"/>  
<var name="new_gtd.message_type" expr="'IAM'"/>  
<var name="new_gtd.UUS[0].pd" expr="0"/>  
<var name="new_gtd.UUS[0].dat" expr="audium_session_xferuus"/>
```

**Create GTD and add UUS attribute with dat field set to contents of CVP session variable xferuus**

```
<transfer name="xfer" bridge="true" connecttimeout="10s" destexpr="audium_session_xferdest" cisco-gtd="new_gtd">  
  <filled>  
    <if cond=" ( xfer == 'far_end_disconnect' ) ">  
      <assign name="audium_exit_state" expr="done" />  
    <elseif cond=" ( xfer == 'busy' ) " />  
      <assign name="audium_exit_state" expr="busy" />  
    <elseif cond=" ( xfer == 'noanswer' ) " />  
      <assign name="audium_exit_state" expr="noanswer" />  
    <elseif cond=" ... " />  
      <assign name=" ... " />  
    <else/>  
      <assign name="audium_exit_state" expr="phone_error" />  
    </if>  
  
    <return namelist="audium_exit_state audium_vxmlLog audium_hotlink audium_hotevent audium_error  
    audium_action" />  
  </filled>  
</transfer>  
</form>
```

**Use session variable xferdest as transfer destination**

**Include cisco-gtd attribute on transfer element**

- VoiceXML Insert element references subdialog document containing GTD assignment and transfer element with cisco-gtd attribute

# Call Signaling – Display Name and Calling Party



# Sending Display Name on Transfers

- Can use the same approach as sending UUI to send Display-Name on transfers
- Build GTD containing Information-for-Display DIS[0].info
- Transfer using Cisco proprietary attribute “cisco-gtd”
- Can use to pass context from IVR application that will be displayed on CUCM IP Phone
- So, several ways to forward information when call is transferred: UUI, Display-Name and also CLI override

# Overriding ANI on transfers

- Could use GTD but there is a simpler method
- Can just add Cisco proprietary attribute “cisco-ani” or “cisco-aniexpr” to transfer element
- Set it to the required digit string, tel:0123456789
- Useful last resort if destination system can't use UUI or Display-Name and only can receive called and calling party numbers

# Including Display-Name and Modified ANI (VoiceXML Insert Example)

```
<form id="start">
  <var name="new_gtd" expr="new com.cisco.objclass.gtd()"/>
  <var name="new_gtd.message_type" expr="IAM"/>
  <var name="new_gtd.UUS[0].pd" expr="0"/>
  <var name="new_gtd.UUS[0].dat" expr="audium_session_xferuus"/>
  <var name="new_gtd.DIS[0].info" expr="audium_session_xferdisp"/>

  <transfer name="xfer" bridge="true" connecttimeout="10s"
    destexpr="audium_session_xferdest" cisco-gtd="new_gtd" cisco-aniexpr="audium_session_xferani" >

    <filled>
      <if cond=" ( xfer == 'far_end_disconnect' ) ">
        <assign name="audium_exit_state" expr="done" />
      <elseif cond=" ( xfer == 'busy' ) " />
        <assign name="audium_exit_state" expr="busy" />
      <elseif cond=" ( xfer == 'noanswer' ) " />
        <assign name="audium_exit_state" expr="noanswer" />
      <elseif cond=" ... " />
        <assign name=" ... " />
      <else/>
        <assign name="audium_exit_state" expr="phone_error" />
      </if>

      <return namelist="audium_exit_state audium_vxmlLog audium_hotlink audium_hotevent audium_error
        audium_action" />
    </filled>
  </transfer>
</form>
```

Create GTD and add DIS attribute with info field set to contents of CVP session variable xferdisp

Include cisco-aniexpr attribute set to session variable xferani



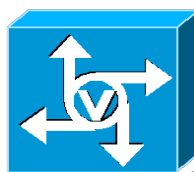
# CVP VoiceXML Passing Context (Using Custom Transfer Element)

Caller (TDM or IP)



**1 Inbound call**

Voice/VXML Gateway



Element Configuration

Voice Element - TransferCisco

General Settings Audio Data Local Hotlinks

Name	Value
* Transfer Destination	5500
Calling-party	874209
Connect Timeout	45
Max Transfer Time	0
* Bridged	true
Transfer Audio	
Display name	Mrs. Smith :-(-
User-to-User	Account=0034701

**2 CVP VXML app runs, sets context and transfers call**

**3 Call transferred**



VXML Server



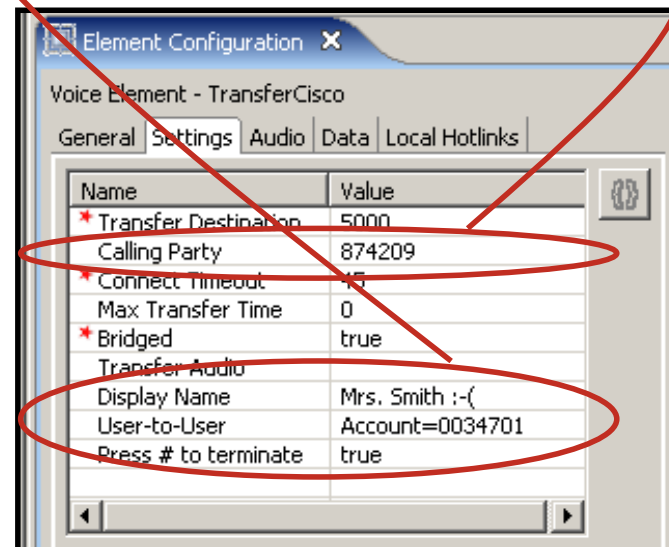
DB

# VoiceXML Form Generated by Custom Transfer Element

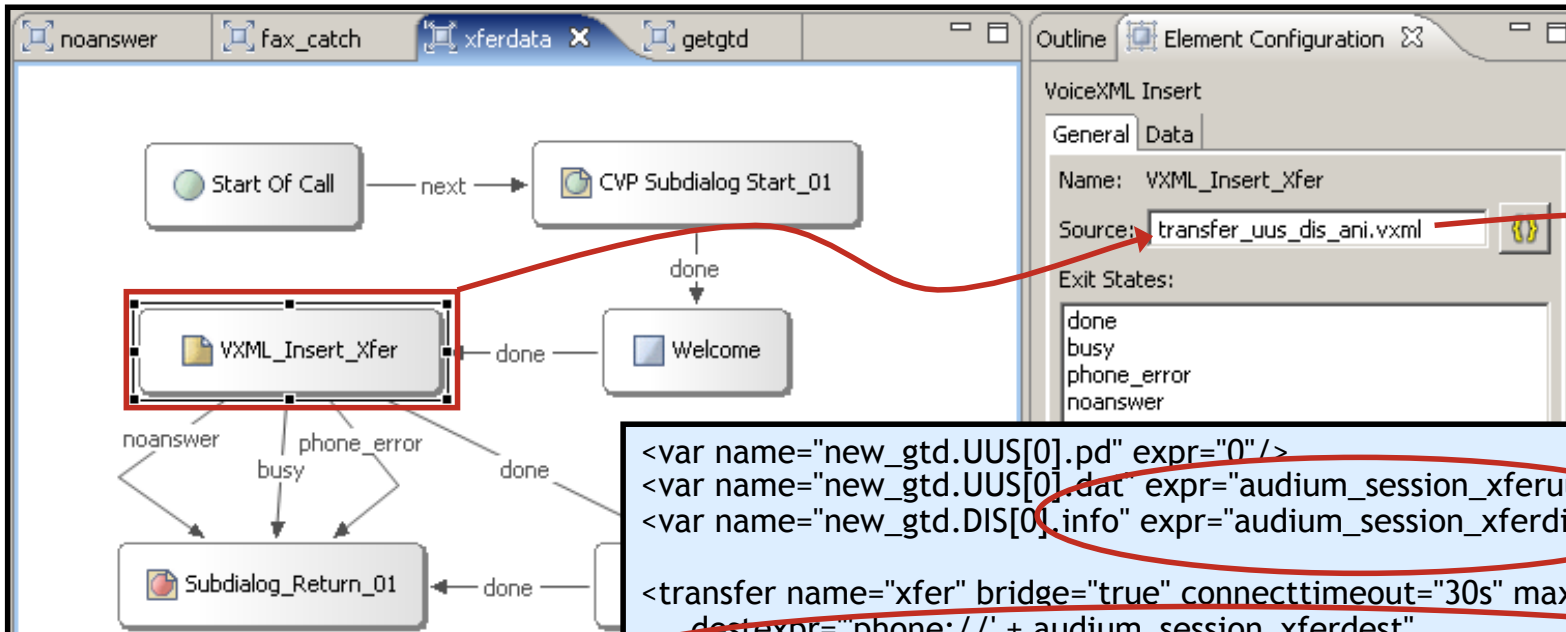
```
<form id="start">
  <var name="gtd" expr="new com.cisco.objclass.gtd()" />
  <var name="gtd.message_type" expr="IAM" />
  <var name="gtd.UUS[0].pd" expr="0" />
  <var name="gtd.UUS[0].dat" expr="Account=0034701" />
  <var name="gtd.DIS[0].info" expr="Mrs. Smith :-(" />
  <transfer name="xfer" bridge="true" connecttimeout="45s" maxtime="0s" dest="phone:/5000" cisco-ani="tel:874209"
            cisco-gtd="gtd" cisco-longpound="true" />

  <filled mode="all">
    <submit next="/CVP/Server" method="post" namelist="audium_vxmlLog xfer" />
  </filled>
</form>
```

- Shows GTD creation and attribute setting
- Shows Cisco proprietary attributes
- Also includes cisco-longpound which allows the caller to terminate the transfer using #
- This example used configurable settings as part of the custom element



# Transfer With Data Example



```

<var name="new_gtd.UUS[0].pd" expr="0"/>
<var name="new_gtd.UUS[0].dat" expr="audium_session_xferuus"/>
<var name="new_gtd.DIS[0].info" expr="audium_session_xferdis"/>

<transfer name="xfer" bridge="true" connecttimeout="30s" maxtime="15s"
  destexpr="// + audium_session_xferdest"
  cisco-aniexpr="tel:' + audium_session_xferani" cisco-gtd="new_gtd">

```

Sent:  
 INVITE sip:6681000@192.168.1.78:5060 SIP/2.0  
 Via: SIP/2.0/UDP 192.168.1.78:5060;x-route-tag="tgrp:2811-a-000";x-ds0num="ISDN 0/0/0:15 ";branch=z9hG4bKC2386  
 Remote-Party-ID: "Cisco CVP Transfer Test" <sip:0123456789@192.168.1.78>;party=calling;screen=yes;privacy=off  
 From: "Cisco CVP Transfer Test" <sip:0123456789@192.168.1.78>;tag=66DB74-1EA3  
 To: <sip:6681000@192.168.1.78>  
 Diversion: <sip:7784423@192.168.1.78>;privacy=off;reason=follow-me;screen=no

--uniqueBoundary  
 Content-Type: application/gtd  
 IAM  
 UUS,0,48656C6C6F2066726F6D20426172636556C6F6E61  
 DIS,Cisco CVP Transfer Test

```

*Jan 14 14:06:37.955: ISDN Se0/0/0:15
Q931: RX <- SETUP pd = 8 callref = 0x0089
  Calling Party Number i = 0x0081, '1000
  Plan:Unknown, Type:Unknown
  Called Party Number i = 0x80, '7784423'
  Plan:Unknown, Type:Unknown

```

```

7:54.140, VXML_Insert_xfer,enter
7:54.656, VXML_Insert_xfer,data,xferdest,6681000
7:54.656, VXML_Insert_xfer,data,xferani,0123456789
7:54.656, VXML_Insert_xfer,data,xferdis,Cisco CVP Transfer Test
7:54.656, VXML_Insert_xfer,data,xferuus,Hello from Barcelona
7:54.656, VXML_Insert_xfer,exit,done

```

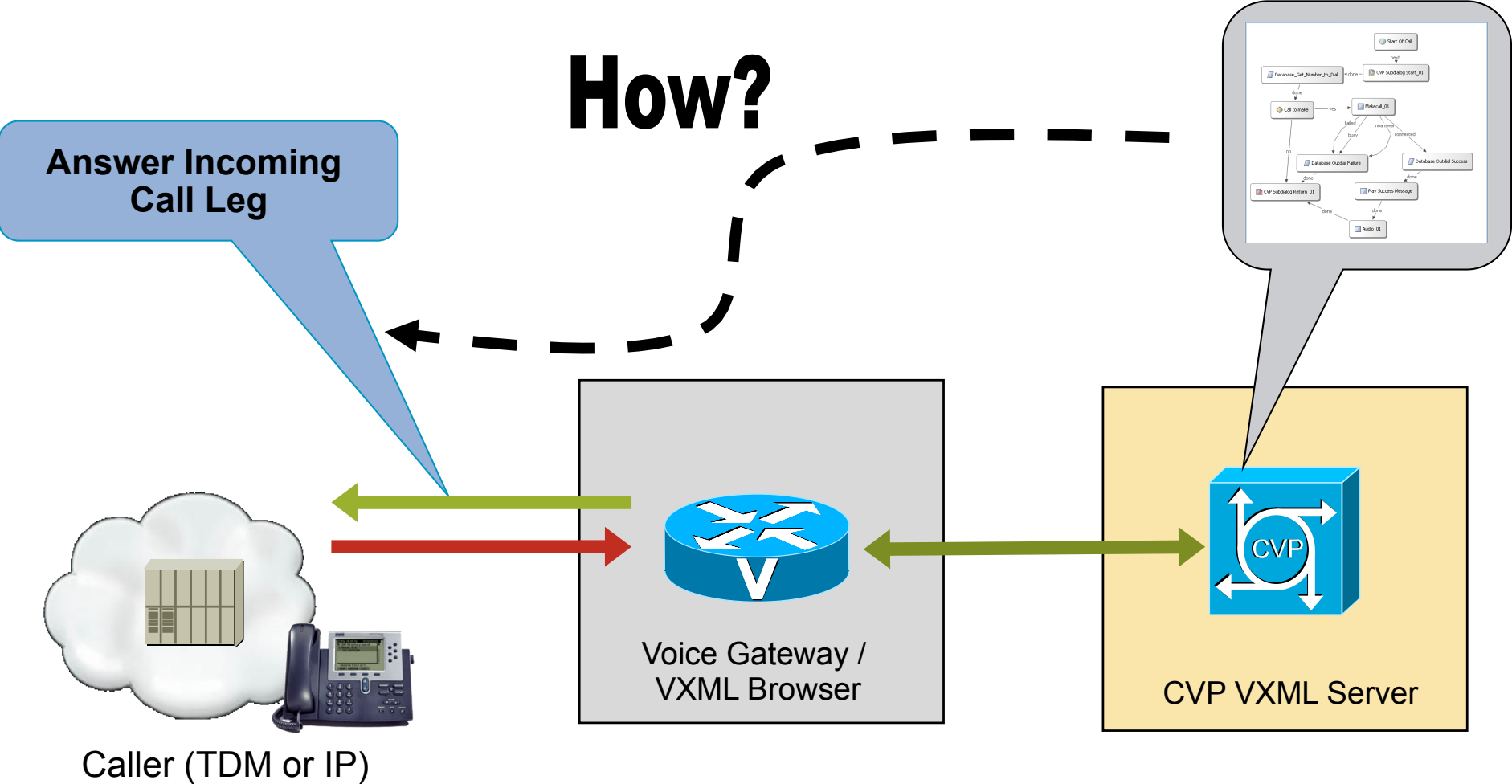
# Call Control From The VoiceXML Application



# Answering Call From Script

- Normal CVP operation is to answer the call immediately at the voice gateway
- But, what if the requirement is to perform an immediate divert attempt before any voice treatment
- Problem – don't want the call to be answered at the ingress gateway as call charging will commence before transfer is attempted
- Need to do two things
  1. Suppress immediate connect at the ingress gateway
  2. Find a way to answer the call from the script explicitly when it's required
- How is it done? ...

# Script Explicitly Answers Incoming Call



# Preventing Immediate Answer

- CVPSelfSevice.tcl script can be modified to not answer the call
- This does mean a minor change to a released/supported CVP TCL script but unfortunately no alternative approach
- Comment-out signalling so that only leg proceeding is used
- **Be Aware: Customised TCL script will not be TAC-supported**

```
#-----  
# Procedure act_Start  
#  
# This procedure is executed when it receives an ev_setup indication event.  
# A setup acknowledgement, call proceeding, and connect message are sent to  
# the incoming call leg and finally the call is handed off the the CVPSelfService app.  
#-----  
proc act_Start { } {  
  
    ...  
#   leg setupack leg_incoming  
#   leg proceeding leg_incoming  
#   leg alert leg_incoming  
#   leg connect leg_incoming  
    ...  
}
```

# Explicit Answer Under Script Control

- Sometimes it's just not possible to do everything we need from VoiceXML
- VoiceXML has no direct way to answer the incoming call
- But using VoiceXML <object> element, it can handoff control temporarily to TCL to perform the leg connect
- Useful alternative approach to make use of gateway functionality that isn't exposed in VoiceXML, especially for call control related requirements



# Invoking TCL from CVP VoiceXML (VoiceXML Insert Example)

```
<form id="start">
```

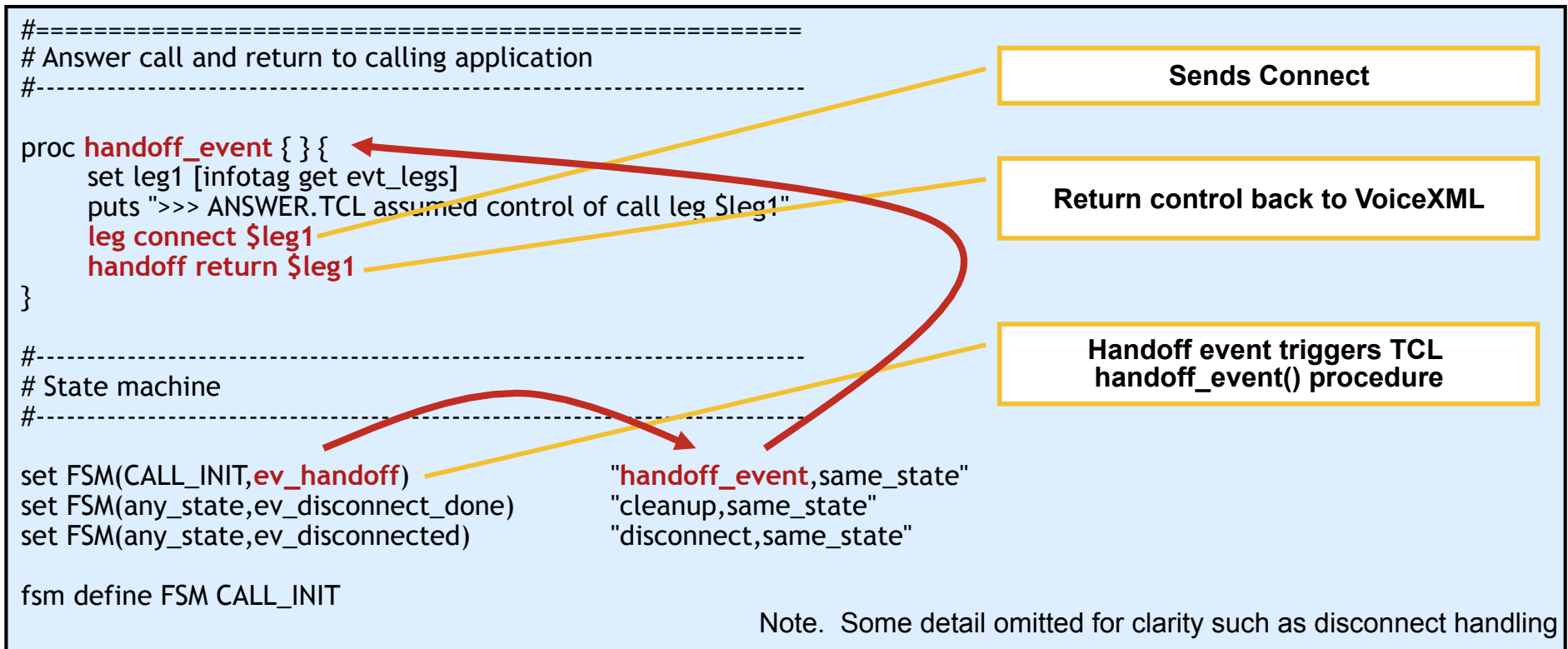
```
<object name="connect" classid="builtin://com.cisco.callhandoff">  
  <param name="return" expr="true"/>  
  <param name="app-uri" expr="builtin://answer"/>  
</object>
```

Use to invoke TCL script and expect to return back here. The service name on the gateway that maps to the the required TCL script is called "answer"

```
<block>  
  <log>***TCL_LEG_CONNECT, returned from TCL</log>  
  <assign name="audium_exit_state" expr="done" />  
  <return namelist="audium_exit_state audium_vxmlLog audium_hotlink audium_hotevent audium_error  
    audium_action" />  
</block>  
</form>
```

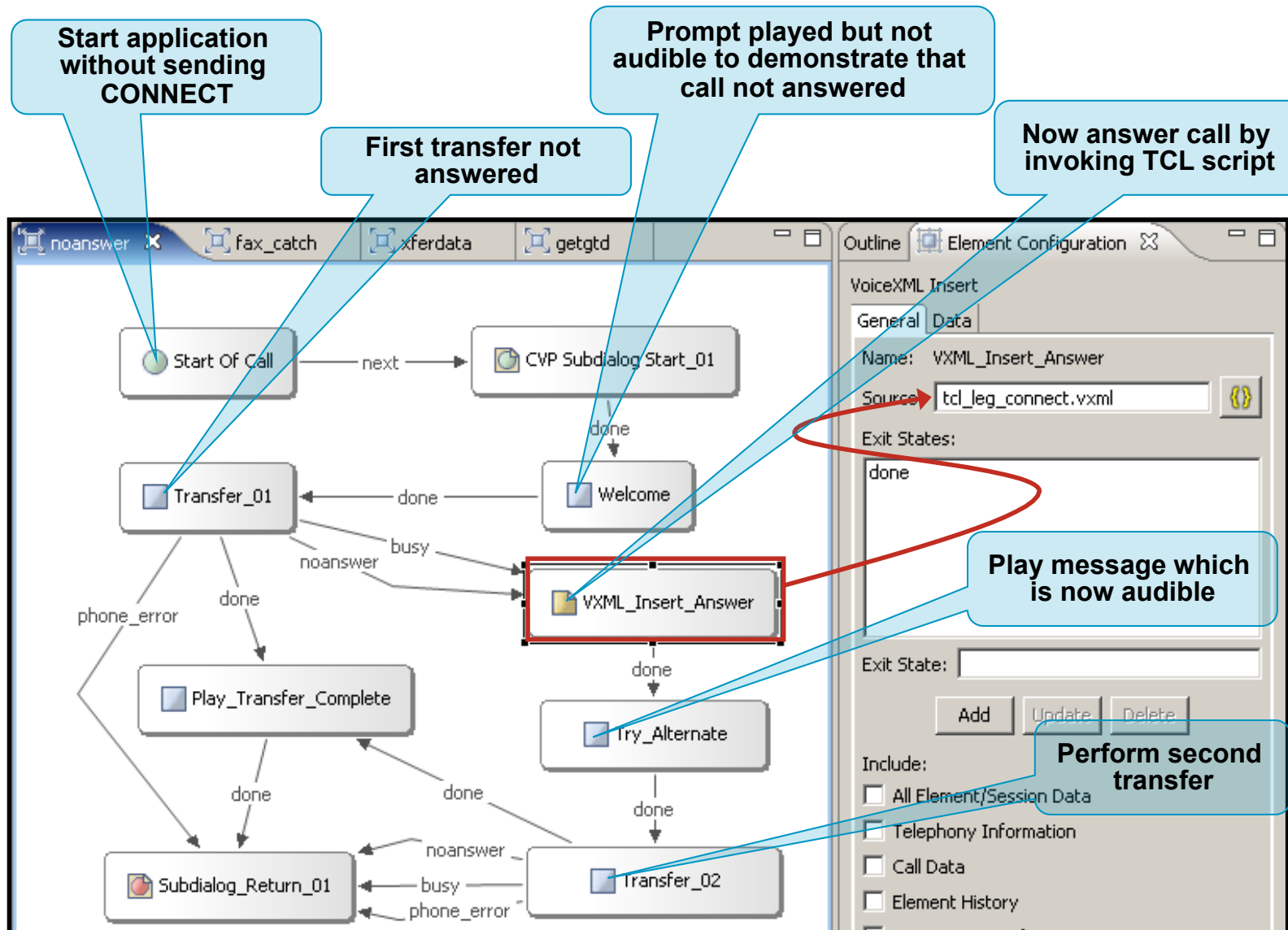
- VoiceXML Insert element references a subdialog document containing the form above
- VoiceXML browser performs handoff to TCL via the <object> element
- TCL script is mapped as a service on the VoiceXML gateway  
application  
service **answer** flash:answer.tcl
- TCL script connects the call and returns control to VoiceXML

# Example TCL Procedure to Answer Call



- Handoff event handler is invoked via the state machine
- It retrieves the ID of the call leg that it now has control of
- Performs a connect on the call leg
- Returns control back to VoiceXML

# Explicit Answer Example



# Explicit Answer Trace

## Incoming call not answered immediately, only CALL PROCEEDING sent back

```
*Jan 14 16:27:15.802: ISDN Se0/0/0:15 Q931: RX <- SETUP pd = 8 callref = 0x00A0
  Calling Party Number i = 0x0081, '1000'
  Called Party Number i = 0x80, '7784423'
*Jan 14 16:27:15.818: ISDN Se0/0/0:15 Q931: TX -> CALL_PROC pd = 8 callref = 0x80A0
```

## CVP application started and first prompt played (unheard)

```
*Jan 14 16:27:15.894: **CVP**0A293708-E18F11DD-80220016-9DB58E40**noanswer**Starting Application...
<audio src="/CVP/audio/welcome.wav" />
```

## First transfer attempt which hits no-answer timeout

```
<transfer name="Mtransfer" bridge="true" connecttimeout="10s" maxtime="0s" dest="phone://97781002">
```

```
*Jan 14 16:27:21.550: ISDN Se0/0/1:15 Q931: TX -> SETUP pd = 8 callref = 0x00A1
  Called Party Number i = 0x80, '7781002'
*Jan 14 16:27:21.578: ISDN Se0/0/1:15 Q931: RX <- ALERTING pd = 8 callref = 0x80A1
*Jan 14 16:27:31.590: ISDN Se0/0/1:15 Q931: TX -> DISCONNECT pd = 8 callref = 0x00A1
*Jan 14 16:27:31.614: ISDN Se0/0/1:15 Q931: RX <- RELEASE pd = 8 callref = 0x80A1
POST /CVP/Server HTTP/1.1
audium_vxmlLog=&Mtransfer=noanswer
*Jan 14 16:27:31.622: ISDN Se0/0/1:15 Q931: TX -> RELEASE_COMP pd = 8 callref = 0x00A1
```

# Explicit Answer Trace

## Answer the incoming call by handing-off to TCL via VoiceXML subdialog

```
<subdialog name="sub" src="tcl_leg_connect.vxml">
```

```
*Jan 14 16:27:31.774: ***TCL_LEG_CONNECT, handing off to ANSWER TCL  
*Jan 14 16:27:31.782: //175//TCL :/tcl_PutsObjCmd: >>> ANSWER.TCL assumed control of call leg 175  
*Jan 14 16:27:31.786: ***TCL_LEG_CONNECT, returned from TCL  
*Jan 14 16:27:31.794: ISDN Se0/0/0:15 Q931: TX -> CONNECT pd = 8 callref = 0x80A0  
*Jan 14 16:27:31.822: ISDN Se0/0/0:15 Q931: RX <- CONNECT_ACK pd = 8 callref = 0x00A0
```

## Play audio prompt and perform second transfer which is answered

```
<audio src="/CVP/audio/alternate.wav" />
```

```
<transfer name="Mtransfer" bridge="true" connecttimeout="10s" maxtime="0s" dest="phone://97781000">
```

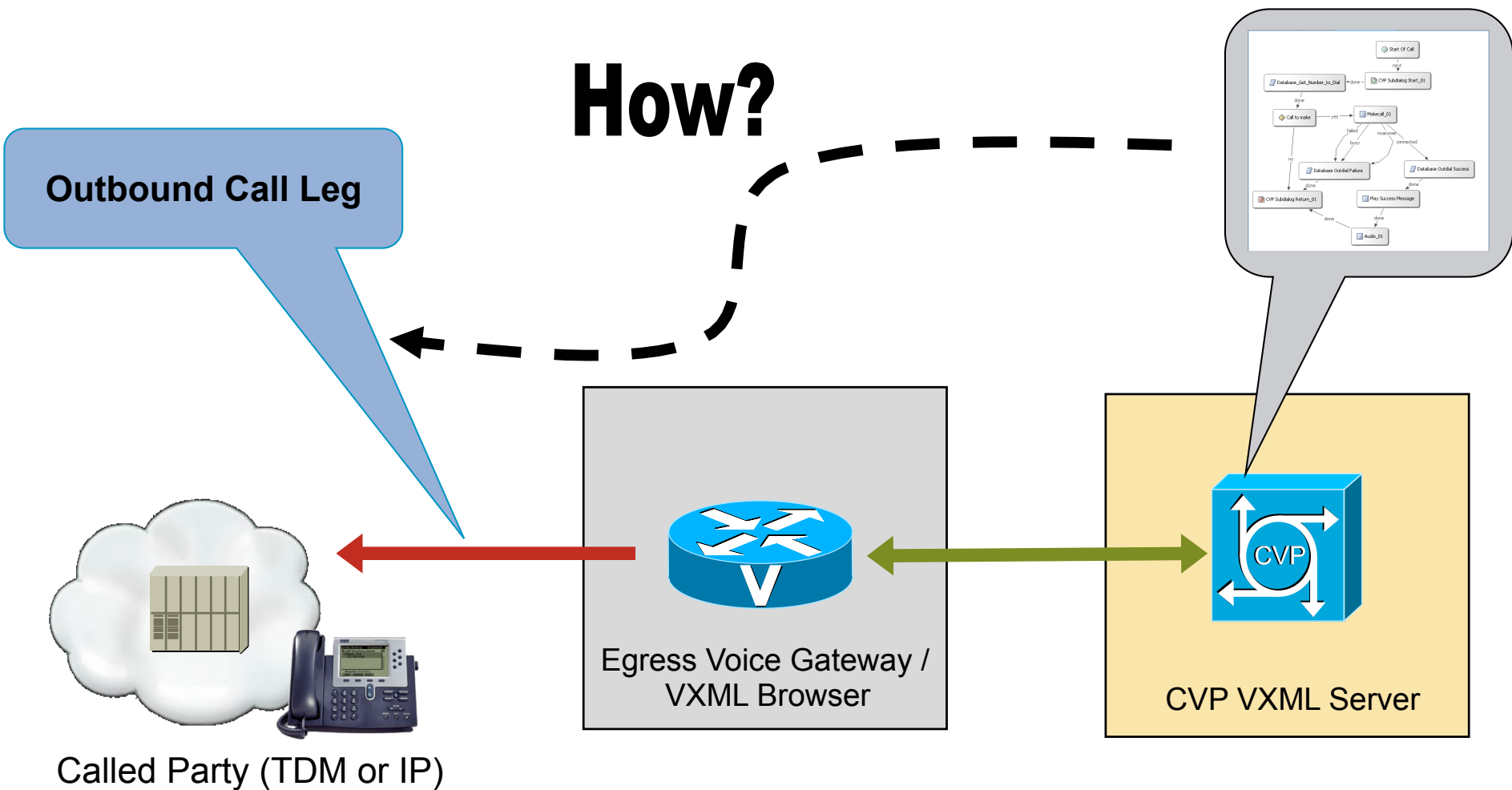
```
*Jan 14 16:27:33.714: ISDN Se0/0/1:15 Q931: TX -> SETUP pd = 8 callref = 0x00A2  
    Called Party Number i = 0x80, '7781000'  
*Jan 14 16:27:33.746: ISDN Se0/0/1:15 Q931: RX <- CALL_PROC pd = 8 callref = 0x80A2  
*Jan 14 16:27:33.746: ISDN Se0/0/1:15 Q931: RX <- ALERTING pd = 8 callref = 0x80A2  
*Jan 14 16:27:33.754: ISDN Se0/0/1:15 Q931: RX <- CONNECT pd = 8 callref = 0x80A2  
*Jan 14 16:27:33.762: ISDN Se0/0/1:15 Q931: TX -> CONNECT_ACK pd = 8 callref = 0x00A2
```

# Making Outbound Calls



# Making an Outbound Call

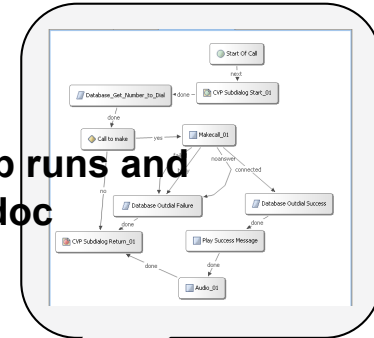
## How?



# CVP VoiceXML Standalone Inbound Calling

**Challenging!**  
**How do you make this deployment model set up an outbound call?**

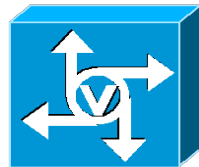
**2 CVP VXML app runs and serves VXML doc**



Caller (TDM or IP)

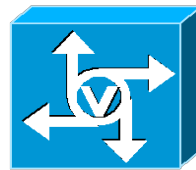


**1 Inbound call**

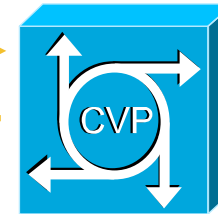


Voice GW

VXML Gateway



VXML Server

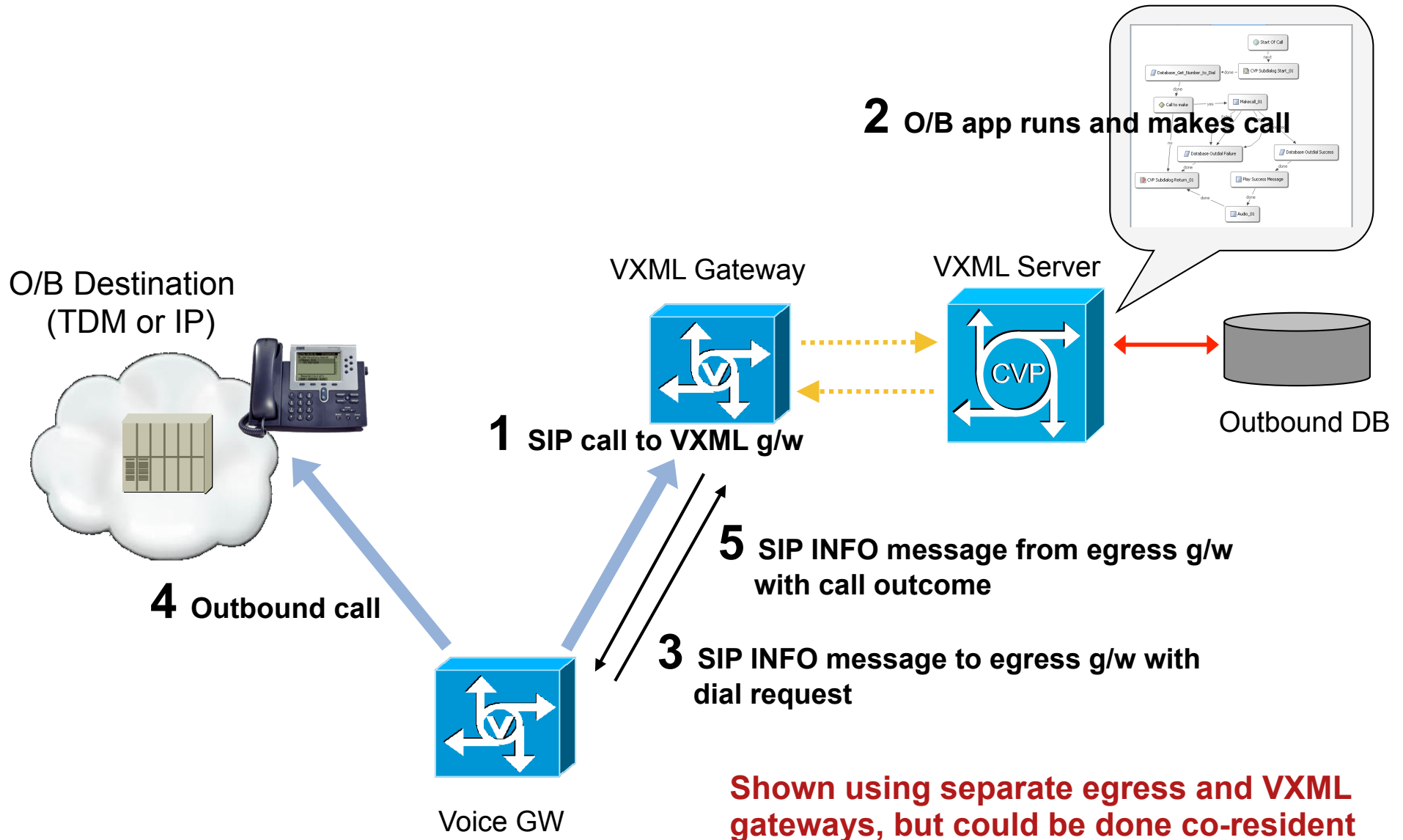


DB

**Shown using separate egress and VXML gateways, but could be done co-resident**



# CVP VoiceXML Standalone Outbound Calling



**Shown using separate egress and VXML gateways, but could be done co-resident**

# CVP VoiceXML Standalone Outbound

- What does it do?
  - Sets up an outbound call and runs an application built with Call Studio
  - Requires only Voice / VXML Gateway and VXML Server (the same components as for inbound)
  - Exactly the same architecture as for inbound
  - Run the outbound script components on selected egress gateways as required
  - Allows us to build applications to perform automated reminder and notification calls such as order readiness, imminent collection/pick-up, status alert

# CVP VoiceXML Outbound Examples

- **Reminder / Alert calls**

CVP outbound application queries back-end system via JDBC, HTTP, WebServices to retrieve pending items for reminder

- Medical appointments
- Car service
- Pharmacy prescription ready for collection
- Incident notification

- **Imminent pick-up notification**

- Driver approaching pick-up location
- Calls automated application, enters pick-up ID and selects option to send a notification call to customer
- Outbound application retrieves pending request and makes call

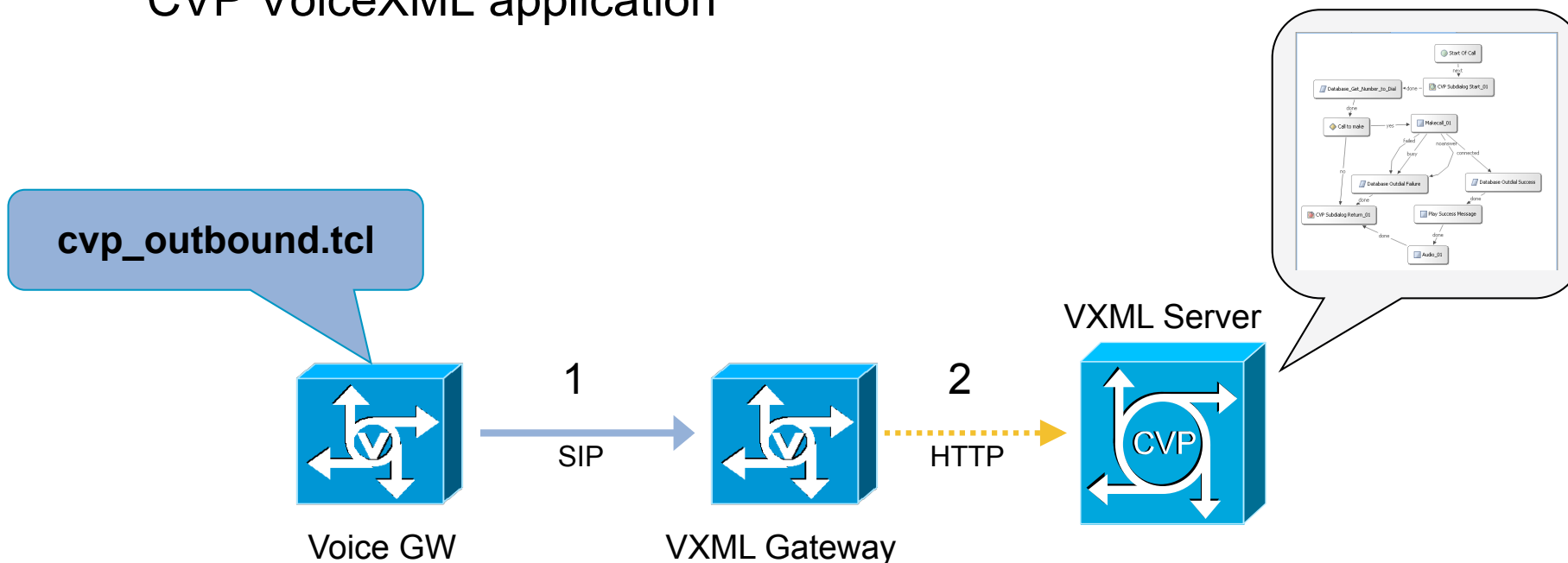
# CVP VoiceXML Standalone Outbound

- Posted on Cisco Developer Community with sample basic application to demonstrate the mechanism in action <http://developer.cisco.com/web/cdc/home>
- Can be downloaded and support is via Developer Services contract or the Developer Community

# CVP VoiceXML Standalone Outbound

## How Does It Work?

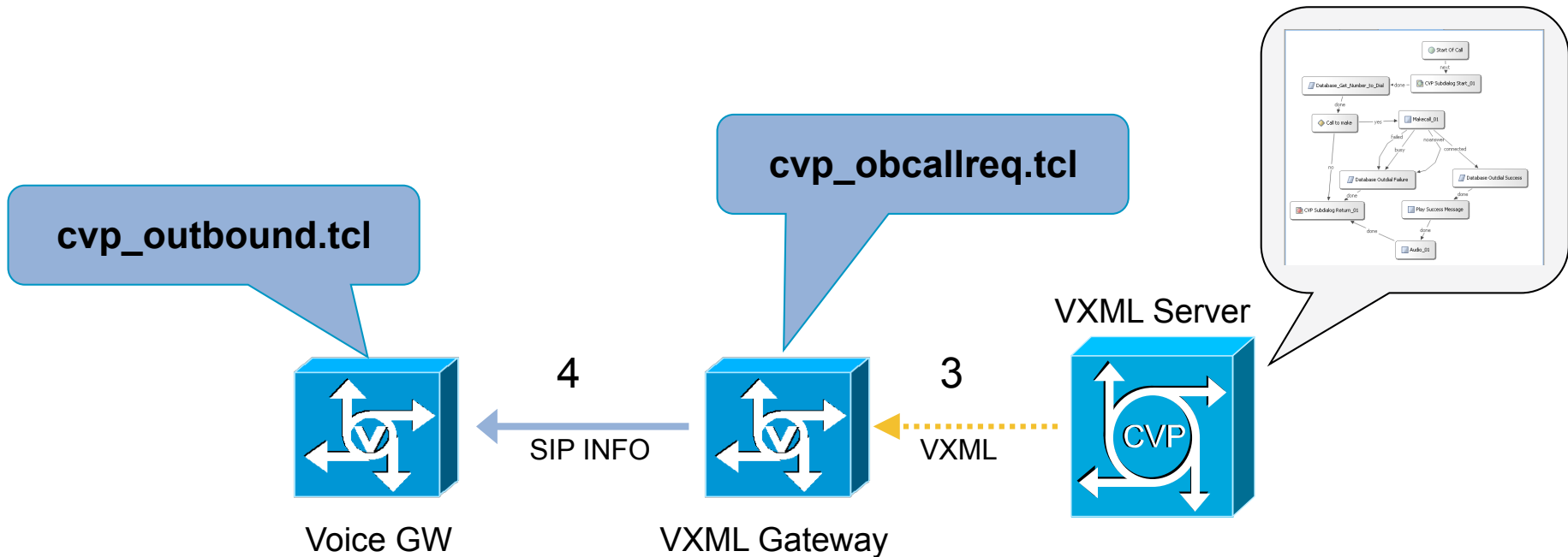
1. TCL script (cvp\_outbound) which runs continuously sets up a SIP call to the VoiceXML gateway
2. VoiceXML gateway, just as for any other incoming VoIP call, makes an HTTP request to the VoiceXML server to invoke a CVP VoiceXML application



# CVP VoiceXML Standalone Outbound

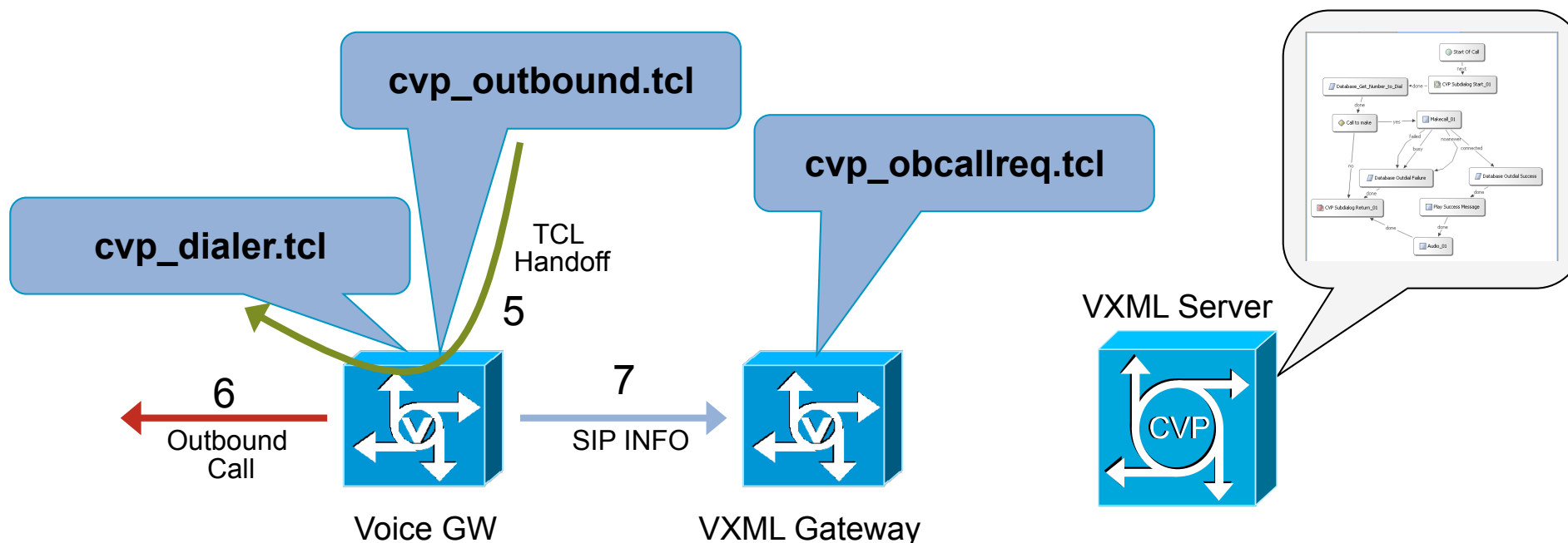
## How Does It Work?

3. CVP VoiceXML application uses the Makecall element to perform temporary handoff to TCL (cvp\_obcallreq) with the argument string containing outbound call request parameters, destination number, etc
4. This TCL script forwards the call request via a SIP INFO message to the cvp\_outbound session on the voice gateway



# CVP VoiceXML Standalone Outbound How Does It Work?

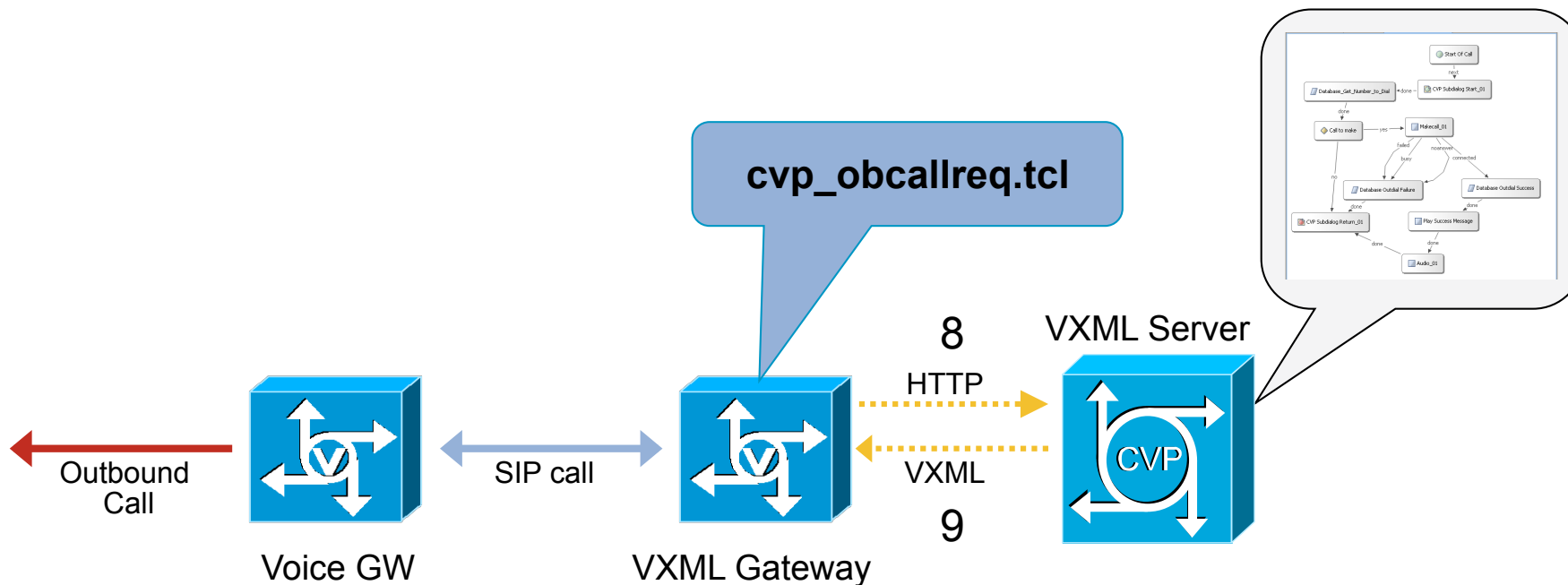
5. The cvp\_outbound session hands-off the call request details to another TCL session (cvp\_dialer) to make the call
6. The cvp\_dialer session makes the outbound call attempt
7. Then reports the call outcome back to cvp\_obcallreq via a SIP INFO message



# CVP VoiceXML Standalone Outbound

## How Does It Work?

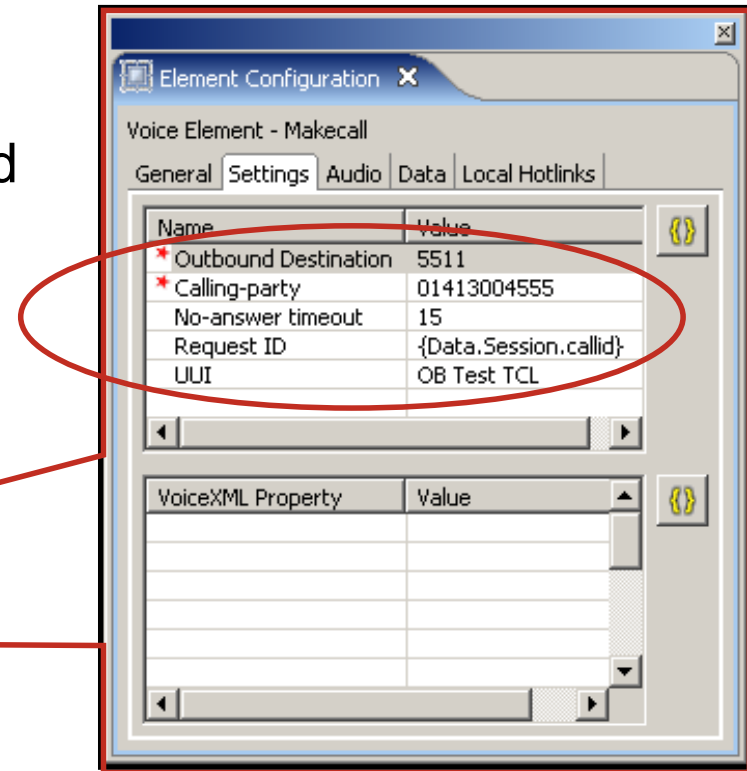
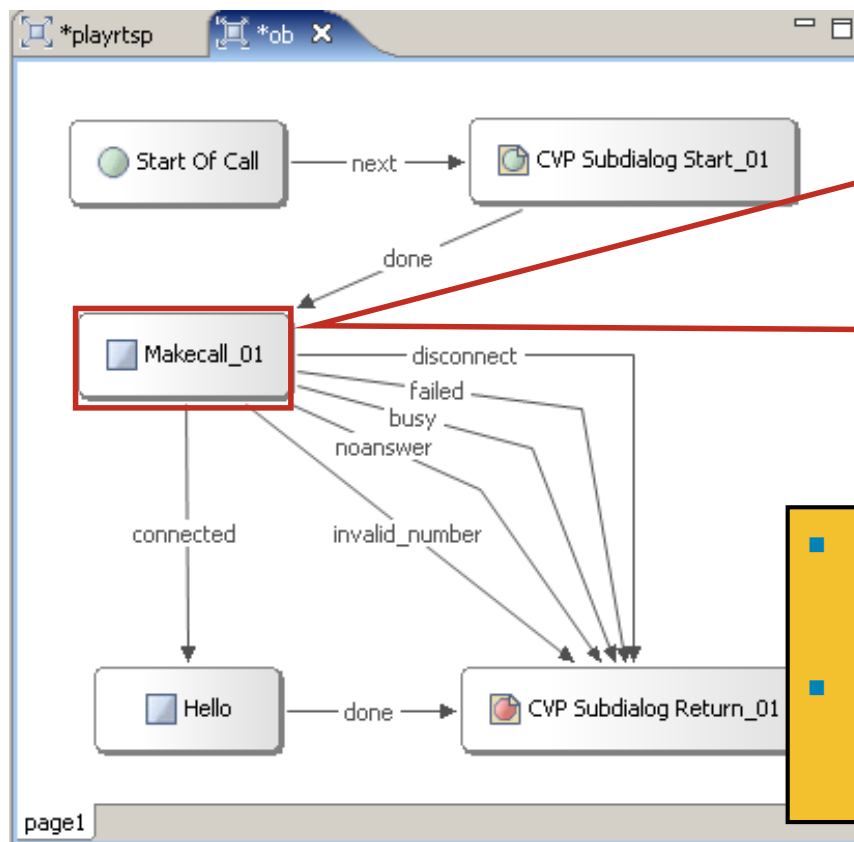
8. The `cvp_obcallreq` session returns the call outcome to VoiceXML and the result is submitted back to the VoiceXML application
9. The Makecall element exit path is taken in the script depending on the call outcome, commencing the actual call treatment if it was answered successfully





# Custom Makecall Element

- Use of custom Makecall element
- Uses configurable settings for outbound call parameters



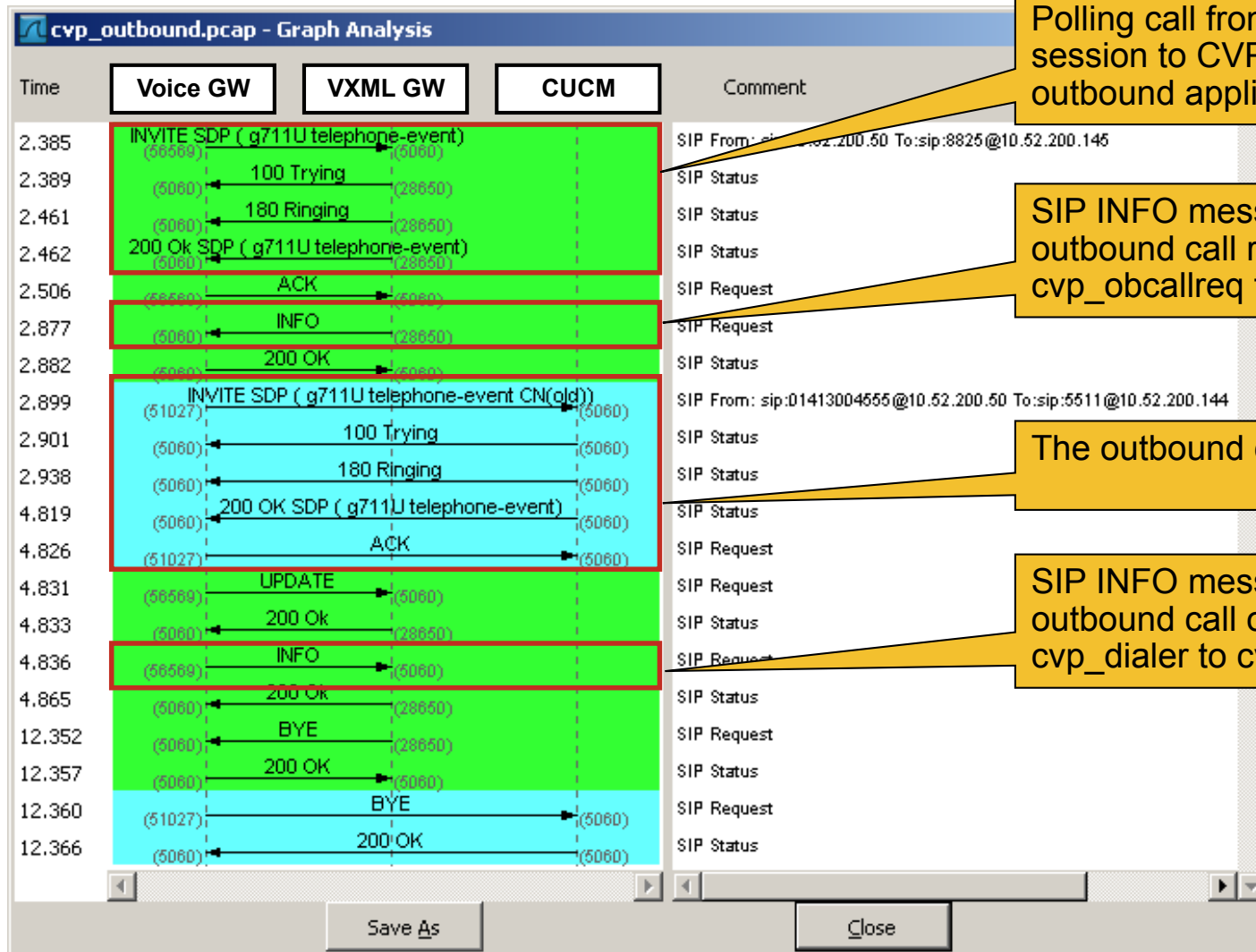
- Generates VoiceXML to perform handoff to TCL
- Outbound call parameters are passed to TCL via arg-string parameter

# VoiceXML Form Generated by Custom Makecall Element

```
<form id="start">
  <object name="obcallreq" classid="builtin://com.cisco.callhandoff">
    <param name="return" expr="true" valuetype="data" />
    <param name="app-uri" expr=""builtin://cvp_obcallreq"" valuetype="data" />
    <param name="arg-string" expr=""dnis=5511 cli=01413004555 rna=15 id=11112222-AAAABBBB-33334444
      uui=&quot;OB Test TCL&quot;;" valuetype="data" />
  </object>
  <block>
    <var name="call_outcome" expr="obcallreq.argstring" />
    <submit next="/CVP/Server" method="post" namelist=" audium_vxmlLog call_outcome" />
  </block>
</form>
```

- Makecall element generates form such as example above to perform TCL handoff to service `cvp_obcallreq`
- Outbound call parameters passed via “`arg-string`” parameter
- Note that UUI can also be sent with the call
- Call outcome data is returned via `<object-name>.argstring` and submitted back to the VoiceXML server

# Outbound Mechanism SIP Message Flow



Polling call from cvp\_outbound session to CVP VoiceXML outbound application

SIP INFO message delivering outbound call request from cvp\_obcallreq to cvp\_outbound

The outbound call itself

SIP INFO message returning outbound call outcome from cvp\_dialer to cvp\_ocallreq

# TCL Session Output Showing Progress (1)

## **CVP\_OUTBOUND session startup parameters**

09:51:44.806 CVP\_OUTBOUND: outbound CVP VXML application poll interval is <15> secs  
09:51:44.806 CVP\_OUTBOUND: poll destination number for outbound CVP VXML application is <8825>  
09:51:44.806 CVP\_OUTBOUND: started poll timer

## **CVP\_OUTBOUND makes SIP call to outbound VXML application**

09:51:59.806 CVP\_OUTBOUND: making call to poll CVP VXML application at <8825>

## **CVPSelfService.tcl shows call arrived**

09:51:59.886 TCL CVP: Call arrived with ANI=sip:10.52.200.145:5060, DNIS=sip:8825@10.52.200.50;transport=udp,  
CALLID=FE4CFABB-DD6911DD-8811E6FA-CA1D09A0  
09:51:59.946 CVP\_OUTBOUND: call to poll CVP VXML outbound application established

## **Outbound application has executed Makecall element and performs handoff to CVP\_OBCALLREQ**

09:52:00.802 CVP\_OBCALLREQ: assumed control of call with argument: <dnis=5511 cli=01413004555 rna=15 id=FE4CFABB-DD6911DD-8811E6FA-CA1D09A0 uui="OB Test TCL">

## **SIP INFO message is sent to CVP\_OUTBOUND**

09:52:00.806 CVP\_OBCALLREQ: sent dialer record to CVP\_OUTBOUND.TCL, waiting for call outcome ...  
09:52:00.850 CVP\_OUTBOUND: facility GTD report containing dialer record

## **CVP\_OUTBOUND makes another polling call immediately as it successfully received an outbound request This time the request hits the limit I configured for concurrent outbound calls**

09:52:00.854 CVP\_OUTBOUND: making call to poll CVP VXML application at <8825>  
09:52:00.854 %CALL\_CONTROL-6-MAX\_CONNECTIONS: Maximum number of connections reached for dial-peer 8825

## TCL Session Output Showing Progress (2)

### **CVP\_DIALER now makes the actual outbound call**

```
09:52:00.858 CVP_DIALER: assumed control of call with argument: <dnis=5511 cli=01413004555 rna=15 id=FE4CFABB-DD6911DD-8811E6FA-CA1D09A0 uui="OB Test TCL">  
09:52:00.862 CVP_OUTBOUND: ERROR: call to poll CVP VXML outbound application failed, status ls_003  
09:52:00.862 CVP_OUTBOUND: started poll timer  
09:52:00.870 CVP_DIALER: event ev_proceeding on call leg 1617  
09:52:00.914 CVP_DIALER: event ev_alert on call leg 1617  
09:52:08.430 CVP_DIALER: event ev_connected on call leg 1617  
09:52:08.434 CVP_DIALER: ID = FE4CFABB-DD6911DD-8811E6FA-CA1D09A0: call to 5511 established
```

### **Outbound call has been answered so SIP INFO message sent with outcome to CVP\_OBCALLREQ**

```
09:52:08.454 CVP_OBCALLREQ: dialer call outcome: connected
```

### **Outbound call has disconnected**

```
09:52:14.758 CVP_DIALER: event ev_destroy_done received
```

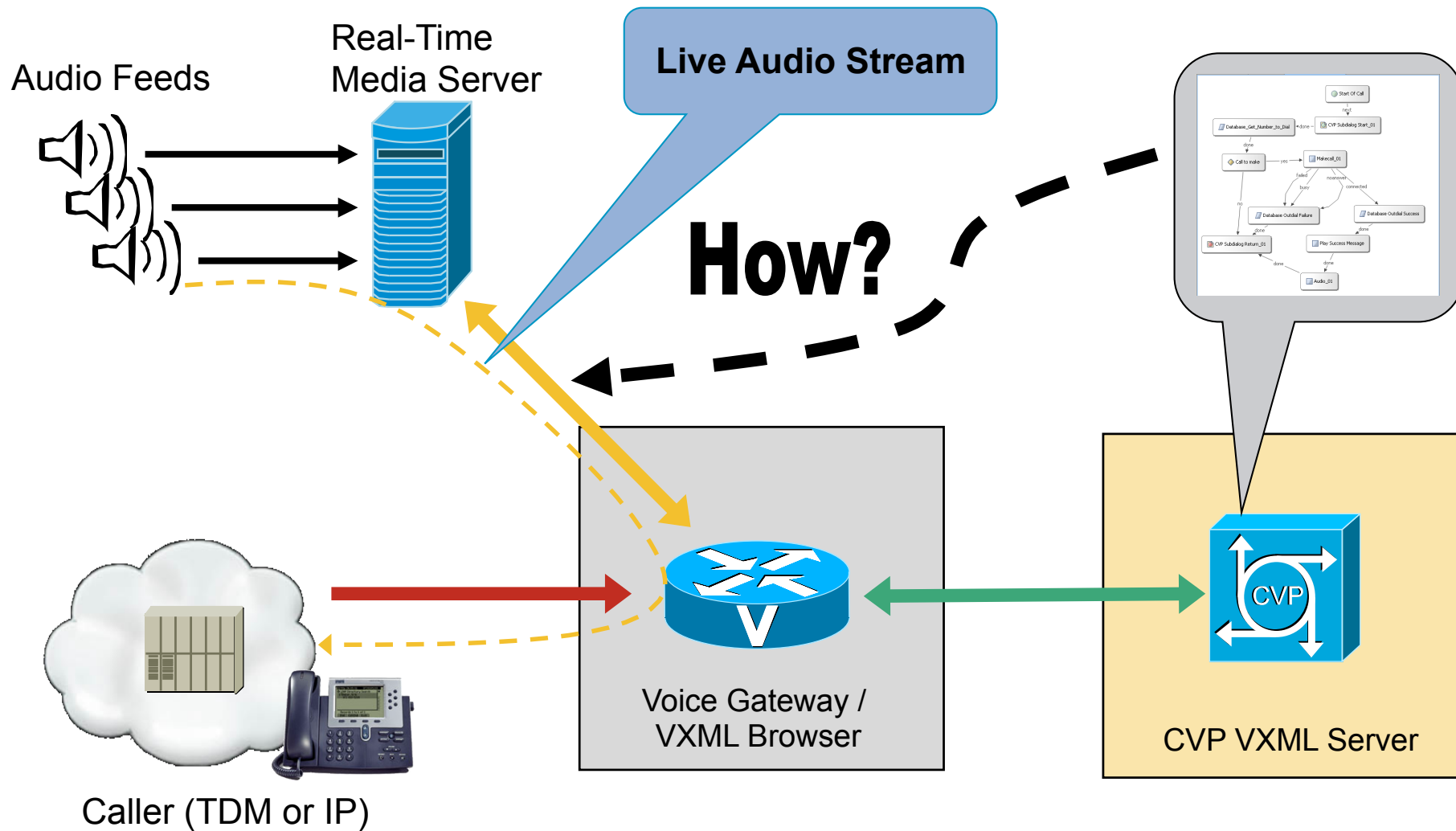
### **Poll sleep timer has now expired again so go through cycle again**

```
09:52:15.862 CVP_OUTBOUND: making call to poll CVP VXML application at <8825>
```

# Real-Time Audio Feeds as Prompts



# Playing Real-Time Audio Feeds



# Using RTSP Media Sources

- What's the issue? It's just another URI to an audio source
- Why not just do this --

The diagram illustrates a call flow where an audio prompt is played via RTSP. The configuration window shows the URI for the audio source is set to `rtsp://10.52.202.48/rtpencoder/liveaudio.sdp`.

- Does play the audio
- But there's a problem!
- You can't continue to the next script element unless the RTSP stream ends or disconnects



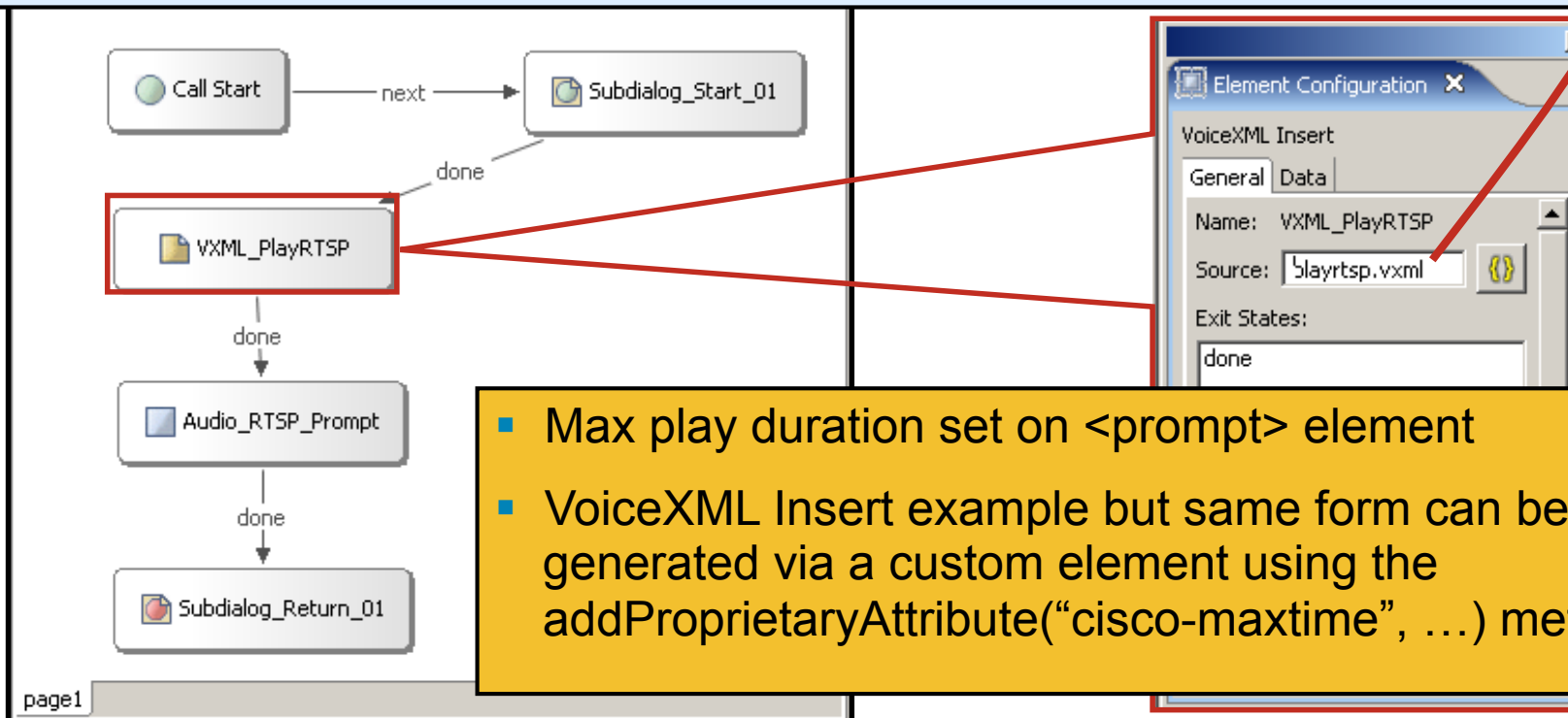
# Using RTSP Media Sources

- That approach is OK if you never want to do anything subsequently and the session timer is set long enough to not expire while the audio is playing
- Alternatively, use the “**cisco-maxtime**” attribute on the prompt element
- Plays the real-time feed up to the maximum seconds specified
- Unfortunately not an option out-the-box so simple customisation is necessary using either VoiceXML Insert or a custom voice element

# Using RTSP Media Sources

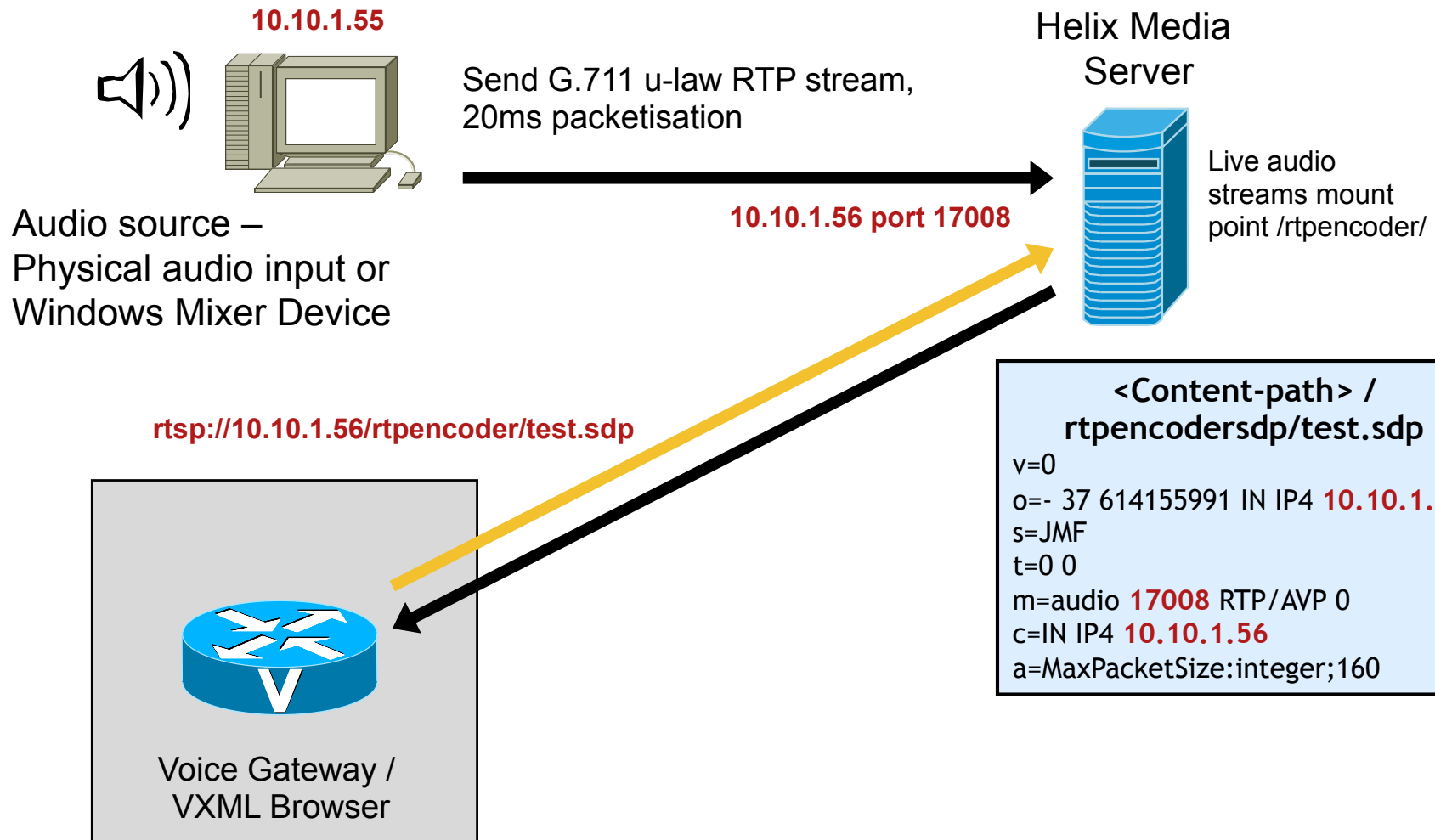
## Setting “cisco-maxtime” Duration

```
<form id="start">  
  <block>  
    <prompt bargein="true" cisco-maxtime="10s">  
      <audio src="rtsp://10.52.202.48/rtpencoder/liveaudio.sdp" />  
    </prompt>  
    <assign name="audium_exit_state" expr=""done"" />  
    <return namelist="audium_exit_state audium_vxmlLog audium_hotlink audium_hotevent audium_error  
      audium_action" />  
  </block>  
</form>
```



- Max play duration set on <prompt> element
- VoiceXML Insert example but same form can be generated via a custom element using the `addProprietaryAttribute("cisco-maxtime", ...)` method

# Using Helix For Streaming Live Audio (Example)

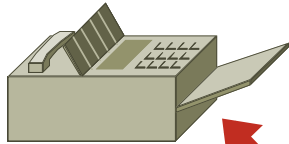


# Fax Detection



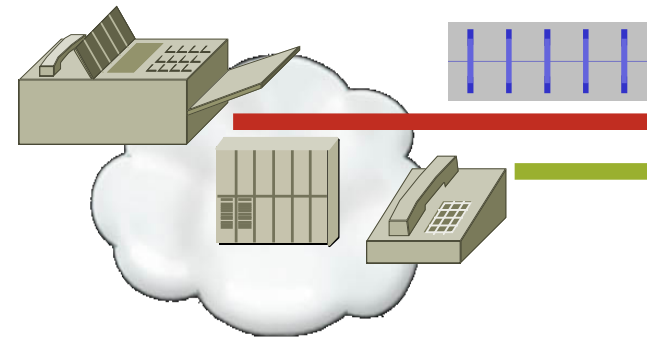
# Detecting Fax While Prompt Playing

Transfer Destination  
(Fax)

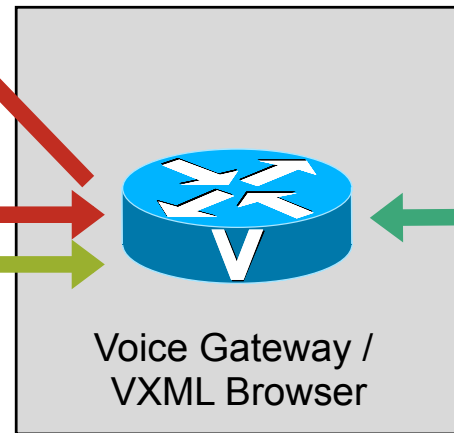


Fax Tone Detection

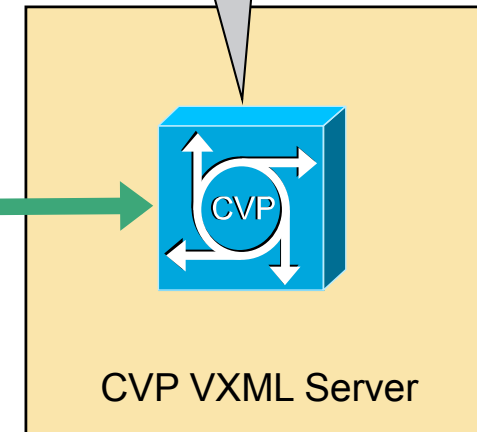
How?



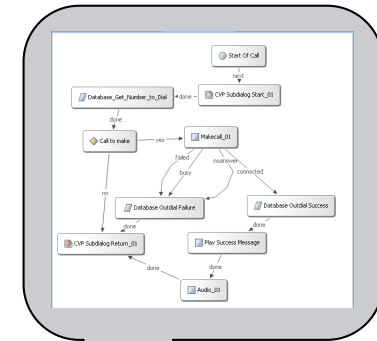
Caller (Phone or Fax)



Voice Gateway /  
VXML Browser



CVP VXML Server



# Fax Tone Detection

- Incoming call may be from a fax machine
- Requirement to treat fax calls differently from voice calls
- For example, transfer fax calls to a fax server for document retrieval as part of combo voice/fax service on one number
- Need to detect fax tones while initial welcome prompt is playing and report result to the VoiceXML application
- Not possible using standard audio or caller input capture elements
- How is it done? ...

# Fax Tone Detection

- The VoiceXML gateway can detect fax tones
- Throws event “**com.cisco.fax.cng**”
- So, need to do two things –
  1. Enable fax tone detection on the gateway
  2. Catch the event in the application

# Enabling Fax Tone Detection

- Could modify CVPSelfService.tcl but that would result in custom modification of a standard CVP script and would apply to all calls
- Better to use exactly the same approach as answering the call from the script
- Use a temporary handoff to TCL to enable CNG tone detection on the gateway
- Invoke via VoiceXML Insert or custom element

```
<form id="start">  
  <object name="faxenable" classid="builtin://com.cisco.callhandoff">  
    <param name="return" expr="true"/>  
    <param name="app-uri" expr="builtin://enable_cng"/>  
  </object>  
  
  <block>  
    <assign name="audium_exit_state" expr="done" />  
    <return namelist="audium_exit_state audium_vxmlLog audium_hotlink audium_hotevent audium_error  
      audium_action" />  
  </block>  
</form>
```

Invoke TCL script with return. The service name on the gateway in this example is "enable\_cng". This is configured on the gateway to map to the actual TCL script.



# TCL Example Enabling CNG Tone Detect

```
#-----  
# Enable CNG tone detection and return to calling application  
#-----  
  
proc handoff_event { } {  
    set leg1 [infotag get evt_legs]  
    set min_cng_time [leg tonedetect $leg1 enable cng]  
    puts ">>> ENABLE_CNG.TCL minimum detection time $min_cng_time secs"  
    handoff return $leg1  
}  
  
#-----  
# State machine  
#-----  
  
set FSM(CALL_INIT, ev_handoff) "handoff_event,same_state"  
set FSM(any_state, ev_disconnect_done) "cleanup,same_state"  
set FSM(any_state, ev_disconnected) "disconnect,same_state"  
  
fsm define FSM CALL_INIT
```

Performs leg tonedetect to enable CNG detection

Return control back to VoiceXML

Handoff event triggers TCL handoff\_event() procedure

Note. Some detail omitted for clarity such as disconnect handling

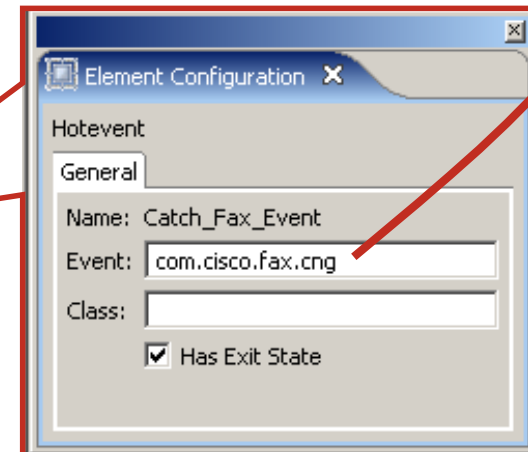
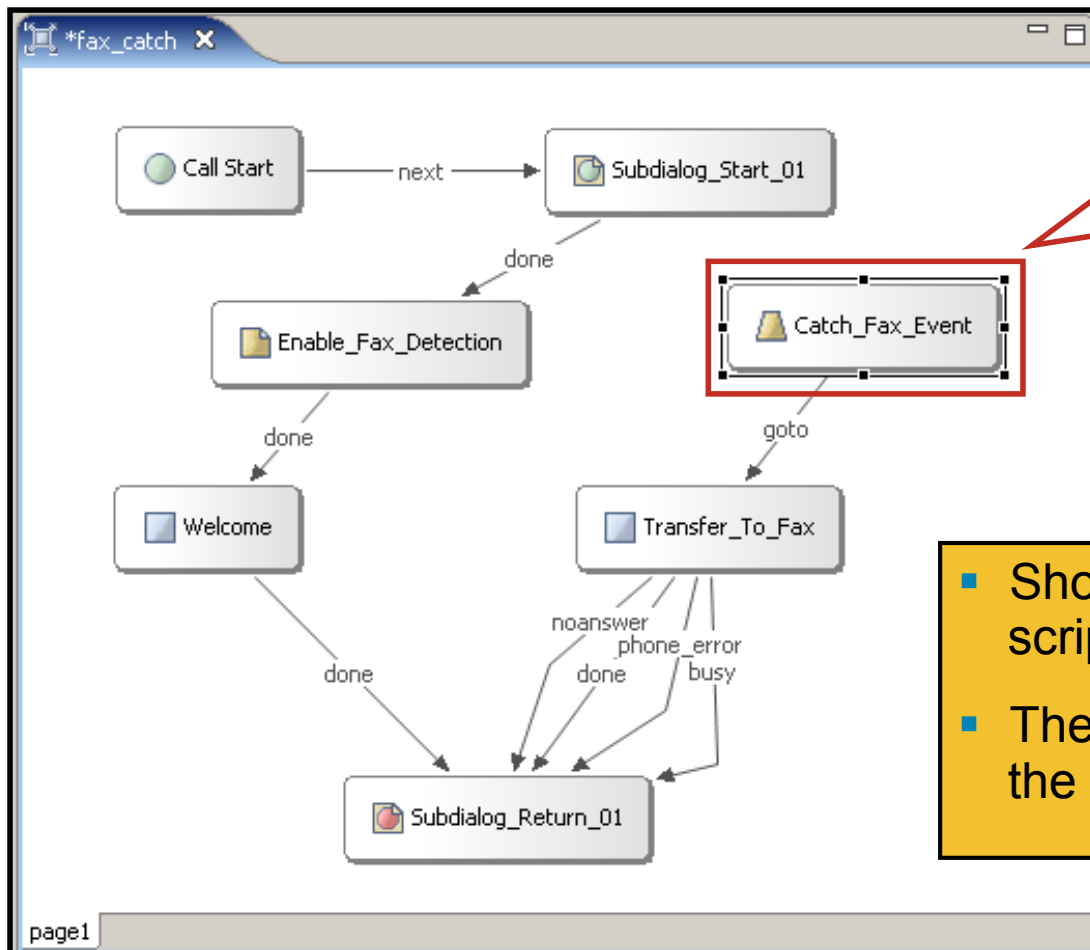
- Handoff event handler is invoked via the state machine
- It retrieves the ID of the call leg that it now has control of
- Performs a leg tonedetect on the call leg for CNG tone
- Returns control back to VoiceXML

# Catching the Event

- Could use a VoiceXML Insert or custom audio element which both enables tone detection and includes a catch element for “com.cisco.fax.cng”
- But, far easier to use a Hotevent which inserts a catch block for this event in the root document
- Can then use standard Audio elements

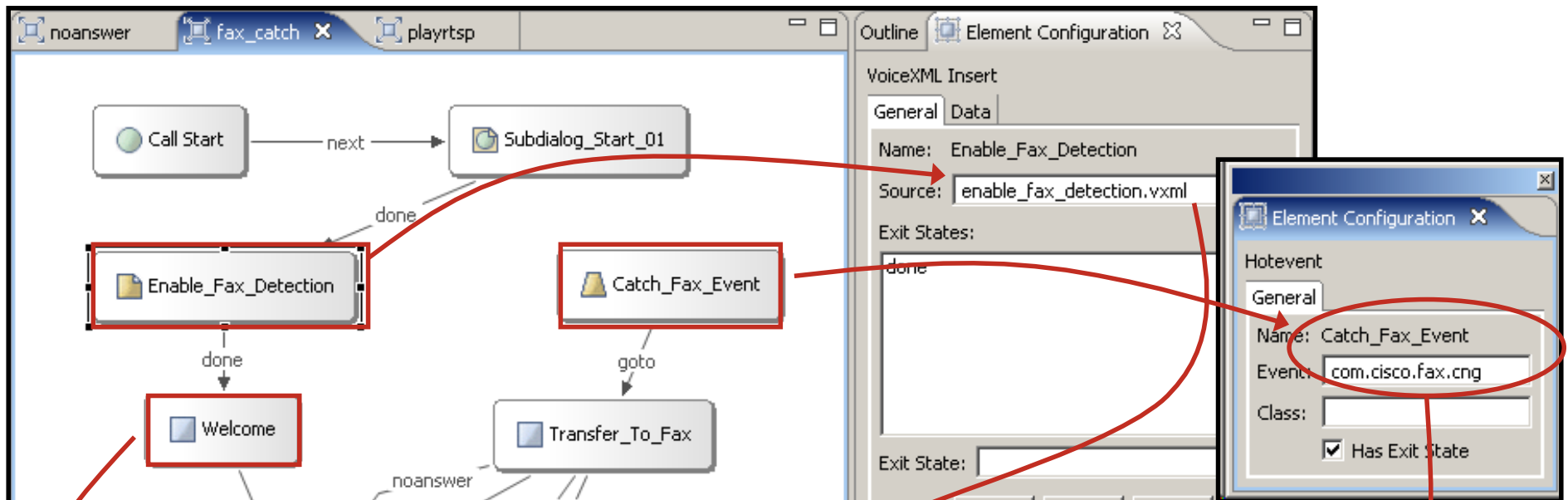
# Catching com.cisco.fax.cng

```
<catch event="com.cisco.fax.cng">  
  <var name="audium_hotevent" expr="Catch_Fax_Event" />  
  <submit next="/CVP/Server" method="post" namelist=" audium_vxmlLog audium_hotevent" />  
</catch>
```



- Shows use of Hotevent in the script
- The resulting catch element in the root document

# Fax Detection Example



```
<subdialog name="sub" src="enable_fax_detection.vxml">
*Jan 15 14:05:06.143: ***FAX DETECT VXML, handing off to ENABLE CNG TONE TCL
*Jan 15 14:05:06.147: //214//TCL :/tcl_PutsObjCmd: >>> ENABLE_CNG.TCL minimum detection time 9 secs
*Jan 15 14:05:06.151: ***FAX DETECT VXML, returned from TCL
```

```
<audio src="/CVP/audio/music-on-hold.au" />
```

```
POST /CVP/Server HTTP/1.1
audium_vxmlLog=&audium_hotevent=Catch_Fax_Event
```

```
<transfer name="Mtransfer" bridge="false" connecttimeout="60s" maxtime="0s" dest="phone://97780021">
```

```
192.168.1.22.1232024662562.20.fax_catch,01/15/2009 05:04:30.171,Subdialog_Return_01,element,hotevent,Catch_Fax_Event
192.168.1.22.1232024662562.20.fax_catch,01/15/2009 05:04:30.171,Subdialog_Return_01,exit,
192.168.1.22.1232024662562.20.fax_catch,01/15/2009 05:04:30.171,Transfer_To_Fax,enter,
192.168.1.22.1232024662562.20.fax_catch,01/15/2009 05:04:34.203,Transfer_To_Fax,exit,
192.168.1.22.1232024662562.20.fax_catch,01/15/2009 05:04:34.203,,end,how,call_transfer
```

# Agenda

- Brief Overview of CVP Standalone VoiceXML
- Incoming Calls and Data Flow
- How-Is-It-Done Tips and Tricks
- **Optimal Prompt Caching**
- Custom Script Elements

# Prompt Caching



# Prompt Caching

- Locating prompts at the gateways can save bandwidth but hard to maintain, especially if large, highly-distributed deployment
- Centralised media server is easier to maintain but don't want to serve repeatedly the same prompts to gateways at every use
- If using “no ivr prompt streamed http” then the http cache is used at the gateway for caching prompts
- Prompt refresh attempt will be made only when the prompt is “stale” and next used
- Prompt refresh time is determined primarily by media server setting returned in the HTTP header but value set with “http client cache refresh <secs>” is used as a last resort

# Prompt Caching – Forcing Refresh

- To force a cache refresh, use new IOS command:  
**“set http client cache stale”**
- When prompt is next used the HTTP GET is sent to the server with the If-Modified-Since header set to the last-modified date/time
- Prompt will only be retrieved if the server copy is newer



# Prompt Caching

## 1 Force Refresh – Prompt Not Modified

```
Router3#set http client cache stale
```

Ref	FreshTime	Age	Size	context
1	3600	1667	# 43320	0

url: http://bristol:7000/CVP/audio/app/welcome.wav

Welcome.wav is marked as stale

```
<prompt bargein="true">  
  <audio src="/CVP/audio/app/welcome.wav" />  
</prompt>
```

Retrieve from media server if modified

```
GET /CVP/audio/app/welcome.wav HTTP/1.1  
Host: bristol:7000  
If-Modified-Since: Wed, 07 Jan 2009 22:59:25 GMT
```

```
*Jan 7 23:44:08.327: Request msg: GET /CVP/audio/app/welcome.wav HTTP/1.1  
HTTP/1.1 304 Not Modified  
Server: Apache-Coyote/1.1  
Date: Wed, 07 Jan 2009 23:50:36 GMT  
Connection: close
```

Not modified so not refreshed and age in cache is reset

Ref	FreshTime	Age	Size	context
1	3600	29	43320	0

url: http://bristol:7000/CVP/audio/app/welcome.wav

# Prompt Caching

## 2 Force Refresh – Prompt Modified

```
Router3#set http client cache stale
```

Ref	FreshTime	Age	Size	context
1	3600	242	# 43320	0

url: http://bristol:7000/CVP/audio/app/welcome.wav

```
<prompt bargein="true">  
  <audio src="/CVP/audio/app/welcome.wav" />  
</prompt>
```

```
GET /CVP/audio/app/welcome.wav HTTP/1.1  
Host: bristol:7000  
If-Modified-Since: Wed, 07 Jan 2009 22:59:25 GMT
```

```
*Jan 7 23:48:18.043: Request msg: GET /CVP/audio/app/welcome.wav HTTP/1.1  
HTTP/1.1 200 OK  
Server: Apache-Coyote/1.1  
ETag: W/"43320-1231372477201"  
Last-Modified: Wed, 07 Jan 2009 23:54:37 GMT  
Content-Type: audio/x-wav  
Content-Length: 43320  
Date: Wed, 07 Jan 2009 23:54:45 GMT  
Connection: close
```

Ref	FreshTime	Age	Size	context
1	3600	72	43320	0

url: http://bristol:7000/CVP/audio/app/welcome.wav

Prompt has been modified on the server so this time the prompt is retrieved

# Prompt Caching

## 3 Force Refresh – Prompt Not Modified

```
Router3#set http client cache stale
```

Ref	FreshTime	Age	Size	context
1	3600	136	# 43320	0

url: http://bristol:7000/CVP/audio/app/welcome.wav

```
<prompt bargein="true">  
  <audio src="/CVP/audio/app/welcome.wav" />  
</prompt>
```

```
GET /CVP/audio/app/welcome.wav HTTP/1.1  
Host: bristol:7000  
If-Modified-Since: Wed, 07 Jan 2009 23:54:37 GMT
```

```
*Jan 7 23:50:49.211: Request msg: GET /CVP/audio/app/welcome.wav HTTP/1.1  
HTTP/1.1 304 Not Modified  
Server: Apache-Coyote/1.1  
Date: Wed, 07 Jan 2009 23:57:16 GMT  
Connection: close
```

Ref	FreshTime	Age	Size	context
1	3600	30	43320	0

url: http://bristol:7000/CVP/audio/app/welcome.wav

Retrieve from media server if modified, now using new modified date/time

Not modified so not refreshed and age in cache is reset

# Agenda

- Brief Overview of CVP Standalone VoiceXML
- Incoming Calls and Data Flow
- How-Is-It-Done Tips and Tricks
- Optimal Prompt Caching
- **Custom Script Elements**

# Customising Elements

- Let's revisit our RTSP audio playing example
- The requirement is to perform a <prompt> element which includes the “**cisco-maxtime**” attribute
- A VoiceXML Insert script element allows us to execute hand-crafted static VoiceXML or dynamically generated if the URI points to a servlet or similar
- The static approach has drawbacks with regard to reusability

Use of “expr” attribute allows a CVP element/session variable to be used to substitute the audio src URI dynamically

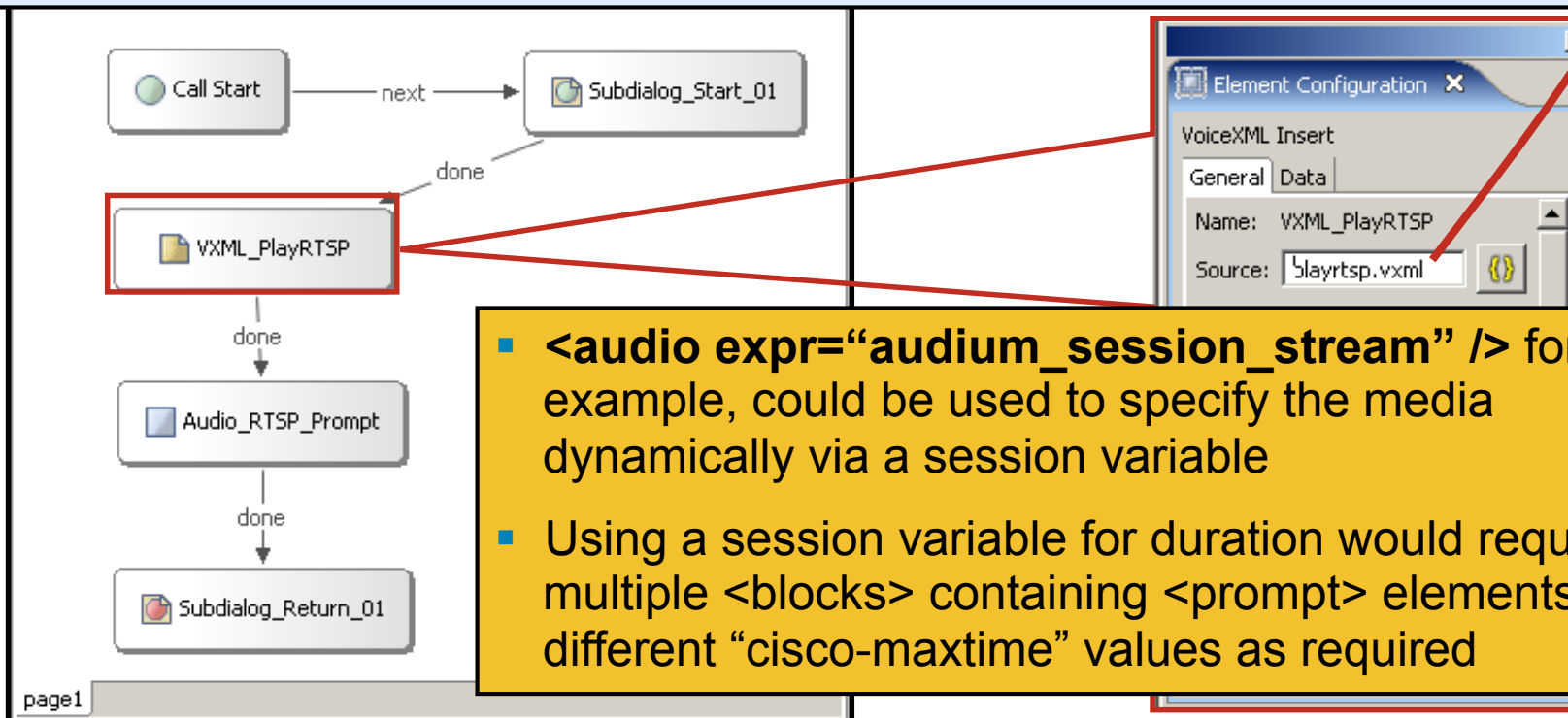
But, the “cisco-maxtime” attribute only allows a constant

To support multiple maxtime values a static document would require multiple form items branched-to using <goto> with “expr” or “exritem” attributes comprising CVP element/session variables

# Customising Elements

## Setting “cisco-maxtime” Duration

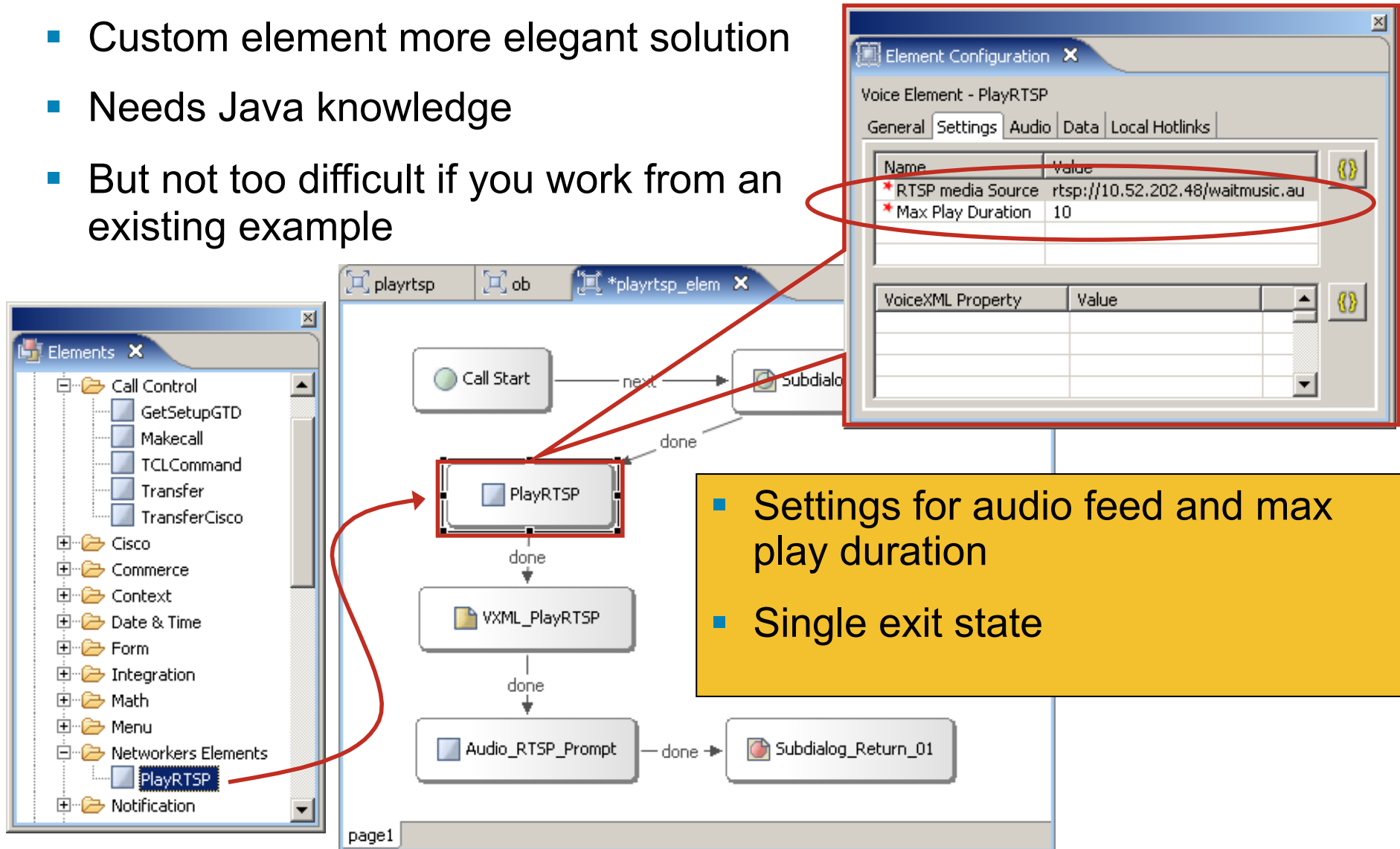
```
<form id="start">
  <block>
    <prompt bargein="true" cisco-maxtime="10s">
      <audio src="rtsp://10.52.202.48/rtpencoder/liveaudio.sdp" />
    </prompt>
    <assign name="audium_exit_state" expr=""done"" />
    <return namelist="audium_exit_state audium_vxmlLog audium_hotlink audium_hotevent audium_error
      audium_action" />
  </block>
</form>
```



- `<audio expr="audium_session_stream" />` for example, could be used to specify the media dynamically via a session variable
- Using a session variable for duration would require multiple `<blocks>` containing `<prompt>` elements with different “cisco-maxtime” values as required

# Alternative To VoiceXML Insert Use a Configurable Custom Element

- Custom element more elegant solution
- Needs Java knowledge
- But not too difficult if you work from an existing example



# PlayRTSP Custom Element Output

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.1" application="/CVP/Server?
audium_root=true&amp;calling_into=playrtsp_elem
&amp;session_id=10.52.202.38.1231626077250.271.playrtsp_elem">
<form id="audium_start_form">
  <block>
    <assign name="audium_vxmlLog" expr="" />
    <assign name="audium_element_start_time_millisecs" expr="new Date().getTime()" />
    <goto next="#start" />
  </block>
</form>
<form id="start">
  <block>
    <prompt bargein="true" cisco-maxtime="10s">
      <audio src="rtsp://10.52.202.48/waitmusic.au" />
    </prompt>
    <submit next="/CVP/Server" method="post" namelist=" audium_vxmlLog" />
  </block>
</form>
</vxml>
```

- Output resulting from custom script element and settings
- So, what's involved in building the custom element? ...



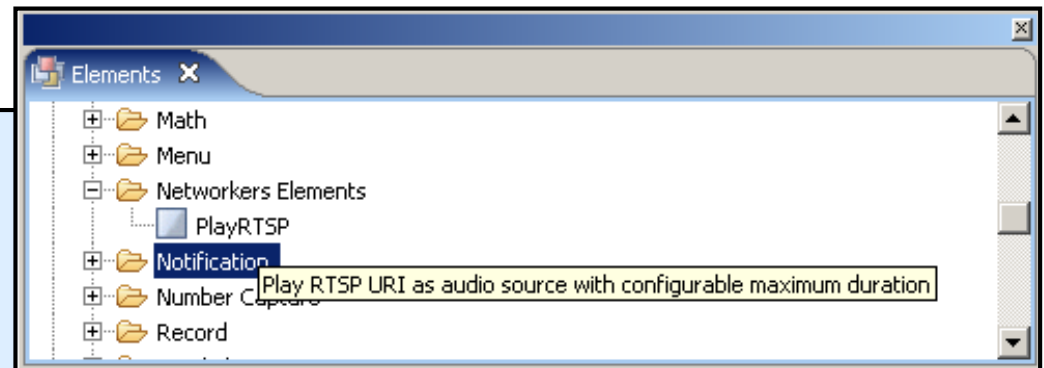
# Creating PlayRTSP Custom Element

- Start with existing VoiceElement example or the template MyVoiceElement.java
- 2 parts to code – element configuration and the VoiceXML body
- Configuration – complete the following methods

```
public String getElementName()  
{  
    return "PlayRTSP";  
}
```

```
public String getDescription()  
{  
    return "Play RTSP URI as audio source with configurable maximum duration";  
}
```

```
public String getDisplayFolderName()  
{  
    return "Networkers Elements";  
}
```



**Name, description, folder in palette**

# Creating PlayRTSP Custom Element

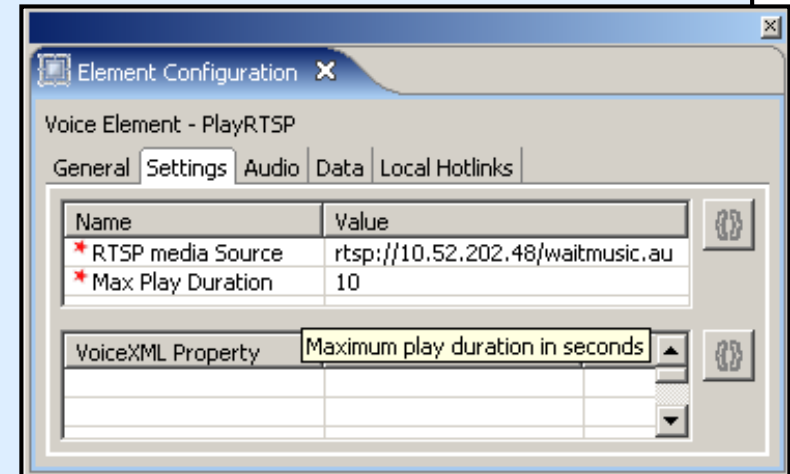
```
public Setting[ ] getSettings() throws ElementException
{
    Setting src = new Setting("RTSP_source", "RTSP media Source",
        "RTSP media stream URI",
        true, true, true, Setting.STRING);

    Setting dur = new Setting("Max_time", "Max Play Duration",
        "Maximum play duration in seconds",
        true, true, true, 0, null);

    dur.setDefaultValue("0");
    Setting[ ] cfg_settings = new Setting[ ] {src, dur};
    return cfg_settings;
}
```

```
public ExitState[ ] getExitStates() throws ElementException
{
    ExitState done = new ExitState("done", "done", "Command invoked");
    ExitState[ ] exitStateArray = new ExitState[ ] {done};
    return exitStateArray;
}
```

## Configuration Settings



## Exit States



# Creating PlayRTSP Custom Element

```
protected String addXmlBody(VMain vxml, Hashtable reqParameters, VoiceElementData ved)
    throws VException, ElementException
{ ...
```

**Build VoiceXML**

```
// Start building the VXML doc by adding main form which must be called "start"
```

```
VForm prompt_form = VForm.getNew(pref, "start");
```

**Create <form id="start">**

```
// Create block and add to form
```

```
VBlock prompt_block = VBlock.getNew(pref);
prompt_form.add(prompt_block);
```

**Create and add <block>**

```
// Add audio element to block followed by submit
```

```
VAudio media_item = VAudio.getNew(pref, src, VAudio.FILENAME_ONLY);
media_item.addProprietaryAttribute("cisco-maxtime", dur);
media_item.setUseDefaultAudioPathForAll(false);
prompt_block.add(media_item);
```

**Create and add <prompt>  
Add "cisco-maxtime"**

```
VAction prompt_submit = VAction.getNew(pref);
prompt_submit.add(getSubmitVAction(null, pref));
prompt_block.add(prompt_submit);
```

**Create and add <submit> action**

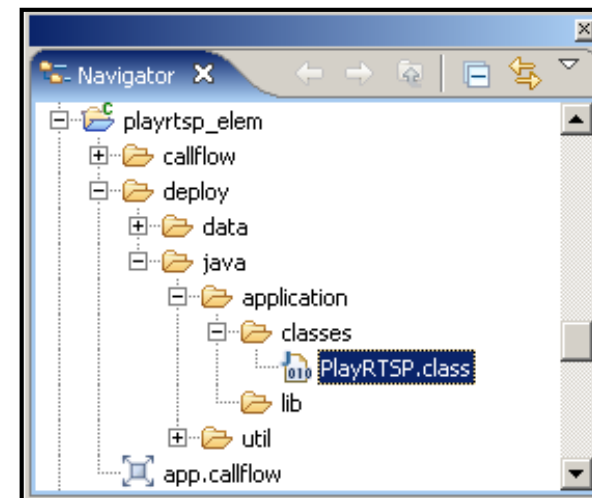
```
// Add form to VXML doc
```

```
vxml.add(prompt_form);
```

**Add form to VoiceXML document**

# Deploying PlayRTSP Custom Element

- Build Java class or jar file
- To make available to Call Studio palette, locate in –  
...\`CallStudio\ eclipse\ plugins\ com.audiumcorp.studio.library.common_<vers>`
- On server locate in –  
`C:\Cisco\CVP\VXMLServer\common\lib`
- Or add the class/jar to the specific project folder “\`deploy\ java\ application\ classes`” or “\`lib`” and deploy with the project



# CDC / Developer Forum

The screenshot shows the Cisco Developer Community website in a Mozilla Firefox browser window. The address bar displays <http://developer.cisco.com/web/cvp/home>. The page features a navigation menu with tabs for Home, News & Events, Tech Centers (selected), Support, Library, Community, and Developer Partner. Below the navigation, there are links for Customer Voice Portal, Forums, Blogs, Wiki, Downloads, Documentation, and Video Tutorials. The main content area is titled 'Intelligent Voice and Video Self-Service' and includes a paragraph about the Customer Voice Portal's capabilities and a bulleted list of use cases. A 'Site Map' sidebar on the right lists various resources like Overview, Getting Started, Resources, Downloads, Documentation, Video Series, Forums, Blogs, Wiki, and Video Tutorials. At the bottom, there are sections for 'What Is It?', 'How Do I Get Started?', and 'What Resources Are Available?'.

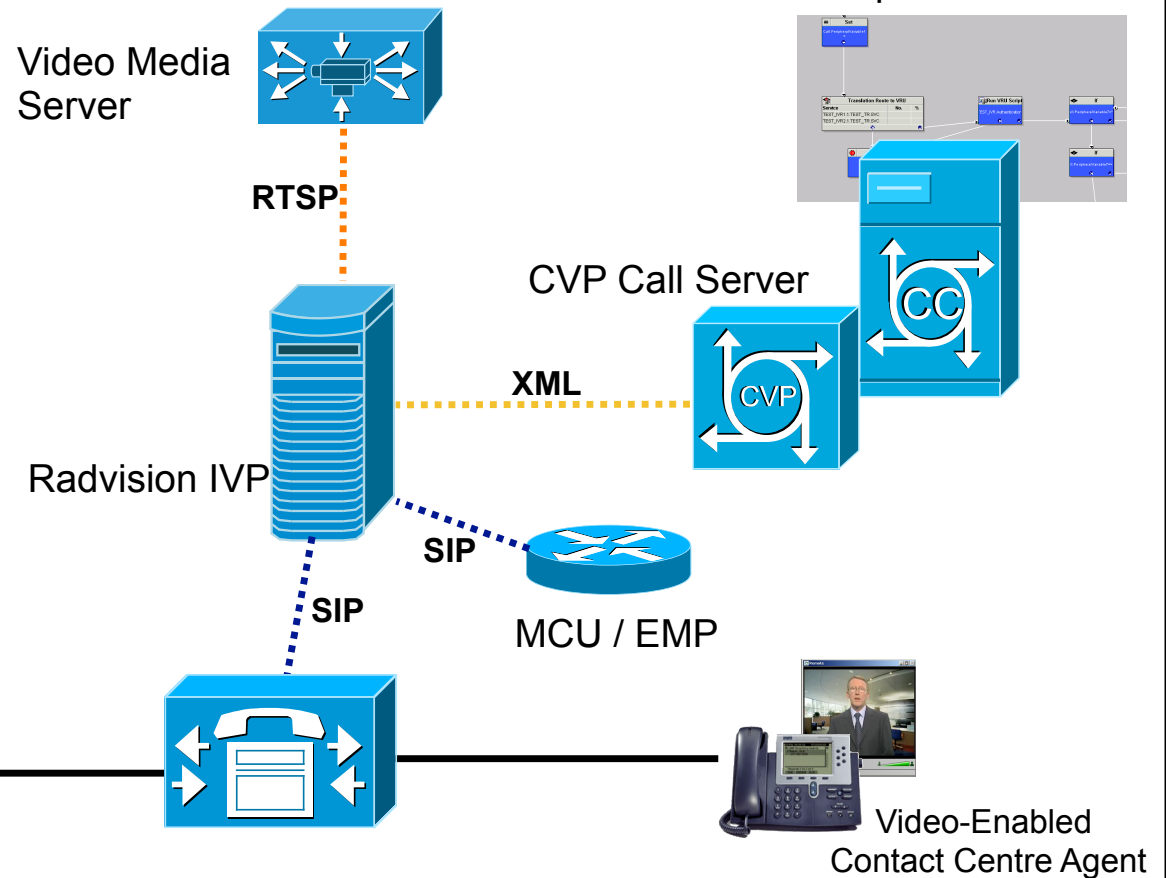
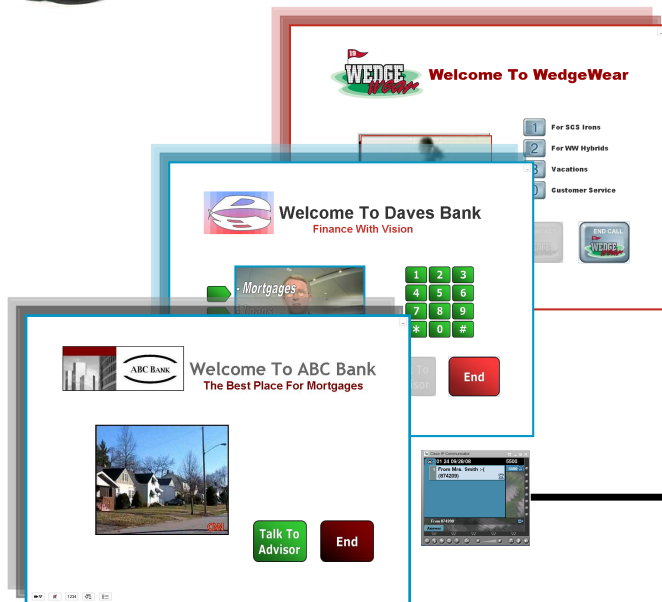
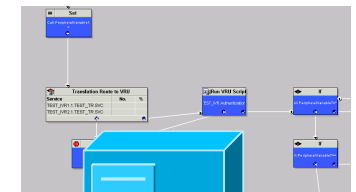
- <http://developer.cisco.com>
- Use the CVP Developer Community for discussion forums, documentation, code samples
- Outbound sample application is available in “Downloads”

# CVP – Not Just About Voice Also, Video Self-Service Applications



- Uses CVP Full Video Service model including Radvision IVP for video IVR and ICM for scripting
- Applications such as self-service video kiosks for branch, retail, public libraries, deaf-signing

Video application scripted on ICM



# Meet The Expert

- To make the most of your time at Cisco Networkers 2009, schedule a Face-to-Face Meeting with a top Cisco expert.
- Designed to provide a "big picture" perspective as well as "in-depth" technology discussions, these face-to-face meetings will provide fascinating dialogue and a wealth of valuable insights and ideas.
- Visit the Meeting Centre reception desk located in the Meeting Centre in World of Solutions

# Recommended Reading

- There are currently no Cisco Press Books recommended for this Presentation - please browse the Cisco Company Store for suitable titles





**CISCO**