

Contact Center IVR Development Guide

February 2024

sinch.com



Revision History

Date	Description
01.10.2014	First version of the document
22.12.2020	Rebranded for Sinch
22.09.2021	Checked and updated content for validity and marked parts that only concern on-premise installations
08.02.2024	Reformulated note on using Python customizers in Sinch Contact Pro cloud.



Table of Contents

1 Getting Started	5
1.1 Personalizing System Configurator	5
1.2 User Rights	5
1.3 Configuring Customized IVR Prompts	6
2 Creating an IVR Application	8
2.1 Add New Application	8
2.2 IVR Editor	8
2.3 Adding Information to Elements	9
2.4 Maximizing the Work Area	10
3 VoiceXML Features	11
3.1 Supported Elements	11
3.2 Commonly Used VoiceXML Elements	11
3.2.1 var	11
3.2.2 form	12
3.2.3 block	12
3.2.4 field	13
3.2.5 transfer	15
3.2.6 soap	15
3.2.7 customstate	16
3.2.7.1 When to use custom functions	17
3.2.8 assign	18
3.2.9 audio	19
3.2.10 goto	20
3.2.11 if, elseif & else	21
3.2.12 prompt	23
3.3 Call Attached Data	24
4 Best Practices for IVRs	26
4.1 Basic Functions (ANumber, BNumber, and so on)	26
5 Troubleshooting	27
5.1 Business Communication Management Log Viewer (BLV)	27
5.2 BLV Find dialog	28
5.3 Turning Collapsed/Tagged-only mode on and off	30
5.4 Changing the Font	31
5.5 F1 Help Commands	32
5.6 Debugging your IVR	33
5.6.1 Setting the CEM log levels to debug	33
5.6.2 Locating the CEM log	34
5.6.3 Reading the CEM log	34
5.6.3.1 Finding log entries related to your IVR	34



5.6.3.2 Adding log entries into your IVR	.36
6 Python	.39
6.1 Manipulating Strings	.39
6.2 Some other Python expressions used in IVR Editor	.41
6.3 Dictionaries, Lists and Tuples	.41
6.4 Dictionaries, Lists and Tuples – Examples	.43
6.5 Accessing Data in Dictionaries	.43
6.6 Accessing Data in Dictionaries – Complex Example	.45
6.7 Accessing Data in Lists and Tuples	.45
7 Python Customizer Example	.47
7.1 DB Query Customizer Sample	.49
8 SoapUI	.50
8.1 Test Your Web Services	.50
8.2 Using Soap Elements inside Sinch Contact Center IVR	.53
8.3 Adding a log entry in your IVR to return a value from a Soap or custom state element	.57
8.4 Evaluating the results of a Soap query	.58
8.5 Using the Python syntax of Dictionaries, Lists and Tuples	.60
8.6 Using counters	.62
8.7 Using Soap elements with multiple input parameters	.62
8.8 Complex Query structure – how to add as parameters	.64
8.8.1 Mapping the xml structure to Python elements	.64
8.8.2 Mapping the xml structure to Python syntax	.64
9 Appendix 2	.67
10 Appendix 3	.69
11 Appendix 4	.71



About this Document

This document is an extension to the System Configurator guide and is aimed at administrators who create IVR applications in System Configurator (SC).

NOTE Your SC application may appear different from the screenshot examples because of the used version and language.



1 Getting Started

1.1 Personalizing System Configurator

In larger installations, you may want to change the maximum number of search results returned, especially when you do not enter a search term.

File Personalize Help		Saved Searches 💽 Go Search 🗊
¥μ. ⊮†	Home	Back
Home Gin System Services System Management Call Switching Reporting Scripting Management Presence Management Outevory Management VR Management VR Management User and Role Management System Tools	Easics × Language English (US) Style Nimbus Look and Feel Log Level Informational Messages Maximum Number of Search Results 999 Ø Display Technical Name Ø Display Default Values in ØK Cancel	History

1.2 User Rights

You must have rights to create, view and modify custom IVRs.



▶↓ ▶↑ User	r and Role Man	agement : Roles : -CO	C365 IVR Adminis	trator				
T 🚔 Home	ave 🔚 Save a	nd Close 🗊 Delete	Copy 🖓 Add N	ew 🖾 Reset 🗶	Close			s
System Services								
► 📄 System Management 🛛 👻 E	Basics							
► 📄 Call Switching	me 🔽	C265 IVP Administrat	or					
► È Reporting		COUSTING Administrat	.01					
Scripting Management De:	scription							
Directory Management								
▶ 📄 Presence Management								
🕨 📄 Queue Management 🛛 🗸 🗸	Jser Rights							
► IVR Management	Remove			View Type	Basic T	pe Custom IVRs		View Sun
Outbound Management								
🔻 📄 User and Role Management	Name	View	Modify	Create	Delete	Manage	Control Contact	Statistics Sum
Users	All	\checkmark	\checkmark	\checkmark	\checkmark	v		
🗎 User Roles								
User Groups								
User Settings Template								
🕒 Update Wizard								
System Tools								

1.3 Configuring Customized IVR Prompts

- In on-premise environments, place your customized prompt file in any folder in the Sinch Contact Center server. For example: C:\Sinch\ContactCenter\. In cloud environments, start from step 2.
- 2. Using System Configurator, open *Prompt Files* and add a prompt file to the Sinch Contact Center system:
 - 1. Select prompt type IVR Prompt.
 - 2. Select language English (US) because it is the default language. Wav files for other languages can also be added.
 - 3. Browse to the correct file in the folder:

₩. ¥T	Queue Managen	nent : Prompt File : Outbound Subscription	Renewal	Back
▼ 🚔 Home	📙 Save 😓 Sav	e and Close 🎁 Delete 🍸 Add New 崎 R	eset 🗙 Close Show ID	18, N M
System Services				
System Management	 Basics 			
Call Switching	Description	Outbound Subscription Renewal		
Scripting Management	Promot Type	IV/R Prompt		
Directory Management	Trompt Type	IVR Floinpt		
Presence Management	x Language Fi	lac		
🔻 🚞 Queue Management	Languageri	165		
Cueues	Delete			
📄 Queue Groups	C. Language		🔺 🧟 Open 🛛 🕹 🕹	
Calendars	English (U	IS)		60
Schedules			Look In: 🖆 Sinch 🔹 👔	
Prompts				
Prompt Files	n		💾 Renewal.wav	
Skills Baply Templates				
Third-Party Extensions				
► IVR Management				
Outbound Management				
🔻 🚞 User and Role Management				
🕒 Users	Access Right	ts		
User Roles	▼ Language	e Files		
User Groups	Delete			
User Settings Template	🕞 Langua	age	File Name	
Opdate wizard System Tools	English) (US)	6C291D901C8F48A3B3B4962F4B20862F way	
- Gystein roots	Crigiton	(00)		
			History & Connected	



- 4. Click **Save**. The file is saved with a GUID name. The system asks if you want to create a prompt connected to the new prompt file.
- 3. Answer Yes and Open.

The system uses default options to create a prompt from the new prompt file.

4. Accept the new IVR prompt by clicking Save (or Save and Close).
After creating the prompt click Refresh to see the new prompt in the Queue
Management > Prompts list.

Note: In the way instructed above, IVR prompt files will be created as GUID into the system. Therefore, prompts cannot be transferred to another Sinch Contact Center system as such, but the above steps must be repeated in the target Sinch Contact Center system.

In on-premise environments, the alternative way is to use a folder for storing prompt files:

- The prompt path can be a local path or a shared folder on a file server. For example: \\SERVERNAME\prompts.
- The advantage of a shared folder is that you do not need to copy the prompts to each server when a failover setup with multiple servers is used.
- You can create your own folders using Windows Explorer under this folder (for example, ME_IVRPrompts) and then refer to this folder in your IVRs.



2.1 Add New Application

- 1. Enter a name and description for the application.
- Select Active and Add Custom IVR to Reported Applications, and optionally Early Queuing. To learn more about early queuing, see System Configurator document (Call Switching > Global Switching Settings > Managing Signaling > Early Queuing and Toll-Free Queuing).
- 3. Add IVR number(s).
- 4. To start creating your new IVR, click Add New.

IVR Manageme	VR Management : Custom IVR : <new></new>							Bac	:k
🔚 Save 🍓 S	ave and Close 👕 Delete 崎 Reset 🕻	X Close			Show ID	Number Viewer	2,	ÞĻ -	ÞŤ.
 Basics 									1
Name	VIP IVR								
Description	VIP Customer IVR Application								
Active Add Custo	m IVR to Reported Applications								
VR Number	rs							⊼	
Delete									
R Number									
562								_	
									P
 IVR Applicat 	ion Versions							⊠	
Add New Delet	e Activate Import Export								
C Version	Created	Updated		Description	Active	Status			

2.2 IVR Editor

An IVR is made up of elements. Major elements (for example, *block*, *callout*, *customstate*, and *form*) must be given a unique name. To add an element, either click the *Add New* button or right-click on an existing element.



Sinch Contact Center IVR file can be stored, imported, and exported in VoiceXML format. Internally Sinch Contact Center compiles VoiceXML into the Python programming language and uses Python when the IVR application is running. Therefore, values and variables could be written as Python statements.

2.3 Adding Information to Elements

Elements are shown in the left frame and their parameters on the right. At least some of the fields must have a value. If a value is missing, a ^① sign is displayed. A message shows what is missing. The sign disappears when all the needed parameters have been entered.



 IVR Editor 						4	¢	⊠
Add New Up Dow	/n Delete			ÞĻ	ЪŤ	Save 🗙 Cancel 📃 Show Child Element Properties		
eustomstate: I	lissing or invalid attri	ibute value: Identifier			•	Missing or invalid attribute value: Identifier	•	•
Element	ID	Value	Target			Element Path custometate		
🕘 customst						Element Description customstate element is used to invoke a Python customizer method call and store	E	
						its result in a variable.		

2.4 Maximizing the Work Area

To widen the work area of the IVR Editor, move the divider between the SC menu and the working area as far to the left as possible. To increase the height of the working area, click on the *Increase Block Height* icon.

 IVR Editor 						(⁴)		
Add New U	Down Delet	te	kμ ⇒t		Save 🗙 Cancel 🗌	Show Child Element Properties	Block	1 Heiah
ldentifier	tate: Missing o	r invalid attribute	value: 🔹 🕨	1	Missing or invalid attril	oute value: Identifier	•	Ţ
Element	ID	Value	Target		Element Path	customstate	5	
▼ ♣ ● cus	to				Element Description	customstate element is used to invoke a Python customizer method call and store its result in a variable.	_	
					Identifier]	
					Method Name]	
				n	Parameter Values	Nina Makkonen (nina.makkonen@sinch.com) is signed in		
						V + - Up Down		
					Кеу	Value		
								F
							7	
					Attributes Specific to S	inch Contact Center		
			,	ŧ	Description			7

To expand the elements of the IVR Editor, click on the 🕨 icon.



3 VoiceXML Features

3.1 Supported Elements

For a list and detailed information of supported VoiceXML elements and extensions to the VoiceXML standard, see the *System Configurator* guide: *IVR Management > Using IVR Editor > Supported VoiceXML Elements, Element Attributes, Data Items and Queries, and Element Blocks.*

3.2 Commonly Used VoiceXML Elements

For more details on the elements presented here, see the *System Configurator* guide (*IVR Management > Using IVR Editor > Supported VoiceXML Elements*).

3.2.1 var

- The var element is used to store values during IVR execution.
- You should declare all your variables at the start of the vxml document.
 Variables declared here can be accessed from all subsequent elements.
- Using this element makes the IVR vxml document easier to read.
- To declare variables for a specific element only, declare it in that element, such as form.
- You can assign initial values to your variables or use the assign element later on.

Element	D	Value	Target
恭			
var	SAPGet_Meter_Details_wsdl	"http://dc2-sapr3-d01	
var	SAPValidate_Meter_Reads	"http://dc2-sapr3-d01	
var	SAPUpdate_Meter_Reads	"http://dc2-sapr3-d01	
var	Blocked_Account_Prompt		
var	Request_Not_Processed		
var	Return_Main_Menu_Prompt		
var	Speak_To_Customer_Rep		
var	Multiple_Meters_Prompt		
var	Z_IVRAfterHoursResidential	"9001"	
var	Z_IVRInHoursResidential	"9000"	
var	Z_Meter_Counter		
var	Z_Meter_Reading		
var	Z_Meter_Readings	[]	
var	Z_Get_Meter_Details		
var	Z_Meter_Types		
var	Z_QueueResidential		

• Values must be valid python data types.

3.2.2 form

- The form element is used as a container for other elements.
- The form name is used in goto elements.
- Commonly used child elements of a form are:
 - block
 - field
 - transfer
 - soap
 - customstate
- If you define var elements, they can be used within the form element only.

3.2.3 block

- The block element can be used as a container.
- The block name is used in goto elements.
- A block element is required for adding:
 - assign
 - audio
 - prompt
 - if

* form	Check_After_Hours		
- block	Change_Speak_To_Operat.		
> if		MyList.count('foo') > 0	
log		DBG> MyList.count('fo	
assign			Speak_T.
assign		("-")	Error_O

Element Path	form (Check_After_Hours) → block
Element Description	block element is a container element for executable content that executes if the condition of the block equals to true.
Element Name	Change_Speak_To_Operator_Prompt
Condition	-
Expression	-
BCM-Specific Attributes	
Description	

Element	D	Value	Target
var	Error_Options	("0", """)	
var	Confirm_Options	("0","1","2","*")	
var IVR_Info s		self.CALL.GetExtraDat	
var	Z_AccountNumber	"000100007287"	
var	Z_ListofMeters		
var	Z_CallingHour	"AFTERHOURS"	
var	Z_Info		
var	Z_NumberofMeters	1	
var	Z_TransferNumber		
var	Z_SAPError_AgentMessage	str(Z_AccountNumber)	
var	Z_TooComplex_AgentMes	str(Z_AccountNumber)	
var	MyList	['foo','bar']	
▶ form	Check_After_Hours		
▶ form	Get_Meter_Details		
▶ form	Check_Meter_Types		
▶ form	Enter_Meter_Readings		
▶ form	Update_Account		
I form	SAP_Error		
▶ form	Too_Complex		
▶ form	Account_Blocked		
▶ form	Back_To_Main_Menu		
▶ form	Transfer Call		



- log
- goto
- A block element can include a condition so that it is only executed if the condition is met (a simple if with no else if or else. In case elseif and else is required Conditional Block will add all elements at once).

3.2.4 field

- The field element is used to to collect information from callers.
- It can be used for simple menu types in addition to collecting longer strings of digits.
- field elements normally require the following child elements:
 - audio

Announces what is expected from the caller.

• filled

Executed when the input meets the minimum and maximum

▼ fie	eld	Meter_Reading_Value_1
•	filled	
•	noinput	

Element Path Element Description	form (Enter_Meter_Readings) \rightarrow field (1/3) field delement facilitates a dialog which allows the VR application to collect information from the user.
Name	Meter_Reading_Value_1
Expression	
BCM-Specific Attributes	
Description	
Minimum Number of Digits	1
Maximum Number of Digits	1
Repeats	
Timeout	5 s •



number of digits specified in the field element.

• noinput

Executed when the timeout specified in the field element is reached and there has been no input at all.

• nomatch

Executed when the timeout specified in the field element is reached and there has been some input but it does not meet the minimum or maximum number of digits. Not required when only 1 digit is expected (as in a simple menu).



3.2.5 transfer

- The transfer element is used to transfer the call to another number. This can be:
 - Another IVR
 - A queue
- A Sinch Contact Center
 internal extension
- Any valid external telephone number.
- Destination can be a variable – use Destination Expression.
- Call Attached Data is a string or variable populated in the previous step.
- For complex Call Attached Data strings, it is easier to create this using customstate.

3.2.6 soap

Note: In Sinch Contact Pro cloud environments, using SOAP requires making a service request to Sinch support.

Element Path	form (TranferToService) → transfer	
Element Description	transfer element transfers caller to another destination, such as a another IVR application.	queue, or
dentifier		
Condition		
Caller Number Expression		
Destination		
Destination Expression	ServiceQueueNumber	
Connect Timeout		s
Call Attached Data		a car
	• • • u	Jp Down
к	ley Value	
Call Attached Data Expression	n	
BCM-Specific Attributes		
Description		

- The soap element is used to fetch data from external data sources by using the SOAP protocol.
- It relies on a SOAP or web service defined in a target system such as SAP ERP or SAP CRM.
- In SAP ERP or SAP CRM, these webservices can be a standard ESoA service or a function module exposed as a web-service.
- An end-point needs to be created in SOAPMANAGER for web-services in SAP ERP or SAP CRM.

3.2.7 customstate

Note: Using this element is not supported in Sinch Contact Pro cloud environments.

Element Path form (Get_Meter_Details) → soap				
Element Description	soap element is used to services by using the S	o pull data from external data sources such as web OAP protocol.		
Name	SAPGet_Meter_Details	;		
Data Source Expression	SAPGet_Meter_Det	ails_wsdl		
Method Name	Bapizivrmrgetmeters			
	•	▼ ● Up Down		
	Kau	Mahua		
"Accountnumber"	Key	Value Z_AccountNumber		
"Accountnumber"	Key	Value Z_AccountNumber		
"Accountnumber"	Key	Value Z_AccountNumber 5 s		
Accountnumber*	Key	Value Z_AccountNumber		
"Accountnumber" Query Timeout Audio Expression	Key	Value Z_AccountNumber 5 s completion		

- The customstate element is used to execute a python customizer.
- If customized functions are needed to a custom-made IVR, they can be added to the system as Python code. Only one customizer can be active at a time for each custommade IVR.
- To call the customizer from the IVR application, use the customstate element.

Element Path form (Transfer_Call) - customstate Element Description customstate element is used to invoke a Python customizer method call and store its result in a variable. Identifier Custom_PrepareCAD Method Name PrepareCAD Parameter Values Image: Compare CAD Key Value "Z_AccountNumber" Z_CalingHour" Z_Info" Z_Info

•	▼ Customizers 조							
Add	Add New Delete Activate Deactivate Export							
R.	Created Modified Class Name Description Active							
	1	18-Jan-2012 23:51:06	18-Jan-2012 23:51:06	NR_Customize		\checkmark	ß	

- 3.2.7.1 When to use custom functions
- Custom Functions should only be used if you cannot achieve the same outcome using VXML elements.
- You should be familiar with Python before attempting to create your own custom functions.
- Custom Functions can be used for:
 - OBDC calls to external databases in Sinch Contact Center on-premise environments



- Doing several looping operations and condition tests to create a python list or dictionary
- SOAP queries if you wish to have the username and password as variables.
- A Python customizer provides a powerful tool to manipulate the Sinch Contact Center system, and therefore you should only use customizers you can trust. Never use customizers from an unknown source, and test all customizers in a test system before use.

Element Path

Element Description

Target Variable

Expression

3.2.8 assign

- The assign element is used to assign a value to a variable. It is recommended to create all variables first.
- Values must be valid python data types:
 - String
 - Number
 - Dictionary
 - List
- You can assign fixed values.
- You can update existing values:
 - Increase a counter
 value
 - Add a value to a list.

Element Path Element Description	<pre>form (Enter_Meter_Readings) → block (Implausible_Reading) → assign assign element is used to assign a value to a variable.</pre>
Target Variable	var: Z_Number_Attempts
Expression	Z_Number_Attempts + 1

assign

• 🔁 💿

var: Z_Number_Attempts

 $\textbf{form} \; (\texttt{Enter_Meter_Readings}) \rightarrow \textbf{block} \; (\texttt{Reset_Number_Attempts}) \rightarrow \textbf{bloc$

Ŧ

Ŧ

assign element is used to assign a value to a variable



3.2.9 audio

- The audio element is used to play audio files (also often referred to as prompts but not to be confused with the prompt element) in the IVR application.
- You can select predefined IVR type Prompt files from a dropdown menu. See how to configure customized IVR prompts in the "Getting Started" section.

Element Path	form (GetCustomerld) \rightarrow field (CollectedCustomerld) \rightarrow audio	
Element Description	audio element allows playing audio sound files in the IVR application.	
Condition		•
	Show All Prompt Types	
Prompt	Custom Welcome	•
Prompt Language		•
Audio File Expression		•
BCM-Specific Attributes		
Description		
Audio File Path	Default Prompt Path	•



3.2.10 goto

- The goto element is used to transfer application execution to a specific element in the current VXML document.
- You transfer to either of the following:
 - an element in the VXML root (normally a form element)
 - an item within a root element (normally a field or block element). Try to keep transfers using this method within the current form element so as to make the VXML more readable.

Element Path	form (Get_Meter_Details) → soap (SAPGet_Meter_Details) → error goto	-
Element Description	goto element is used to transfer application execution to a specific element in current document.	
Target Element	form: SAP_Error	•
Target item		•
Target Element Expression		,
Target item Expression		
Element Path Element Description	Korn (Enter_Meter_Readings) → field (Meter_Reading_Value_1) → filled → if → goto (1 / 2) goto element is used to transfer application execution to a specific element in current document.	
Target Element		•
Target item	field: Meter_Reading_Value_2	,
Target Element Expression		•
Target item Expression		



3.2.11 if, elseif & else

- The if element is used to specify conditional statements that allow choosing different options based on, for example, variable values.
- You need to specify at least one subsequent element (for example, assign, goto, audio, or prompt).
- elseif and else are regarded as child elements of the if element.
- Subsequent elements of elseif and else do not appear as child elements but act in that way.
- Try to design your IVR so that a condition is always matched (either if, elseif or else).

Note: The application evaluates the condition in if and elseif elements when executing child

Element	D	Value
▼ if		<pre>str(Z_BillType) == "A"</pre>
😰 audio		PROMPT: Actual_Bill_Type_Message_Prompt
elseif		<pre>str(Z_BillType) == "E"</pre>
😰 audio		PROMPT: Estimate_Bill_Type_Message_Prompt
else		
😥 audio		PROMPT: Interim_Bill_Type_Message_Prompt

Element Path	form (Update_Account) → block (Play_BilDate_Future) → if
Element Description	if element is used to specify conditional statements that allow choosing
	different options based on, for example, variable values.
	12
Condition	<pre>str(Z_BillType) == "\"</pre>
BCM-Specific Attributes	
Description	



elements in the if block. If you have multiple child elements in if or elseif elements, do not use conditions that can change during execution. For example, instead of using queue queries in a condition, use variables where you assign the value before an if block.



3.2.12 prompt

- The prompt element is used to build audio messages that include numerical information, such as cardinal and ordinal numbers, dates, and times.
- You need to specify the language used.
- You need to add at least one say-as child element.
- In the say-as element, you need to specify:
 - Data Type (Digits, Number, Ordinal, Date, Date and Time or Time)
 - Gender (Feminine or Masculine)
 - Declension

 (Nominative, Genitive only required for certain languages e.g. Finnish)
 - A value child element to the say-as element.
- Values must be valid Python data types or variables declared and assigned earlier.

Element	ID	Value	Target
▼ prompt		English (UK)	
▼ say-as		Digits	
value		<pre>str(Z_Meter_Readings[Z_Meter_Counter])</pre>	



3.3 Call Attached Data

- Call-attached data is nothing more than a Python dictionary (see chapter 6.3).
- You can add any information gathered in the IVR to this dictionary. For example:

```
{"Z_CADAlert": Caller selected Support - Account Inquiries,
"Z_AccountNumber": 123456}
```

 The OII Integrations interface of Sinch Contact Center converts this dictionary into an xml message using the Application ID specified in the OII setup (default is SAP_BCM/OII). For example:

```
<ItemAttachedData><Application
id="SAP_BCM/OII"><FirstBName>FE0CBDF0-FBB6-4FF3-8325-
C77322A15FAD </FirstBName><Z_CADAlert>Caller selected
Support - Account
Inquiries</Z_CADAlert><Z_AccountNumber>123456
</Z_AccountNumber><BNumberName>9216C832-FF83-4DF8-8A1A-
333CD34FAF76</BNumberName>
<FirstANumber>15136023488</FirstANumber><OrigQueue>9216C832-
FF83-4DF8-8A1A-333CD34FAF76
</OrigQueue></Application></ItemAttachedData>
```

- If you integrate Sinch Contact Center with SAP CRM, you may wish to send data to an additional Application ID: CRM_IC/BUPA.
 - Account identification in SAP CRM IC WebClient is done using a phone number or GUID of a Business Partner. This GUID has to be sent to the Application ID: CRM_IC/BUPA.
 - To automatically confirm a Business Partner, an 'X' needs to be sent using the key BPCONFIRMED in the same Application ID.
 - All other information sent using this Application ID can be overwritten by SAP CRM and therefore lost when the call is transferred.
- If you integrate Sinch Contact Center with SAP CRM or SAP ERP using SAPphone, you may wish to send data to an additional Application ID:



SAPphone.

Sinch Contact Center OII converts the XML data to send to SAPphone

In order to use different Application IDs, you need to create your own xml string.
 The customizer highlighted in chapter 3.2.7 does this for you:

```
<Application
```

```
id="CRM_IC/BUPA"><CURRENTCUSTOMER></CURRENTCUSTOMER><CURRENT
CONTACT>
</CURRENTCONTACT><BPCONFIRMED></BPCONFIRMED></Application>
<Application id="SAPphone"><Calldata obj="KEYVALUE"
inst="0001" key="Z_AccountNumber">123456</Calldata>
</Application>
```

 CAD passed to your IVR from another IVR is held in the dictionary "IVRInfo"

Flomont Dath	ver (10 / 28)		
Liement Path	val (19720)		
Element Description	var element is used to declare a VoiceXML variable within the scope specified by its parent element.		
Name	IVR_Info		
Initial Value	<pre>self.CALL.GetExtraData().get("IVRInfo")</pre>		

- To retrieve the value of Z_AccountNumber you can use:
- self.CALL.GetExtraData().get("IVRInfo",{}).get("Z_AccountNumber")



4 Best Practices for IVRs

Declare any information you might want to change or use later as a variable. These include, for example:

- Prompt files and collections of different prompt files
- Soap wsdl urls
- Valid keys to be selected in menus
- Counters
- Information to be sent as CAD.

Map a process into a form. A process can be, for example:

- Asking the caller for input and checking the result
- Calling a Soap query and then checking the result
- Handling errors and asking the caller for the next step
- Preparing the CAD and transferring the call.

Use fields for all caller input, as they:

- Can be contained in a form (a menu cannot)
- Are more flexible with checking for input and determining the next step.

4.1 Basic Functions (ANumber, BNumber, and so on)

- **DirectoryService-API** (To see how to use this function, see ANumber recognition IVR exercise from the IVR repository).
- **GetNumberInfo** method allows checking Agent status (see Appendix 2).
- GetCurrentPRSProfile method is used to check agent profile status.
- **AGENT.FIND** method can be used to search for an agent object based on address info or user GUID.
- **Example of modified** For *Example_IVR_Conditional.xml* and usage of *Queue.Query*, see appendices 3 and 4. For a sample file, contact support.
- **QUEUE.QUERY** API is documented in the System Configurator document: *IVR Management > Using IVR Editor > Element Attributes > Data Items and Queries.*



5 Troubleshooting

Note: Both the BLV log viewer application and log files are directly only available in Sinch Contact Center on-premise environments. Sinch Contact Pro cloud customers should contact Sinch support with a service request to get access to log files.

5.1 Business Communication Management Log Viewer (BLV)

Tip: Use only one Core while debugging an IVR application. This way all events can be collected from a single log:

CEM Server Instance 🛛 🗛	Active
ACME_Core	
ACME_Core2	

Sinch Contact Center on-premise software package includes a BLV log viewer application customized for Sinch Contact Center log files:

REV (L) - D:\SAP\BCM\Logs\MRP_QA_Core\CE	M_DC2-BCMAP-Q01_MRP_QA_Core_20120116_01.log
01:59:59 180 (14196/OueSt) DBG> Channel message to remote end 119 bytes: CMD=OueStats:TotPaperWork=0:Avail=1:TotOperators=2: SAP ID=CONTROL:TotC(
conn: AgentServer:MRP OA Agents	,
01-59-59 180 (14196/0neSt) DBC\ ChannelID hev=024DE80DD69E9845B724CEED113E71C4 accij=_M2222E28224\c2
01:59:59 180 (14196/0ueSt	DBC) Chamberly develop and PAGCAD
01-59-59 190 (14196/0ueSt	/ DBC/ Resauge generation/010600000 [10 215 66 26:0_010 215 66 24:210101; Sending percage: Channel=024df90dd69f9945b224ofe4112e71
01.59.59.100 (14196/Quest) RCC Connection/010000000 [10:215:00.70.0-710.215:00.74.21010]. Sending message: Channel-02401000017040160017240160113671
01.55.55.100 (14150/Quebu	DBS/ Connection/bioboboo [10.215.00.70.0->10.215.00.74.21016]. Sendasync obor5200 5 02460000 1 02460000 1 02460000 2 0
01:59:59.180 (14196/QueSt	DBG> Inreadrooi[0182D400]: Begin pending operation, pending=2
01:59:59.180 (14196/QueSt) DBG> Connection/0186D8D8 [10.215.66.76:0->10.215.66.74:21018]: Incremented references=3
01:59:59.180 (14196/QueSt) DBG> Connection/0186D8D8 [10.215.66.76:0->10.215.66.74:21018]: Begin pending operation, pending=2
01:59:59.183 (10116/IpcWorker) DBG> ThreadPool[0182D480]: WorkerThread GetQueuedCompletionStatus result=1 poverlapped=0E0F52C4 key=0186D8D8 bytes=139
01:59:59.183 (10116/IpcWorker) DBG> ThreadPool[0182D480]: IIoCompletion pending=2
01:59:59.183 (10116/IpcWorker) DBG> Connection/0186D8D8 [10.215.66.76:0->10.215.66.74:21018]: OnIoCompletion 0E0F52C4 result=1 bytes=139
01:59:59.183 (10116/IpcWorker) DBG> Connection/0186D8D8 [10.215.66.76:0->10.215.66.74:21018]: OnIoCompletion 0E0F52C0 send
01:59:59.183 (10116/IpcWorker) DBG> Connection/0186D8D8 [10.215.66.76:0->10.215.66.74:21018]: OnSendCount=[358584]
01:59:59.183 (10116/IpcWorker) DBG> Connection/0186D8D8 [10.215.66.76:0->10.215.66.74:21018]: Sending pending message completed
01:59:59 183 (10116/IncWorker) DBG> Peer/018346F0 [Cem:MRP OA Core]: Incremented references=13
01:59:59 183 (10116/IncWorker	DBG> Peer/018346F0 [Cem:MRP_04 Core]: Decremented references=12
01:59:59 193 (10116/IpoWorker	DBC Connection/01650808 [10 215 66 76:0-x10 215 66 74:210161]; Ended pending operation pendings1
01-59-59 192 (10116/TpoWorker	DBCs Connection / 01650-016 [10:215:66:76:0->10:215:66:74:21019]. Ended pending operation, pending-1
01.59.59.103 (10116/TpoUorker	DBC/ Commediation processing and a second se
01.59.59.103 (10116/10CWORKER	DBCS Intearrout of 204001. Ended pending operation, pending-1
01.50.50 104 (14106/0uest) DBC/ Securessage DBC/ Channels agents to prove and 124 better, bridlet, CAR ID-CONTROL TetCollegest TetBerryHerberg, 01000040E 211R 4201 D
01:59:59.164 (14196/Quest) DBS> Channel message to remote end 1/4 bytes: avail=1;_Sar_iD=CONIROL; lotCallers=1; lotraperwork=0; %aC%C40r-31AB-43DA-Br
, conn: AgentServer:MRP_UA_Agents	
01:59:59.184 (14196/QueSt) DBG> ChannelID hex=0249040C54AD484E8136ACC8FC1680AE asc11=7 I=7I7HN?6777777
01:59:59.184 (14196/QueSt) DBG> Message generated E46CC40
01:59:59.184 (14196/QueSt) TRC> Connection/0186D8D8 [10.215.66.76:0->10.215.66.74:21018]: Sending message: Channel=024904cc54ad484e8136acc8fc1680a
01:59:59.184 (14196/QueSt) DBG> Connection/0186D8D8 [10.215.66.76:0->10.215.66.74:21018]: SendAsync 0E0F52C0 5 0E46CC68 1 0E46CC69 1 0E46CC6A 2 0
01:59:59.184 (14196/QueSt) DBG> ThreadPool[0182D480]: Begin pending operation, pending=2
01:59:59.184 (14196/QueSt) DBG> Connection/0186D8D8 [10,215.66.76:0->10.215.66.74:21018]: Incremented references=3
01:59:59.184 (14196/ÖueSt) DBG> Connection/0186D8D8 [10.215.66.76:0->10.215.66.74:21018]: Begin pending operation, pending=2
01:59:59 186 (10116/IncVorker	DBG> ThreadPool[0182D480]; VorkerThread GetOuevedCownletionStatus result=1 noverlanged=0E0E52C4 key=0186D8D8 hytes=194
01:59:59 186 (10116/IncVorker) DBG> ThreadEcol[0182D480]: IIoCompletion mending=2
01.59.59 186 (10116/IncHorker) DBG) Connection/0186D8D8 [10 215 66 76:4-10 215 66 74:21018]: OnLoCompletion 0E0E52C4 recult=1 https://doi.org/10.1010/0101010101010101010101010101010
01:59:59 186 (10116/IpcWorker	DBC Connection (1186DBD8 [10:215.66.76:0-110.215.66.74:21018]; On LoCompletion (DEGE2C) cand
01:59:59 196 (10116/TpoWorker	DDC/ Connection/01060000 [10:215.66.76:0.10.215.66.74:21010]; OnCondCount=[250005]
01.59.59.100 (10110/1pcworker) DBC/ Commettion/010000000 [10:215:00.70.0-710.215:00.74.21010]. Oneendcount-[350505]
01.59.59.100 (10110/10CWOrker) DBCS Connection/orobobos [10.15.00.70.00/01/15.00.74.21016]. Sending pending message completed
01:59:59.186 (10116/1pcworker) DBG> Feer/0183ab70 [Cem: hkr_QA_Core]: Incremented references=13
UI:59:59.186 (10116/1pcworker) DBG> Peer/0183AbF0 [Cen: NKP_04_Core]: Decremented references=12
01:59:59.185 (10116/1pc@orker) DBC> Connection/U186D8D8 [10.215.66./6:0->10.215.66./4:21018]: Ended pending operation, pending=1
01:59:59.186 (10116/IpcWorker) DBG> Connection/0186D8D8 [10.215.66.76:0->10.215.66.74:21018]: Decremented references=2
01:59:59.186 (10116/IpcWorker) DBG> ThreadPool[0182D480]: Ended pending operation, pending=1
01:59:59.266 (05960/Message Sender) DBG> Send buffer is empty
01:59:59.570 (05912/Interface Clear	ner) TRC> Checking if any communication thread has stopped
01:59:59.570 (05912/Interface Clear	ner) DBG> [Message Receiver] thread is running : ThreadID = [9408]
01:59:59.570 (05912/Interface Clear	ner) DBG> [Message Sender] thread is running : ThreadID = [5960]
01:59:59.570 (05912/Interface Clear	ner) TRC> Running communication threads = [2]
01:59:59.571 (09052/IncWorker) DBG> ThreadPool[0182D560]: WorkerThread GetOueuedCompletionStatus result=1 poverlapped=0182A424 key=0186EC10 bytes=5
01:59:59.571 (09052/IncNorker) DBG: ThreadBool(0182D560): UnCompletion pending=1
01:59:59 571 (09052/TpcWorker) DBG: Connection/0186EC10 [10 215 66 76:0-10 215 66 73:21018]; OnToCompletion 01824424 result=1 bytes=5
01:59:59 571 (09052/IpcWorker	DBG: Connection/0186EC10 [10 215 66 76:0-10 215 66 73:21018]: On ToCompletion 0182420 Received
01.59.59.571 (09052/TpOWorker	DBC Connection is to be a construction of the provide statements of the construction o
01.59.59.571 (09052/10CWORKER) DBC/ TEE/ VIIGHOFO [CEM.INT_QM_COTE]. Inclemental Telefences*13
01.32.32.371 (02032/10CWORKER) DBS Connection products (10.215.00.76:0-)10.215.00.73:210101: Detected message of type (class Libioc:: DochandshakeMest
1 Just Furtherning 2 Just Furtherning 3 Just 9100 4	Si Shaving quanthing



You can view log data updated in real-time in **Live** mode or view the data available at the time you open the log in **Historical** mode. To toggle between the modes, type I (lower case) on your keyboard. Using the up arrow in the file also turns off **Live** mode. Note the (L) appended to BLV in the Windows title bar to indicate **Live** mode:

BLV (L) D:\SAP\BCM\Logs\MRP_QA_Core\C	E٢	1_DC2-6	BCMAP-Q01_MRP_QA_Core_20120116_01.log
01:59:59.180 (14196/QueSt)	DBG>	Channel message to remote end 119 b
, conn: AgentServer:MRP_QA_Agents			
01:59:59.180 (14196/QueSt)	DBG>	ChannelID hex=024DF80DD69F9845B724C
01:59:59.180 (14196/QueSt)	DBG>	Message generated E46CC40
01:59:59.180 (14196/QueSt)	TRC>	Connection/0186D8D8 [10.215.66.76:0
01:59:59.180 (14196/QueSt)	DBG>	Connection/0186D8D8 [10.215.66.76:0
01:59:59.180 (14196/QueSt	Ĵ,	DBG>	ThreadPool[0182D480]: Begin pending
01:59:59.180 (14196/QueSt)	DBG>	Connection/0186D8D8 [10.215.66.76:0
01:59:59.180 (14196/QueSt	j,	DBG>	Connection/0186D8D8 [10.215.66.76:0
01.59.59 183 (10116/TecMorker	Ń	DBC \	ThreadPool[0182D4801. NorkerThread

5.2 BLV Find dialog

Use the **Find** dialog to search the log and highlight matches to your rules in it. To open the dialog, press Ctrl + F on your keyboard.

Highlight / Collapse Rules	Exclude R	ules	
Rule Set Default	- Rule Set	Default	▼ + ·
AC IC RE 1 1 ERR>/EXC>/WRN>/error/fail	AC IC	RE	-
2 🗹 🗹 🗌 17:	2 -		•
3 🗆 🗖 🗖 debug	3 🗆		•
4 🗆 🗖 🗖 Relation	▼ 4 □ □		
5 🗆 🗆 🗖 IVR	▼ 5 □ □		
	• 6 6		
	7 🗆		
8 🗆 🗖 🗖 billing			
	• 9 F F		
10	■ 10 □ □		

You have the following options for finding matches in the log:

- To jump from one match to a pattern to the next, enter the pattern in the **Find pattern** field and click **Find**. This closes the dialog and moves you back to the main screen. You can quickly find the next or previous match with f or F keys.
- To highlight matches or only show lines that include them in the log:
 - 1. Define the necessary rules in the Highlight / collapse rules section.



2. Select how you want to be able to see rule matches in the log:

- To only highlight the matches with a color but show the whole log file, click the **AC** checkbox once on the rule line. This makes the check mark grey.
- In addition to highlighting, to be able to see only those lines in the log that include these matches, click the AC checkbox twice. This makes the check mark black.
- If you leave the checkbox unselected, the rule will not be used in highlighting.
- Click Tag all. This closes the dialog and tags all lines that match with the currently active Highlight / collapse rules. Tagging is a way of marking lines permanently. In the main screen, you can then jump between them easily (with t and T keys) or filter the view to show only tagged lines (C key).
- To filter out noise from a view, enter patterns for this into the **Exclude rules** section. To use the rule in the main screen, press the e key. This hides all lines that contain matches with the currently active Exclude rules.



5.3 Turning Collapsed/Tagged-only mode on and off

Collapsed mode only shows you the lines containing matches to your Highlight / collapse rules that have black check marks (clicked twice). To toggle between **Collapsed** and **Full** modes, press the C key on your keyboard. Note the (C) appended to BLV in the Windows title bar to indicate **Collapsed** mode:

🕺 BLV (C) - D:\SAP\BCM\Logs\MRP.	_QA_Core\CEM_DC2-BCMAP-Q01_MRP_QA_Core_20120116_02.log
02:05:12.585 (09564/IpcWorker) DBG> Connection/01861048 [10.215.66.76:0->10.215.66
02:12:48.833 (14196/QueSt) DBG> Connection/0186D8D8 [10.215.66.76:0->10.215.66
02:12:48.837 (14196/QueSt) DBG> Connection/0186D8D8 [10.215.66.76:0->10.215.66
02:26:47.891 (14196/QueSt) INF> ConfQUEUE_ScheduleDiff : 606425.8 <mark>9100</mark> 003242

Tagged-only mode doesn't hide any lines. Instead, it tags the lines that include matches to your Highlight / collapse rules and allows you to move from one tagged line to the next. **Tagged-only** mode considers rules with either grey or black check marks (clicked once or twice). To toggle between **Tagged-only** and **Full** modes, type C on your keyboard. Note the (T) appended to BLV in the Windows title bar to indicate **Tagged-only** mode.

👰 BLV (T) - D:\9	AP\BCM\Logs\MRP_QA	_Core\CE	M_DC2-BCMAP-Q01_MRP_(QA_Core_20120116_02.log
02:05:12.585	(09564/IpcWorker) DBG>	Connection/01861048	10.215.66.76:0->10.215.66
02:12:48.833	(14196/QueSt) DBG>	Connection/0186D8D8	10 215 66 76 0 310 215 66
02:12:48.837	(14196/QueSt) DBG>	Connection/0186D8D8	10.215.66.76:0->10.215.66
02:26:47.891	(14196/QueSt) INF>	ConfQUEUE_ScheduleDi	ff : 606425.8 <mark>9100</mark> 003242

Press Ctrl+Shift+T to clear all tags.



5.4 Changing the Font



To open the Font Dialog, type o on your keyboard.



5.5 F1 Help Commands

Alt+T = Ctrl+ <nbr> - Ctrl+Shift+<nbr> - Ctrl+Shift+ - Ctrl+F = Ctrl+S = Ctrl+S = Ctrl+Shift+T = Ctrl+Shift+T = Ctrl+WheelUp = Ctrl+WheelDown = c = f</nbr></nbr>	 Toggle tabs No selection = Toggle highlight <nbr></nbr> Selection active = Put selection to highlight <nbr></nbr> Selection active = Add selection to highlight <nbr></nbr> Copy selection to clipboard Open find dialog or find next occurrence of selected text Togle number highlighting Save selection, marked area, matched or tagged lines to file Tag or untag line Clear all tags Increase font size Collapse to or uncollapse from matching lines only view Collapse to or uncollapse from tagged lines only view Find next Find previous Cycle between normal, hex and hex-with-line-breaks views Turn live viewing on or off Goto next tagged line Start or cancel area marking Tag all lines within marked area Oncen fext actionality in the selection of the select
s =	- Show or hide status line
Down Up	= Scroll line down = Scroll line up
Home Ctrl+Home End Ctrl+End PageUp Ctrl+PageUp PageDown Ctrl+PageDown Ctrl+PageDown Right Ctrl+Right Left Ctrl+Left Esc Ctrl+Del F1 F3 F12 Shift+F12 Left click	 Show start of line Goto start of file Show end of line Goto end of file Scroll page up Change to previous file Scroll page down Change to next file Scroll column right Scroll column left Scroll page left Cancel selection and instant highlight Clear file (close, truncate and reopen) Show this help Open find dialog Toggle allways-on-top mode Cycle through different window transparency levels
Ctrl + Left click Left Doubleclick Right Doubleclick	 Start stream selection Start box selection Select and instantly highlight word or GUID or text in parens: () [] {} <> word = digits letters underscore GUID = 32 hex digits maybe separated with dashes Tag or untag line



5.6.1 Setting the CEM log levels to debug

Make sure the CEM log level is high enough to show the IVR detail. In a production environment, do not leave the log levels as Debug after debugging, as the log file size will grow exponentially. Other options are: Warning, Info, and Trace. Warning is usually the default value.

Open Registry Editor on the server and check the following values. Create any missing values:

- Debug = 7
- DebugSecure = 0
- LogLevel = debug
- LogModule.Cem.Level = *inherited*





5.6.2 Locating the CEM log

CEM logs are usually located in folder: *drive*:\Sinch\ContactCenter\Logs*InstallationName*_Core

Log filename contains the following parts: CEM_MachineName_InstallationName_Core_Date_Hour

Use the latest CEM log for debugging.

Note: Log files use UTC Date and Time regardless of your server time zone settings. **Tip**: Sort the files using the Date Modified column in descending order to find the latest file.

Name	Date modified +	Туре	Size	
CEM_DC2-BCMAP-Q01_MRP_QA_Core_20120116_22.log	1/17/2012 11:11 AM	LOG File	6,603 KB	
CEM_DC2-BCMAP-Q01_MRP_QA_Core_20120116_21.log	1/17/2012 11:00 AM	LOG File	38,465 KB	
CEM_DC2-BCMAP-Q01_MRP_QA_Core_20120116_20.log	1/17/2012 10:00 AM	LOG File	41,804 KB	
CEM_DC2-BCMAP-Q01_MRP_QA_Core_20120116_19.log	1/17/2012 9:00 AM	LOG File	35,510 KB	
CEM_DC2-BCMAP-Q01_MRP_QA_Core_20120116_18.log	1/17/2012 8:00 AM	LOG File	35,282 KB	
CEM_DC2-BCMAP-Q01_MRP_QA_Core_20120116_17.log	1/17/2012 7:00 AM	LOG File	32,506 KB	
CEM DOD DOMAD OOL MODI ON CHILL DOLDOLLO IN LA	121720010 C-00 AM	Log pla	01 702 VD	

5.6.3 Reading the CEM log

5.6.3.1 Finding log entries related to your IVR

- 1. Make a call to your IVR and search for the number you called (or called from).
- Search for the entry 'CustomIvr_' after this time.
 Your IVR has a unique GUID (in this case CustomIvr_7E4C9738C56B4A7A8FB307C88DC45ABA).



3. Add this to your search terms in BLV.

R BLV - D:\SAP\BCM\Logs\MRP_QA_Core\CEM_DC2-BCMAP-Q01_MRP_QA_Core_20120116_22.log								
BLV - Find	BLY - Find X							
Find pattern		From start or end of file						
Higlight / co	ollapse rules	Exclude rules						
Ruleset	_	Ruleset	-					
Hil Ign	ore case	Sel Ignore case						
1 🔽 🗆	9001		•					
2 🔽 🗆	Customlvr_7E4C9738C56B4A7A8FB307C88	2 🗆 🗖	•					
3 🗆 🗖		3 🗆 🗖	•					
4 🗆 🗆			•					
5 🗆 🗆	▼		•					
<u>6</u> []			•					
ΖΠΠ			•					
8 🗆 🗖			•					
9 🗆 🗆	•	9 🗆 🗆	•					
10 🗆 🗖	V		•					
	<u><u>F</u>ind <u>C</u>lose</u>	Iag all Cancel						

4. Click Find.

BLV log viewer shows matching entries:




5.6.3.2 Adding log entries into your IVR

There's no breakpoint feature in the IVR. By adding log entries to your IVR, you can find this entry in the CEM log and use it as a reference point. Unique log entries can be used to pin-point each stage. See more about the log element in the *System Configurator* document: *IVR Management > Using IVR Editor > Supported VoiceXML Elements > Log Element*.

• • = = ▼ IVR Editor m Delete ✓ Apply | X Cancel | Show Child Element Properties Element D Z_QueueDisconn Element Path form (Check_Account_Number) → block (1 / 2) → log var var Z QueueResidential Element Description log element allows writing messages to CEM log. Z_IVRAccountBalance Expression "Hello World" var Z_IVRCopyLastInvoice Z_IVRCreditCard Level DBG Z IVRMeterReading BCM-Specific Attribute Z_IVRNextInvoiceDate Z_IVRNextMeterReading. Description IVR Info self.CALL.GetExtraData().g. Z_AccountNumber Z_CallingHour str("AFTERHOURS") Z_info Z TransferNumbe Check_Account_Numbe - blo DBG> "Hello World" Block_Check_Account_N...

Enter your log entry text to the Expression field:

Search results in Core log:

DELY (C) - D:\SAP\BCM\Logs\MRP_QA_Core\CEM_DC2-BCMAP-Q01_MRP_QA_Core	_20120116_22.log	
22.2.2.2.2.5.6. (a.G. (b.G. (b	IMP_ Drvpplication_processform (CALIO=COEEIIOTAF480C2EDADT7DAT708): IMP_ Drvpplication_processform (CALIO=COEEIIOTAF480C2EDADT7DAT708): Start processing for IMP_ STAI_condition_Block (200) IMP_ Drvpplication_conditionaBlock (CAL_IO=COEEIIOTAF480C2EDADT7DAT708): STAT_PAMME(I) IMP_ Drvpplication_conditionBlock (CAL_IO=COEEIIADTAF480C2EDADT7DAT708): STAT_PAMME(I) IMP_ Drvpplication_conditionBlock (C	<pre>m : 'Check_Acco' : statementsSta : nextState = < 1 elementIndex i elementName = i nextState = < i formName = Ch k(0] = ('block', 'owarldds2')</pre>
	Inv: TreepIletion.exiluateOpression (CAL_ID=COEDIAO74746(COEDIAE7DAA706)) : "tegression propoletion, constituted in Coediae (CAL_ID=COEDIAO74746(COEDIAE7DAA7066)) : tetace completed into [STA] procession (CAL_ID=COEDIAO74746(COEDIAE7DAA7066)) : tetace completed into [STA] procession (CAL_ID=COEDIAO74746(COEDIAE7DAA7066)) : tetace into [STA] procession (CAL_ID=COEDIAO744746(COEDIAE7DAA7066)) : tetace into [STA] procession (CAL_ID=COEDIAO744746(COEDIAE7DAA7066)) : tetace into [STA] procession (CAL_ID=COEDIAO744746(COEDIAE7DAA7066)) : tract_process[]] : teta into [STA] procession (CAL_ID=COEDIAO744746(COEDIAE7DAA7066)) : tract_process[]] : teta into [STA] procession (CAL_ID=COEDIAO744746(COEDIAE7DAA706)) : tract_process[]] : teta into [STA] procession (CAL_ID=COEDIAO744746(COEDIAE7DAA706)) : tract_process[]] : teta into [STA] procession (CAL_ID=COEDIAO744746(COEDIAE7DAA706)) : tract_process[]] : teta into [STA] procession (CAL_ID=COEDIAO744746(COEDIAE7DAA7066)) : tract_procession (CAL_ID=COEDIA070474766) : tract_procession (CAL_ID=COEDIA0704766) : tract_procession (CAL_ID=COEDI	': ''HillorAvelle smentIndex = 1 smentName = None vtState = dound mName = Check_A ementIndex = 13 riables = ('Z_Qu ementName = None

You can also add a log entry to show the current value of a variable, such as

Z_CallingHour. Enter the name of the variable to the **Expression** field.

To find the variable entry in the log more easily, you can add unique text, such as *Calling hour is*: "Calling Hour is : " + str(Z_CallingHour)



 IVR Editor 							+ - 五
Add New Up	Down Delete		ÞĻ.	эŤ		🖌 Apply 🗙 Cancel 🗌 Show	Child Element Properties
Element	D	Value	Target			Flement Dath	form (Check After Hours) - block (Change Speak To Operator Promot) - log
var	Z_Meter_Reading						Torm (check_And_hours) > block (change_open_to_open.tor_houp) > log
var	Z_Meter_Readings	[]			н	Element Description	log element allows writing messages to CEM log.
var	Z_Get_Meter_Details				н		
var	Z_Meter_Types				н	Expression	"Calling Hour is : " + str(Z_CallingHour) 💌
var	Z_QueueResidential				н	Level	DBG
var	Error_Options	("0", "*")			н		
var	Confirm_Options	("0","1","2","*")				BCM-Specific Attributes	
var	IVR_Info	<pre>self.CALL.GetExtraData()</pre>				Description	
var	Z_AccountNumber	"000100007287"			1		
var	Z_ListofMeters						
var	Z_CallingHour	"AFTERHOURS"					
var	Z_Info						
var	Z_NumberofMeters	1					
var	Z_TransferNumber						
var	Z_SAPError_AgentMessage	<pre>str(Z_AccountNumber) + ",</pre>			1		
var	Z_TooComplex_AgentMessage	<pre>str(Z_AccountNumber) + ",</pre>			н		
▼ form	Check_After_Hours				н		
* block	Change_Speak_To_Operator_Prompt						
log		DBG> "Calling Hour is : "					
				•			

Search results:





6 Python

Note: This chapter is not relevant for Sinch Contact Pro cloud customers.

6.1 Manipulating Strings

Python term	Description	Example	Values in Example	Result
str	String	str(a) + str(b)	a = "Hello ", b = "World"	Hello World
lstrip()	Remove a given character from the left	a.lstrip("0")	a ="000001"	1
split	Split a string at a given character. Produces a List	str(a).split(".")	a = "1.00"	["1", "00"]
replace	Replace a given character in string	a.replace("o", "")	a = "Hello World"	Hell Wrld
isdigit()	Checks if a string is comprised of just digits	a.isdigit() b.Isdigit()	a = "Hello World" b = "12345"	False True

These functions can be combined: a.replace("#", "").isdigit() returns

True when a = 12345#

Python term	Description	Example	Values in Example	Result
len	Length	len(a)	a = "Hello World"	11
	Character at position b	a[b]	a = "Hello World", b = 1	е
	Characters from b to c	a[b:c]	a = "Hello World", b = 1, c=3	el
	Characters up to b	a[:b]	a = "Hello World", b = 3	Hel
	All but characters up to b	a[b:]	a = "Hello World", b = 3	lo World
	Last character	a[-1]	a = "Hello World"	d
	Last but one character	a[-2]	a = "Hello World"	I
	Last b characters	a[-b:]	a = "Hello World", b = 3	rld



All but the last b characters	a[:-b]	a = "Hello World", b = 3	Hello Wo
-------------------------------	--------	--------------------------	----------



6.2 Some other Python expressions used in IVR Editor

Python term	Description	Example	Values in Example	Result
int	Integer	int(a)	a = 1.03	1
datetime.datetime. now()	Returns current date and time in datetime format	datetime.datetime. now()		2012-01-18 10:24:08.259000
datetime.datetime. strptime	Formats a string using a given format in datetime format	datetime.datetime. strptime(a, "%Y- %m-%d")	a = "2012-04-22"	2012-04-22 00:00:00

You can use datetime to check if a date is in the past: datetime.datetime.strptime(a, "%Y-%m-%d") < datetime.datetime.now() For more python examples, see:

- http://docs.python.org/tutorial
- Google search: python term

6.3 Dictionaries, Lists and Tuples

Dictionaries {}

- A dictionary is an unordered set of key: value pairs, with the requirement that the keys are unique (within one dictionary).
- Example: {'First':1,'Second':2,'Third':3,'Fourth':4}

Lists []

- Lists are a list of values, each one numbered, starting from zero. The first one is numbered 0, the second 1, the third 2, etc. You can remove values from the list and add new values to the end.
- Example: ['First', 'Second', 'Third', 'Fourth']

Tuples ()



- Tuples are just like lists, but you can't change their values. Again, each value is numbered starting from zero, for easy reference.
- Example:

('Sunday', Monday,', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday')



A simple dictionary:

{'Multi': None, 'Meters': None, 'Blocked': 'X'}

A dictionary inside a dictionary:

{'Return': {'Type': 'E', 'Number': '016', 'Id': 'ZIVRMES'}, 'Multi': None, 'Meters': None, 'Blocked': 'X'}

A dictionary inside a tuple:

('item', {'Division': '51', 'Installation': '3000194370', 'Register': '001', 'Equipment': '00000000040099113', 'Oldreading': '29165', 'Newreading': None, 'Meternumber': '00000000008000104'})

A dictionary inside a tuple, inside a list, inside a dictionary:

{'Meters': [('item', {'Division': '51', 'Installation': '3000194370', 'Register': '001', 'Equipment': '00000000040099113', 'Oldreading': '29165', 'Newreading': None, 'Meternumber': '00000000008000104'}), ('item', {'Division': '52', 'Installation': '3000417096', 'Register': '001', 'Equipment': '00000000010117702', 'Oldreading': '5846', 'Newreading': None, 'Meternumber': '00000080823161079'})], 'Blocked': None}

6.5 Accessing Data in Dictionaries

Elements of a dictionary can be accessed by key in three different ways:

- mydict["somekey"] Throws a KeyError exception if mydict doesn't contain somekey
- mydict.get("somekey") Returns None if mydict doesn't contain somekey



• mydict.get("somekey",1) Returns a default value (1 in this case) if mydict doesn't contain *somekey*

To prevent a KeyError exception being thrown in the IVR, best practice is to use mydict.get("somekey"). For example, if the dictionary is: mydict = {'Multi': None,

'Meters': None, 'Blocked': 'X'}

mydict["Blocked"]	Returns X
mydict.get("Blocked")	Returns X
mydict["Single"]	Throws a KeyError exception
mydict.get("Single")	Returns None

You can use a variable as the key: myvar = "Blocked"

mydict.get(myvar) Returns X



6.6 Accessing Data in Dictionaries - Complex Example

Elements of a dictionary within a dictionary can be accessed in the following ways:

<pre>mydict = {'Return': {'Type': 'E',</pre>	'Number': '016',	'Id': 'ZIVRMES'},
'Multi': None, 'Blocked': 'X'}		

mydict["Return"]["Type"]	Returns E
mydict.get("Return",{}).get("Type")	Returns E
mydict["Return"]["Letter"]	Throws a KeyError exception
mydict.get("Return",{}).get("Letter")	Returns None
mydict["Result"]["Letter"]	Throws a KeyError exception
mydict.get("Result",{}).get("Letter")	Returns None

Again, in order to prevent a KeyError exception being thrown in the IVR, best practice is to use mydict.get("somekey1",{}).get("somekey2")

The first .get *has* to return a dictionary object in order to avoid a KeyError, so it returns a new empty one (default value) in case there is no *Result* element available in mydict

6.7 Accessing Data in Lists and Tuples

Elements of a list or tuple can only be accessed using a zero-based index. There is no .get method available for lists and tuples.

1	nylist = ['foo','bar']
mylist[0]	Returns 'foo'
mylist[1]	Returns 'bar'
mylist[-1]	Returns the last element in the list; 'bar' in this case
mylist[2]	Throws an IndexError exception (list index out of range)

Use mylist.count("somevalue") to test if *somevalue* exists. 0 is returned if it doesn't exist.

```
mylist.count("foo") Returns 1
```



mylist.count("food")	Returns 0
mylist[mylist.index('foo')]	Returns 'foo' but can be used only if 'foo' exists in
	the list or tuple
len(mylist)	Returns 2 – the number of items in the list

You can append items to a list. For example, mylist + ['gaa'] results in mylist = ['foo','bar','gaa']



7 Python Customizer Example

Note: For Sinch Contact Pro cloud, Sinch needs to verify that any customizer you want to add won't cause security or performance issues. Create a service request via the support portal for the customizer review. This is billable work and based on actual review hours.

Python customizers can be used for various purposes with Sinch Contact Center IVR application. This sample describes how to make a database query by using Python customizer. For further details about IVR structure and the sample Python customizer file, contact support. **NOTE:** Custom directory database table is called *"VipId"* in this customizer sample.

Prerequisites:

- Create new table *CustomerID* to the Sinch Contact Center directory database.
- Modify Python customizer to access this table by using Windows authentication.

The customstate element is used to invoke a Python customizer method call and store its results in a variable. For more information about customstate configurations, see the System Configurator (SC) document.

The required values are:

(form) Identifier = FindCustomerFromDatabase (Parent element for customstate child element)

This is an identifier (name) for parent element. The identifier appears in the ID column, and it is used as an address when the call is transferred to this element.

• (customstate) Identifier = CustomerGUID

This is an identifier (name) for child element. The identifier appears in the dropdown list of expressions with the prefix customstate:, the results of the customized method are called with that expression.



• (customstate) Method Name = getCustomerById

This specifies the python customer method to call.

- (customstate) Parameter Values (Key and Value pairs) = 'CustomerID' and CollectedCustomerId
 - *'Customerld'* is the name of the custom method parameter. The value of this parameter contains the customer id that is used to search for a customer form the database.
 - *CollectedCustomerld* is an identifier of the field element where the given customer ID DTMF digits are collected.
- (customstate) Parameter Values (Key and Value pairs) = 'DirectoryField' and 'GUID'
 - 'DirectoryField' is the name of the custom variable for connecting the custom directory field to the appropriate GUID for search purposes in Python customizer.
 - GUID is the GUID of the custom directory field in Sinch Contact Center directory database.

▼ Directory Mapac	rement							
Directory Ter	mplates P Add 1	lew 🛅 Cop	oy 🍵 Delete 🍫 Ref	resh				
Directory Fie	lds G		ID		Name =	Descriptio	n	
Directory Gr	oups 8194	7343 CD85	430F-9DE3-D40D07DE3 46D7-A568-4A284359	AB96	Cost Center			
Presence Mana	gement 2E26	FA64-8536-	468E-A09A-FE15927EE	8806	CustomerID	CustomerID		
▼ Oueue Menerer	nent F55E	F596-F03E-	432F-6496-66F2F8550	87E	Department			
▼ IVR Editor							\$\$ \$P\$	
Add New Up Dow	n Delete		HL HT	📕 Save	X Cancel C Sho	w Child Element Prop	erties	
Element	ID	Value	Target	Element P	atto	form (FindCusto		
ቆ			Element Pa					
var	VIPQueueNumber	'1011'		Element Description cus		customstate el call and store its	customstate element is used to invoke a Python customizer metho call and store its result in a variable.	
var	ServiceQueueNumber	'1012'						
▶ form	GetCustomerid			Identifier	Identifier CustomerGUID		omerGUID ustomerByld	
∗ form	FindCustomerFromDatabase							
▼ customstate	CustomerGUID		getCustomerByld	Method Name getCust		getCustomerByl		
▼ error				Parame	ster Values			
🔿 goto			→ TranferToService					
block							👻 🙂 Up Down	
> form	TranferToService				Key		Value	
	TransferToVip			'Custo	merID'		CollectedCustomerid	
form				Direct	and the last		IDEDEEACA DEDE ACOE ACOA EE1COD7EDOCCI	



7.1 DB Query Customizer Sample

DB Query Customizer provides a good example with comments on how to connect to internal or external databases by using Sinch Contact Center's own ODBC API and use the results. For the customizer, contact support.



8 SoapUI

Note: In Sinch Contact Pro cloud environments, using SOAP requires making a service request to Sinch support.

SoapUI can be used to test your web services and to show the structure of the input and output messages. By testing your web services first in SoapUI, you will eliminate errors resulting from the Soap query first, which will make it much easier to use your queries inside your IVR. By installing and running SoapUI from the server(s) where the Sinch Contact Center Core VU is installed, you can first test if there are any firewall issues. SoapUI is freeware and is available from: <u>http://www.soapui.org/</u>

8.1 Test Your Web Services

- 🗆 🗙 Eile Tools Desktop Help 🕒 🕲 🗟 🧔 🤌 💽 💥 📕 🚺 📓 Search Forum # 🕡 soapUI Starter Pag ď 🖉 🗵 Navigator 00 Projects Hi there! You're getting this message since your computer is offline and soapUI can't access Internet. Without Internet connection you might experience some issues when doing Web Service Testing. les the latest news and other 🖻 New soapUI Project New soapUI Project <u></u> tion from SmartBear Creates a new soapUI Project in this workspace Account Balance Project Name: Initial WSDL/WADL: dl11/allinone/ws_policy/document?sap-client=236 Browse... Create Requests: Create sample requests for all operations Create TestSuite: Creates a TestSuite for the imported WSDL or WADL Create MockService: Creates a Web Service Simulation of the imported WSDL Add REST Service: Relative Paths: Stores all file paths in project relatively to project file (requires save) Create Web TestCase: Creates a TestCase with a Web Recording session for functional web testing OK Cancel Property Value Name Projects Description File C:\Documents and Se Project Root Properties soapUI log http log jetty log error log wsrm log memory log
- 1. Create a new project and insert your wsdl url.



2. Enter the user and password that will be used with the web services (this assumes the web service is set up to use Basic Authentication).

SoapUI 4.0.1		
Elie Tools Desktob Helb		Search Forum 🛛 📸 🕖
b E Projects	soapUI Starter Page	ь ^к Ф. Х
Projects Account Balance	Hithere! You're getting this message since your computer is offline and Internet connection you might experience some issues when Usually this is where you would see the soapUI starter page with Basic Authentication Basic Authentication Specify Basic Authentication required for [dc2-sapr3-d01.mrp.net.nz:8001] Username: SAPBCM Password: ••••••••••••••••••••••••••••••••••••	d soapUl can't access Internet. Without doing Web Service Testing. which includes the latest news and other nformation from SmartBear.
Property Value		
Name Account Balance		
Properties	soapUI log http log jetty log error log wsrm log memory log	

3. Remove the SOAP1.2 interface, as it is not required.

🖙 soapUl 4.0.1	
<u>Eile T</u> ools <u>D</u> esktop <u>H</u> elp	
🛡 🕲 🖻 🔌 📀 🐇 🖡 🕅 📓	Search Forum 🛛 🏙 🕖
Projects Account Balance Account Balance ZBAPIZIVRGETACCOUNTBALANCE Baptingetaccountbalance Account Balance Control Control Control Control Report Rep	SoopUI Starter Page HI there! You're getting this message since your computer is offline and soapUI can't access Internet. Without ternet connection you might experience some issues when doing Web Service Testing. Sually this is where you would see the soapUI starter page which includes the latest news and othe slevant information. ake sure to connect to Internet to get access to all services and information from SmartBear. hanks1 he SmartBear Sweden Team the project
Properties	soapUI log http log jetty log error log wsrm log memory log



4. Open Request 1 and enter the user and password that will be used with the web services (this assumes the web service is set up to use Basic Authentication). This is your query structure. We are only interested in the information in the Body.

soapUI 4.0.1			
e <u>T</u> ools <u>D</u> esktop <u>H</u> el	5		
ଷ 🖺 🔞 🍰 🤄) 🕺 🖡 🕐 🐚	Search Forum	
=		So Request 1	약 다. 조
Projects		🕨 늘 💱 🖸 🗖 🛱 🔳 http://dc2-sapr3-d01.mrp.net.nz:8001/sap/bc/srt/rfc	/sap/zbapizivrgetaccountbalance/236/
Account Balance Account Balance Account Balance Baptivrge Sametric Baptivrge Sametric Baptivrge Sametric Sametric	ACCOUNTBALANCE accountbalance at 1 Value Request 1 341 UTF-8 http://dc2-sapr3-d01.mr true	<pre>scapenv: Envelope min: scapenv=http://sch <scapenv:ender></scapenv:ender> <scapenv:ender></scapenv:ender> <scapenv:endy> <ure:ender></ure:ender> </scapenv:endy> </pre>	
SSL Keystore Skip SOAP Action	false		
Enable MTOM Force MTOM	false		
Inline Response Attachm.	, false		
Expand MTOM Attachment	is false		
Disable multiparts	true		1
Encode Attachments	false	Head Attachm W JMS H JMS Prop Header Attac	hment SSL WSS JMS
Enable Inline Files	false		
Strip whitespaces	false 💌		

5. Fill in the parameter and submit the request.

🖻 soapUl 4.0.1				
<u>File T</u> ools <u>D</u> esktop <u>H</u> elp	5			
🖣 🕲 🗟 🕡 🦽 🧟) 💥 🚚 🕐 🌆		Search Forum	# 0
5 =		St Request 1		
🖥 🖭 Projects			t pay 8001 / cap / bc/ext/rfc/cap / thaniti most accou	inthalance/226/
Account Balance			 Interference 	ancouldine of 2007
E Z ZBAPIZIVRGET	ACCOUNTBALANCE	Submit request to specified endpoint URL anv="http://sch	> W	-
E Sapizivrget	accountbalance	<soapenv:header></soapenv:header>	2	
AP Reques	SC I	α <ur> <ur> <ur> <ur></ur></ur></ur></ur>	Ra	
		<accountnumber>000101955090<td></td><td></td></accountnumber>		
Request Properties				
Property	Value			
Name	Request 1			
Description				
Message Size	352			
Encoding	UTF-8			
Endpoint	http://dc2-sapr3-d01.mr			
limeout				
Bind Address	huar			
Honow Redirects	SADRCM			
Raceword	*******			
Domain				
WSS-Password Type				
WSS TimeToLive				
SSL Keystore				
Skip SOAP Action	false			
Enable MTOM	false			
Force MTOM	false			
Inline Response Attachm.	, taise	▼		
Expand MI OM Attachment	is raise	 ✓ 	1	•
Encode Attachments	false			
Enable Inline Files	false	Head Attachm W JMS H JMS Pro	Header Attachment SSL V	vss JMS
Strip whitespaces	false 🔻			5:37
Properties		soapUI log http log jetty log error log wsrm log memory log		



Now you should have your return structure. Again, we are only interested in the information contained in the Body. Running this query in the IVR returns the same structure but in a Python format using dictionaries, lists, and tuples.



8.2 Using Soap Elements inside Sinch Contact Center IVR

The example used in this chapter:

<urn:Bapizivrgetaccountbalance>

<Accountnumber>123456</Accountnumber>

</urn:Bapizivrgetaccountbalance>



▼ IVR Edito	or					
Add New Up Down Delete						
🔔 One or	more ma	andatory child e	lements mis	sing: (MENU, FO	RM]	
Eleme	nt	ID				Value
ሔ 🛆						
var		Z_AccountBala	ance_wsdl	"http://mys	apsystem.cor	p/myaccountbalancebapi.wsdl"
var			her	"123456"		
	Add N	lew Sibling 🕨	Child Element		customstate	
	🗊 De	elete	Child Ele	ement Block 🕨	soap	
	Move	Up			form	
					menu	
					noinput	
					nomatch	
					var	



- 1. Give your Soap element a unique name.
- 2. Enter the url of your wsdl file between quotes (that is, as a string) or use a variable defined earlier:
 - "http://mysapsystem.corp/myaccountbalancebapi.wsdl"
 - str(Z_AccountBalance_wsdl)
- 3. The method name is the name specified in the urn:

<urn:Bapizivrgetaccountbalance>

<Accountnumber>123456</Accountnumber>

</urn:Bapizivrgetaccountbalance>

Element Path	soap
Element Description	soap element is used to pull data from external data sources such as web services by using the SOAP protocol.
Name	SAP_GetAccountBalance
Data Source Expression	<pre>str(Z_AccountBalance_wsdl)</pre>
Method Name	Bapizivrgetaccountbalance

- Add the input parameters your soap service is expecting: xml child nodes of the urn node.
 - The key is the name of the node.
 - The key needs to be within quotes to stop it being evaluated as a Python variable.
 "Accountnumber"
 - The value can be a string or a variable.
- Click to add your entry.

Element Description	soap element is used services by using the	soap element is used to pull data from external data sources such as web services by using the SOAP protocol.		
lame	SAP_GetAccountBal	ance		
Data Source Expression	Z_AccountBalanc	e_wsdl		
Nethod Name	Bapizivrgetaccountba	alance		
Parameter Values				
"Accountnumber"	Z Accou	untNumber 🔽 💽 🕒 Up Down		
K	еу	Value		



- Save your changes so you don't lose your entry.
- 7. Change the Query Timeout period if required
- 8. You can assign a variable containing a prompt to be played during execution
- 9. Enter the credentials as required.

This depends how your Soap end-point was set up in the target SAP CRM or SAP ERP system in soapmanager.

Query Timeout	5 s v
Audio Expression	
	Interrupt Audio Upon Completion
	Repeat Audio
User Name	
Password	******************
Certificate Name	
Certificate Issuer	
Certificate Store	

10. Don't forget to add error and timeout handlers. The system executes the timeout element when the Query Timeout value defined earlier is reached.

▼ soap	SAP_GetAccountBalance	
🔔 error		
🖄 timeout		



8.3 Adding a log entry in your IVR to return a value from a Soap or custom state element

▼ IVR Editor						• - 조
Add New Up Do	wn Delete		÷4	ъŤ	🖌 Apply 🗙 Cancel 🗌 Show Chi	Id Element Properties
Element	D	Value	Target		Element Path for	rm (Get Meter Details) → block (Check SAP Meter Details Result) → log
var	Z_QueueResidential			٠	Element Description	
var	Error_Options	("0","*")			Element Description 10g	g element allows writing messages to CEM log.
var	Confirm_Options	("0","1","2","*")			Everancian	
var	IVR_Info	<pre>self.CALL.GetExtraData().get</pre>			Expression	Soap element SAPGet_Meter_Details returned " + str(SAPGet_Meter_Details) 💌
var	Z_AccountNumber	"000100007287"			Level	BG 💌
var	Z_ListofMeters				BCM-Specific Attributes	
var	Z_CallingHour	"AFTERHOURS"			Dom opcono randato	
var	Z_Info				Description	
var	Z_NumberofMeters	1				
var	Z_TransferNumber					
var	Z_SAPError_AgentMessage	<pre>str(Z_AccountNumber) + ", SA</pre>				
var	Z_TooComplex_AgentMessage	<pre>str(Z_AccountNumber) + ", To</pre>				
▶ form	Check_After_Hours					
▼ form	Get_Meter_Details					
▼ soap	SAPGet_Meter_Details					
error						
timeout						
* block	Check_SAP_Meter_Details_Result					
log		DBG> "Soap element SAPGet_Me		Ŧ		

To add a log entry to show the return value from a soap or custom state, add the name of the element in the Expression:

Expression = SAPGet_Meter_Details

By adding a unique text to the log entry, you can find this entry in the CEM log more easily:

Expression = " Soap element SAPGet_Meter_Details returned
" + str(SAPGet_Meter_Details)

You can then work out how to parse the result.





Search Result - return value from a soap or custom state element

8.4 Evaluating the results of a Soap query

In the CEM log, you will see the result of the Soap query when you run the IVR. Examples:

{'Country': BE, 'Number': 475334500, 'Return': X, 'ReturnMsg': None}

{'Toocomplex': None, 'Balance': {'Accountnumber': '000101955090', 'Overduebalance': '0.00', 'Grossbalance': '4785.00-', 'Paymenttype': 'D', 'Netbalance': '4785.00-', 'Discountdate': '1900-01-01', 'Duedate': '2007-08-30'}, 'Return': {'Type': 'I', 'Number': '014', 'MessageV4': None, 'MessageV2': None, 'MessageV3': None, 'Message': 'Account Balance Obtained', 'MessageV1': None, 'LogMsgNo': '000000', 'Id': 'ZIVRMES', 'LogNo': None}}



{'Return': {'Type': 'I', 'Number': '021', 'MessageV4': None, 'MessageV2': None, 'MessageV3': None, 'Message': 'Meters Returned', 'MessageV1': None, 'LogMsgNo': '000000', 'Id': 'ZIVRMES', 'LogNo': None}, 'Multi': None, 'Meters': [('item', {'Division': '51', 'Installation': '3000194370', 'Register': '001', 'Equipment': '0000000040099113', 'Oldreading': '29165', 'Newreading': None, 'Meternumber': '0000000008000104'}), ('item', {'Division': '52', 'Installation': '3000417096', 'Register': '001', 'Equipment': '00000000010117702', 'Oldreading': '5846', 'Newreading': None, 'Meternumber': '00000080823161079'})], 'Blocked': None}



8.5 Using the Python syntax of Dictionaries, Lists and Tuples

The following example is a simple python dictionary with 4 key-value pairs. **Note:** BE and X are not strings but can be var or field.

{'Country': BE, 'Number': 475334500, 'Return': X, 'ReturnMsg': None}

The following example contains 2 dictionaries: *Balance* and *Return* inside a dictionary with 3 key-value pairs: *Toocomplex*, *Balance* and *Return*.

{'Toocomplex': None, 'Balance': {'Accountnumber': '000101955090', 'Overduebalance': '0.00', 'Grossbalance': '4785.00-', 'Paymenttype': 'D', 'Netbalance': '4785.00-', 'Discountdate': '1900-01-01', 'Duedate': '2007-08-30'}, 'Return': {'Type': 'I', 'Number': '014', 'MessageV4': None, 'MessageV2': None, 'MessageV3': None, 'Message': 'Account Balance Obtained', 'MessageV1': None, 'LogMsgNo': '000000', 'Id': 'ZIVRMES', 'LogNo': None}}

Assuming your Soap element is called mysoapquery:

- To access the value of Toocomplex, you can use: mysoapquery.get("Toocomplex")
- To access the value of Type, you can use: mysoapquery.get("Return",{}).get("Type")

Complex Return structure

{'Return': {'Type': 'I', 'Number': '021', 'MessageV4': None, 'MessageV2': None, 'MessageV3': None, 'Message': 'Meters



Returned', 'MessageV1': None, 'LogMsgNo': '000000', 'Id': 'ZIVRMES', 'LogNo': None}, 'Multi': None, 'Meters': [('item', {'Division': '51', 'Installation': '3000194370', 'Register': '001', 'Equipment': '0000000040099113', 'Oldreading': '29165', 'Newreading': None, 'Meternumber': '0000000008000104'}), ('item', {'Division': '52', 'Installation': '3000417096', 'Register': '001', 'Equipment': '00000000010117702', 'Oldreading': '5846', 'Newreading': None, 'Meternumber': '00000080823161079'})], 'Blocked': None}

This can be broken down using mysoapquery.get("Return") into:

{'Type': 'I', 'Number': '021', 'MessageV4': None, 'MessageV2': None, 'MessageV3': None, 'Message': 'Meters Returned', 'MessageV1': None, 'LogMsgNo': '000000', 'Id': 'ZIVRMES', 'LogNo': None}

Using mysoapquery.get("Meters",{})[0] results in:

('item', {'Division': '51', 'Installation': '3000194370', 'Register': '001', 'Equipment': '00000000040099113', 'Oldreading': '29165', 'Newreading': None, 'Meternumber': '0000000008000104'})

Using mysoapquery.get("Meters",{})[0][1].get("Division") results in:

51



8.6 Using counters

Using a variable as a counter, you can read each of the entries in *Meters*. Assuming we use the variable *mycounter* and assign it an initial value of 0:

• Using mysoapquery.get("Meters",{})[mycounter] results in:

('item', {'Division': '51', 'Installation': '3000194370', 'Register': '001', 'Equipment': '000000000040099113', 'Oldreading': '29165', 'Newreading': None, 'Meternumber': '00000000008000104'})

• Using mysoapquery.get("Meters",{})[mycounter][1].get("Division") results in:

51

• Incrementing *mycounter* by 1 results in:

('item',{'Division': '52', 'Installation': '3000417096', 'Register': '001', 'Equipment': '00000000010117702', 'Oldreading': '5846', 'Newreading': None, 'Meternumber': '00000080823161079'}) 52

8.7 Using Soap elements with multiple input parameters

Add all the input parameters your soap service is expecting: xml child nodes of the urn node.



<urn:Bapizivrccupdateacc>

<Accountnumber>?</Accountnumber>

- <Cardnumber>?</Cardnumber>
- <Cardtype>?</Cardtype>
- <Expirydate>?</Expirydate>
- <Mode>?</Mode>
- </urn:Bapizivrccupdateacc>

	form → soap					
Element Description soap element is services by using the services of the serv		s used to pull data from external data sources such ng the SOAP protocol.	n as web			
ame	SAP_CreditCa	rdUpdate				
Data Source Expression		reditCardUpdate_wsdl)				
ethod Name	Bapizivrccupd	Bapizivrccupdateacc				
Decementary Making						
Parameter Values		• • •	Up Dow			
Parameter Values	▼	▼ (*) ⊙ (Value	Up Dow			
Parameter Values	ey	Value Z_AccountNumber	Up Dow			
Parameter Values Ke "Accountnumber"	•y	Value Z_AccountNumber Z_CreditCardNumber	Up Dow			
Parameter Values Ke "Accountnumber" "Cardnumber"	₹	Value Z_AccountNumber Z_CreditCardNumber Z_CreditCardType	Up Dow			
Parameter Values "Accountnumber" "Cardnumber" "Cardtype" "Expirydate"	▼	Value Z_AccountNumber Z_CreditCardNumber Z_CreditCardType Z_CreditCardType Z_CreditCardTyploate	Up Dow			



8.8 Complex Query structure – how to add as parameters

8.8.1 Mapping the xml structure to Python elements

The syntax is quite flexible. Some constructs can be expressed in several ways. Others must be expressed with specific syntax.

parameter-value	value
value	simple-value complex-value
simple-value	number string "None"
complex-value	node-dictionary node-list node-tuple
node-dictionary	"{" node-name ":" value [","…] "}"
node-list	"(" complex-value [","] ")"
node-tuple	"(" node-name ["," attributes ["," value]] ")"
attributes	attribute-dictionary "None"
attribute-dictionary	"{" attribute-name ":" simple-value [","] "}"
node-name	string
attribute-name	string

8.8.2 Mapping the xml structure to Python syntax

You must use the list syntax for an element with sub-elements if there are:

• Sub-elements with attributes. Attributes are specified as a dictionary within a list:





• Multiple instances of some sub-element. There are no attributes here so the second item in each list is None:

	Kerr		Maha	
_				
	<td>somerequest></td> <td></td> <td></td>	somerequest>		
	</td <td>ids></td> <td></td> <td></td>	ids>		
		<id>321</id>		
		<id>123</id>		
	<i< td=""><td>ds></td><td></td><td></td></i<>	ds>		
	<urn:s< td=""><td>somerequest></td><td></td><td></td></urn:s<>	somerequest>		



In other cases, you can use either dictionary or list syntax for an element:

<urn:somerequest></urn:somerequest>
<person></person>
<firstname>John</firstname>
<surname>Smith</surname>

```
Dictionary syntax:
```

"person"

	Кеу	Value				
	"person"	{"firstname": "John", "surname": "Smith"}				
List sy	ntax:					
	Kev	Value				

(("firstname",None,"John"),("surname",None,"Smith"))

Here we give the list of meters as a list of lists. Each list item is then a tuple of three things: node name, attributes, and value. The value is just a dictionary specifying the sub-nodes.

<urn:Bapizivrmrupdate>
<Accountnumber>1234567890</Accountnumber>
<Meters>
<item>

<Installation>Inst A</Installation>



<Meternumber>Meter 1</Meternumber>

<Newreading>12345</Newreading>

</item>

<item>

<Installation>Inst B</Installation>

<Meternumber>Meter B</Meternumber>

<Newreading>54321</Newreading>

</item>

</Meters>

</urn:Bapizivrmrupdate>

Key	Value
"Accountnumber"	1234567890
"Meters"	(("Item",None,("Installation":"Inst A","Meternumber":"Meter 1","Newreading":12345}),("Item",None,{"Installation":"Inst B","Meternumber":"Meter 2","Newreading":54321}))



9 Appendix 2

Agent presence can be revealed with the GetNumberInfo function:

```
self.m_AppConf.GetNumberInfo('<agent extension number>')
```

The function returns a Python dictionary containing information of the user.

Examples of return values:

•	Wrong Number: { <mark>'Type': 'UNKNOWN'</mark> , 'OrgNbr': '67676767', 'MappedNbr': '67676767'}
•	Agent logged out: {'IvrLanguage': 'ET', 'MappedNbr': '4401', 'Language': 'EN', 'MaxQue': 3, 'PBX': '', 'OrgNbr': '4401', 'FwdTo': '4555', <mark>'Type': 'LoggedOut'</mark> }
•	<pre>Agent logged in and free (Presence-profile, free): {'Available': 1, 'IvrLanguage': 'ET', 'Slave': 0, 'MappedNbr': '4401', 'Language': 'EN', 'Calls': 0, 'UIStatus': 'StatusPaperWork', 'MaxQue': 3, 'PBX': '', 'Paused': 0, 'OrgNbr': '4401', 'FwdTo': '4555', 'Outbound': 0, 'Type': 'LoggedIn' </pre>
•	<pre>Agent logged in and busy (Presence-profile, busy): {'Available': 0, 'IvrLanguage': 'ET', 'Slave': 0, 'MappedNbr': '4401', 'Language': 'EN', 'Calls': 1, 'UIStatus': 'StatusPaperWork', 'MaxQue': 3, 'PBX': '', 'Paused': 0, 'OrgNbr': '4401', 'FwdTo': '4555', 'Outbound': 0, 'Type': 'LoggedIn' </pre>
•	Agent logged in and free (Absence-profile, free) {'Available': 0, 'IvrLanguage': 'ET', 'Slave': 0, 'MappedNbr': '4401', 'Language': 'EN', 'Calls': 0, 'UIStatus': 'StatusPause', 'MaxQue': 3, 'PBX': '', 'Paused': 1,

'OrgNbr': '4401', 'FwdTo': '4555', 'Outbound': 0, 'Type': 'LoggedIn'

The function can be called and used in the following way:

▼ IVR Editor							¢¢ ¢	₽ ⊼
Add New Up Down	Delete		Þ↓ - ÞŤ	E	Save 🗙 Cancel 🔲 Show C	Child Element Properties		
Element	D	Value	Target	F	Flement Path	form (GetTransferDestina	ation) \rightarrow field (DestinationInput) \rightarrow filled \rightarrow var	^
롦					Elonion Patri	term (sectrumsterbestingien) / neta (bestingieninpar) / nited / var		
✓ form	GetTransferDestination			Element	Element Description	var element is used to declare a parent element.	clare a VoiceXML variable within the scope specified by its	
▼ field	DestinationInput							-
🚇 audio		PROMPT: 'en\\wel			Name	DestinationStatus		1
▼ filled								211
var	DestinationStatus	self.m_AppConf.G		Initial Value Expression self.m_AppConf.GetNumberInfo(Destination		tNumberInfo(DestinationInput)	J	
∗ if		DestinationStatu			Initial Value as Dictionary Data-			
🔿 goto			→ GetTransferDestinati					
▼ form	DoTransfer					•	• • • • op bown	
transfer			DestinationInput	Key Value		Value		



▼ NR Editor 🇠								
Add New Up Down	Delete		Þ∔ - ÞŤ	E	Save X Cancel Show Child Element Properties			
Element	ID	Value	e Target Elamant Dath form (CatTransforDestination) -		form (GetTransferDestination) \rightarrow field (DestinationInput) \rightarrow filled \rightarrow if			
æ					Cieffient Path			
▼ form	GetTransferDestination			Element Description	based on, for example, variable values.			
▼ field	DestinationInput							
🚇 audio		PROMPT: 'en\\wel			Condition	DestinationStatus["Tune"] = "LoggedIn"		
▼ filled				1		and the second sec		
var	DestinationStatus	self.m_AppConf.G			BCM-Specific Attributes			
▼ if	DestinationStatu			Description				
🔿 goto			→ GetTransferDestinati					
✓ form	DoTransfer							
transfer			DestinationInput					



10 Appendix 3

QUEUE.QUERY example 1

The task is to create a conditional IVR that forwards calls to a secondary queue if average waiting time in the primary queue exceeds a specified time value. We modify the ready-made example that checks if there are more calls waiting than the defined maximum number of waiting calls, so that the average waiting time is used as a critical factor.

Also, instead of a customer parameter recommended in the original example, a variable is added.

Primary Queue= 2100 = DefaultQueueSecondary Queue = 2101 = OverflowQueueAverageWaiting= 5= time in seconds

Steps to create:

- 1. Import the "Example_IVR_Conditional.xml" from the Examples and Templates folder.
- Add Element <var> ID "AverageWaiting" Value "5" and <var> ID "OverflowQueue" Value"2101"
- 3. Modify DefaultQueue value to be "2100"
- 4. Modify Element <if> by selecting from the dropdown menu value "Queue Status Query: Average Waiting Time" and edit the line to be like QUEUE.QUERY(str(DefaultQueue), "Sta_AverageWaitingTime") > int(AverageWaiting)
- 5. Modify Element <assign> Value to point to "OverflowQueue"
- 6. Modify attribute Descriptions accordingly as they are shown in the log files.



▼ NR Editor						
Add New Up Down Delete						
Element	ID Value Ta					
ቆ						
var	DefaultQueue	2100				
var	OverflowQueue	2101				
var	DestinationQueue	DefaultQueue				
var	AverageWaiting	5				
▼ form	F1					
* block						
▼ if		QUEUE.QUERY(str(
assign		OverflowQueue	DestinationQueue			
★ form	F2					
transfer			DestinationQueue			



11 Appendix 4

QUEUE.QUERY sample 2

Task is to create an conditional IVR that forwards calls to secondary queue if estimated waiting time in primary queue exceeds spesified time value. To verify functionality add counters to log printing. Primary Queue = 2100 = DefaultQueue Secondary Queue = 2101 = OverflowQueue EstWaiting = 30 = Estimated Waiting Time in seconds QueMaxWaitTime = 660 = Default value if not defined in SC "Queue Management"> "Max.Waiting Time"

Steps to create:

- 1. Import the "Example_IVR_Conditional.xml" from the Examples and Templates folder.
- Add Element <var> ID "AverageWaiting" Value "5" and <var> ID "OverflowQueue" Value"2101"
- Add <var> EstWaiting, Value 30 and <var>QueMAxWaitTime, Value 660 and Modify DefaultQueue value to be "2100"
- Modify Element <if> by selecting from the dropdown menu value "Queue Query: Estimated Waiting Time" and edit the line to be like QUEUE.QUERY(str(DefaultQueue), "Que_EstWaitTime") > int(EstWaiting)
- Add Element <if> from the dropdown menu value "Queue Query: Estimated Waiting Time": QUEUE.QUERY(str(DefaultQueue), "Que_EstWaitTime") <
 - int(QueMaxWaitTime)
- 6. Modify Element <assign> Value to point to "OverflowQueue"
- Add New Child Element <log> by selecting froom dropdown and edit "Expression":
- 8. "IvrApplication.logprint (CALL_ID= " + str({CALLID}) + " Average waiting time = " + str(QUEUE.QUERY(str(DefaultQueue), "Sta_AverageWaitingTime"))


- 9. Add <log>with expression: "IvrApplication.logprint (CALL_ID= " + str({CALLID}) +
 - " Got over estwait time limit, FWD to OverflowQue, EstWait= " +
 - str(QUEUE.QUERY(str(DefaultQueue), "Que_EstWaitTime"))

10. Modify attribute Descriptions accordingly as they are shown in the log files.

