Rapid Development Framework (RDF)

For Amazon Connect 

User Guide 0.1

April 2023

Revision History

VERSION	DATE	DESCRIPTION	AUTHOR
0.1	Apr 18, 2023	This is a draft version. While reasonable efforts have been made to ensure that the information in this document is complete and accurate, CompuCom assumes no liability for any errors and reserves the right to make changes and corrections without the obligation to notify any person or organization of such changes.	Julius Malixi

Table of Contents

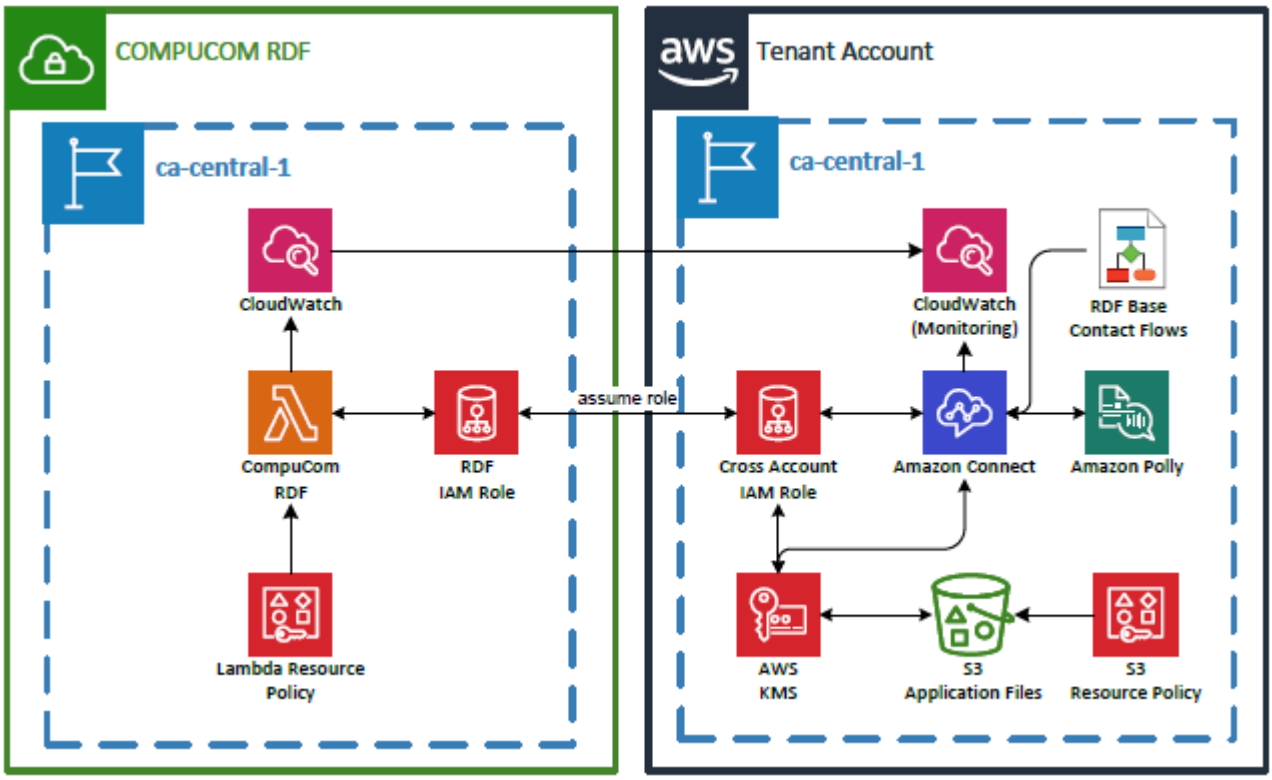
Chapter 1: Overview

The CompuCom Rapid Development Framework (RDF) is a flexible and simple configuration tool used to assist in the rapid development of Interactive Voice Response (IVR) and Queue applications on Amazon Connect. The framework was developed to overcome limitations in Amazon Connect related to Multilingual Support, Call Flow Labels, single prompts within Play and Input blocks, Computations and Scripting, Global and Local Error Handling, IVR Caller Dispositions, Contact Attribute and Data Persistence across modules, DNIS Routing and Integration with backend applications such as SOAP and REST web services.

Amazon Connect provides a great range of features to implement a Contact Center of any size, and the CompuCom RDF tool combines these capabilities into a development framework that delivers robust applications using building blocks normally found in traditional IVR Integrated Development Environments (IDE).

The CompuCom RDF tool was developed using Python and provides developers with simple access to the Python scripting language enabling developers to quickly perform computational and conditional operations within their applications without having to continually break out into custom lambda functions. Seamless integration to other AWS Cloud Services, Web Services, and external applications can be easily customized.

The CompuCom RDF architecture comprises of several pre-built Amazon Connect Contact Flows (RDF Base Contact Flows) that interacts with a Lambda function using cross-account IAM roles hosted on a remote AWS account. It also leverages an S3 Bucket to store application configuration and audio files and a cross-account KMS Key to access these S3 resources. Optional features include DynamoDB tables. Below is a high-level architecture diagram of the CompuCom *Rapid Development Framework*:



Chapter 2: Getting Started

This section describes the initial steps required to get the CompuCom RDF environment ready on Amazon Connect.

Prerequisites

This User Guide assumes that the user has a thorough understanding of IVR application development and has already created an AWS Account with an Amazon Connect Instance and Phone Number.

- ➔ To create a new AWS Account: https://portal.aws.amazon.com/billing/signup#/start/email

- ➔ To create an Amazon Connect Instance: https://docs.aws.amazon.com/connect/latest/adminguide/amazon-connect-instances.html

- ➔ To claim a Phone Number: https://docs.aws.amazon.com/connect/latest/adminguide/tutorial1-claim-phone-number.html

Contact Flow Set Up

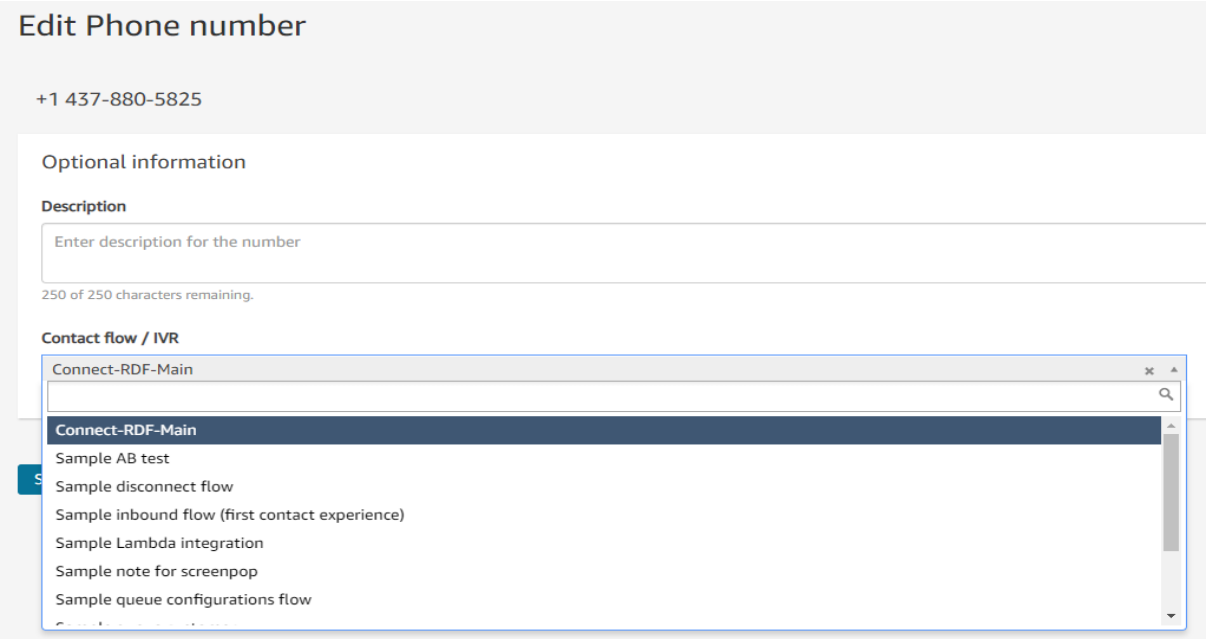
Once you have your Amazon Connect instance and Phone Number ready, download and clone the following RDF Base Contact Flows to a local folder from Code Commit: <url>

- ➔ Clone git repository for Amazon Contact Flows:

Filename	Contact Flow Type
Default customer whisper	Default customer whisper
Connect-RDF-Outbound	Outbound whisper
Connect-RDF-Queue	Customer queue
Connect-RDF-Whisper	Agent whisper
Connect-RDF-Agent	Transfer to queue
Connect-RDF-Main	Contact flow

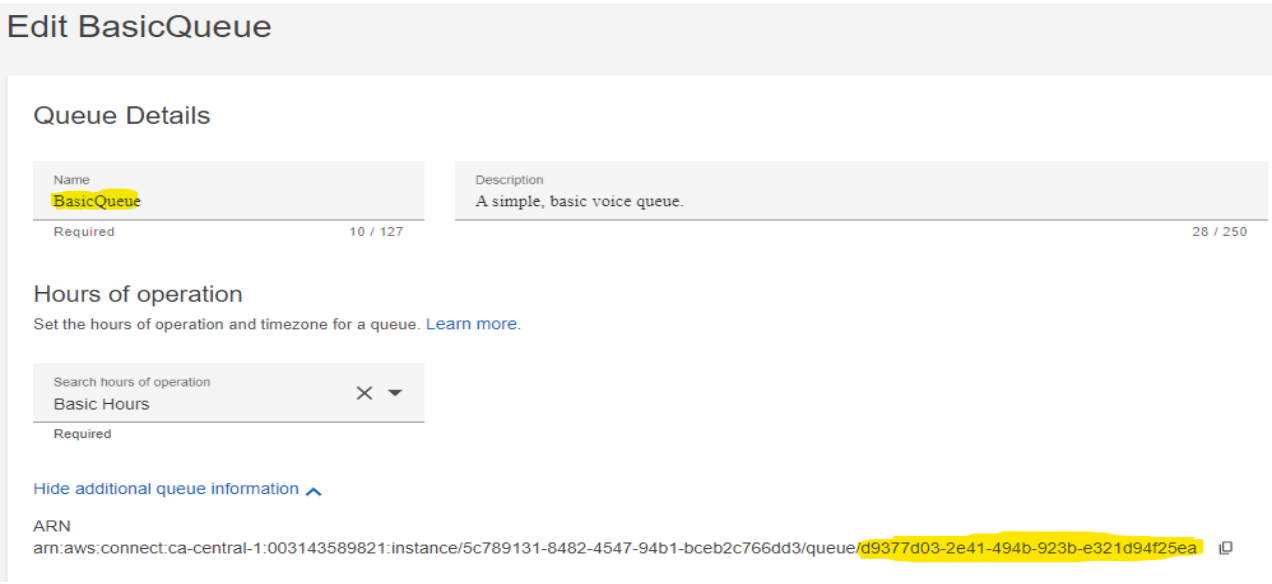
- ➔ Import and publish the above Contact Flows to your Amazon Connect instance, ensuring you select the correct Contact Flow Type. (**Important:** Import and publish the Contact Flows ***in the order listed in the above table.*** This will ensure that any settings dependent from another Contact Flow are published and available in the correct order). For additional details on how to import Amazon Contact flows, refer to the following: https://docs.aws.amazon.com/connect/latest/adminguide/contact-flow-import-export.html

- ➔ Assign the Phone Number(s) to use the **Connect-RDF-Main** Contact Flow:

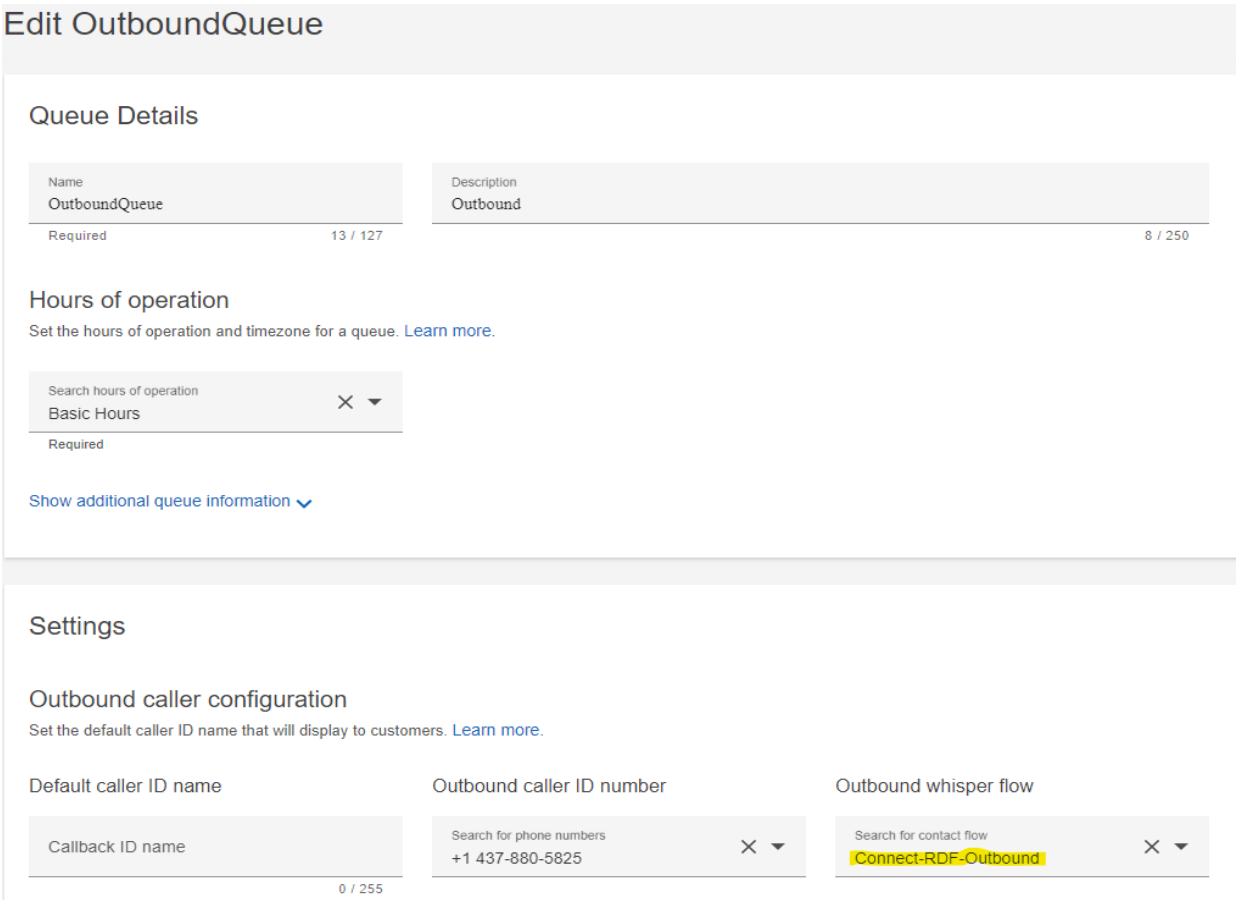


Queue Set Up

➔ If you already have a list of queues needed for your application, create all the queues, and note the Queue **Name and ID** highlighted below. You will need the Queue Name and ID to add to the **queues.csv** configuration file explained in Chapter 4 of this guide:



➔ Create an OutboundQueue and assign the **Connect-RDF-Outbound** Contact Flow and make sure to select the correct Outbound caller ID number you want to use for your instance:



➔ Update the Basic Routing Profile and select the OutboundQueue created in the previous step as the Default outbound queue.

Update Business Hours

➔ Update and rename your Business Hours schedule as needed: https://docs.aws.amazon.com/connect/latest/adminguide/set-hours-operation.html

Add/Update Agent Statuses

➔ Add/Update custom agent statuses: https://docs.aws.amazon.com/connect/latest/adminguide/agent-custom.html

Create Cross-Account IAM Role

Follow these steps to create a cross-account IAM role on your AWS account to allow the remote RDF lambda function to assume this role:

➔ Go to **IAM -> Roles** and click on **Create role.**

➔ Under **“Select trusted entity”**, select **AWS account** as the Trusted entity type.

➔ Under **“An AWS account”**, choose **Another AWS account** and enter the RDF Account ID (037338416846), and click **Next**.

➔ Under **“Add permissions”**

  ➔ In the Filter policies box, type Connect.
  In the list of Amazon Connect policies, select **AmazonConnectFullAccess**

  ➔ In the Filter policies box, type DynamoDB (optional).
  In the list of DynamoDB policies, select **AmazonDynamoDBFullAccess**

➔ For Role name, type **xaccount-rdf-role** (this must be the exact name used for this role)

➔ Choose **Create role.**

Create Cross-Account KMS Key

Follow these steps to create a cross-account KMS Key to be used for the S3 Bucket that is used to store application configuration and audio files:

➔ Go to **Key Management Service** and create Customer Managed KMS Key -> Symmetric, Encrypt and decrypt, KMS, Single-Region and click **Next**.

➔ Under **“Add Labels”**, enter Alias -> **xaccount-s3-rdf-kms** and click **Next**.

➔ Under **“Define key administrative permissions”**, select an appropriate Key administrator from the list of IAM users and roles and check **“Allow key administrators to delete this key”**, then click Next.

➔ Under **“Define key usage permissions”**, scroll down to **“Other AWS accounts”** and click **“Add AWS Account”**. Enter the RDF account number to use this Key -> 037338416846, then click **Next**.

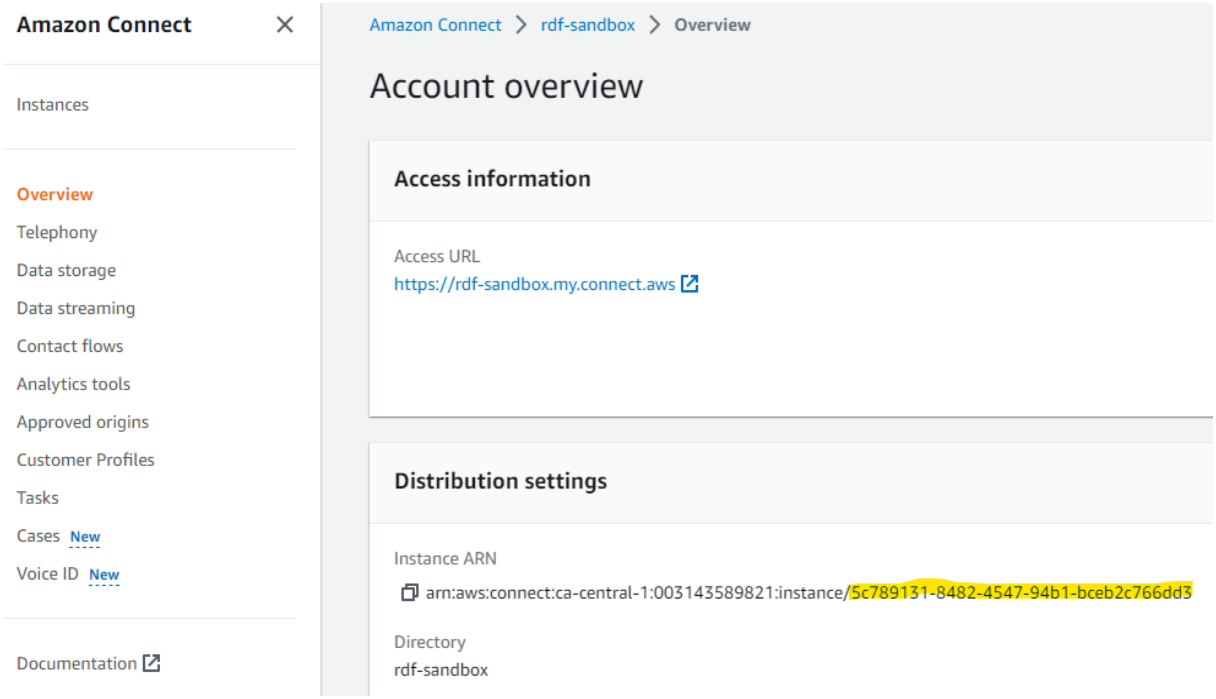
➔ Under **“Review”**, add the following entry to the KMS Key Policy for **connect.amazonaws.com**:

```
{
   "Sid": "Enable Amazon Connect",
   "Effect": "Allow",
   "Principal": {
      "Service": "connect.amazonaws.com"
   },
   "Action": "kms:decrypt",
   "Resource": "*"
},
```

Create Application S3 Bucket

Follow these steps to create an S3 Bucket used to store application configuration and audio files:

➔ Before creating your S3 Bucket, go to Amazon Connect and take note of the Amazon Connect Instance ID (as highlighted in yellow below):



➔ Go to the **S3** Console and select **Create bucket**.

➔ For Bucket Name, use the Amazon Connect Instance ID prefixed with "amazon-connect-" as follows: ***s3://amazon-connect-5c789131-8482-4547-94b1-bceb2c766dd3***

(Replace the Instance ID in the example above with your Connect Instance ID)

➔ Under **"Default encryption"**, choose **"AWS Key Management Service key (SSE-KMS)"** as the Encryption key type, select **"Choose from your AWS KMS Keys"** and from the drop-down list, select the KMS Key **xaccount-s3-rdf-kms** that was created from the previous step.

➔ Click **Create Bucket** and then go to your S3 Bucket's Permissions.

➔ Edit your **Bucket Policy** and add the following entry to provide RDF and Amazon Connect with S3 access to this bucket (update only the highlighted areas of this policy with your S3 bucket name, Amazon Connect ARN and AWS Account Number):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::037338416846:role/service-role/RDF-role-k0m3ejei"
      },
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "arn:aws:s3::: amazon-connect-5c789131-8482-4547-94b1-bceb2c766dd3/*"
    },
    {
      "Sid": "statement1",
      "Effect": "Allow",
      "Principal": {
        "Service": "connect.amazonaws.com"
      },
      "Action": [
        "s3:ListBucket",
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3::: amazon-connect-5c789131-8482-4547-94b1-bceb2c766dd3",
        "arn:aws:s3::: amazon-connect-5c789131-8482-4547-94b1-bceb2c766dd3/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:connect:ca-central-1:003143589821:instance/5c789131-8482-4547-94b1-bceb2c766dd3",
          "aws:SourceAccount": "003143589821"
        }
      }
    }
  ]
}
```

➔ Create the following folders in your application S3 Bucket:

- config/
- prompts/

Create DynamoDB Tables (Optional)

If your application requires DynamoDB tables, create your tables by following these instructions:

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/getting-started-step-1.html

Chapter 3: Hello World!

This chapter walks you through a sample Hello World application that will get your Amazon Connect instance to begin processing calls with the CompuCom RDF. This sample application provides sample code of the common features used in the framework such as different Play Prompts, Web Services, System Administration and Voice Recordings.

➔ Clone git repository for Sample Application Configuration Files:

Filename	Description
config.csv	Instance Configuration File
queues.csv	Queues Configuration File
sample-callflow.csv	Call Flow Configuration File
sample-tts-csv	TTS/SSML Configuration File
sample-sysadmin	System Administration Configuration File
sample-holidays.csv	Holidays Configuration File

➔ Update the queues.csv file and replace the Value column for BasicQueue with the Queue ID from your Connect instance:

	A	B
1	KEY	VALUE
2	BasicQueue	bd0dd595-5160-4aae-89c4-c9743d4897c8

➔ Upload the configuration files to the **config/** folder of your S3 Bucket.

➔ Clone git repository for standard Audio files:

Prompts Folder	Description
en-US/	English Audio Folder containing default audio files
fr-CA/	French Audio Folder containing default audio files

➔ In the prompts folder of your S3 Bucket, create a sub-folder named “sample”. This “sample” sub-folder is used for demonstration purposes only. In a real-world application, this name will be replaced by a tenant or application name.

➔ Upload the English and French audio folders to the **prompts/sample/** folder of your S3 Bucket.

➔ Your Application S3 Bucket should have the following folder structure:

- ***config/***
- ***prompts/sample/en-US***
- ***prompts/sample/fr-CA***

➔ Send a request to CompuCom with your Amazon Connect Instance ID to provide proper permissions for your Instance to invoke the RDF lambda function.

➔ Make a test call to your Amazon Connect phone number.

Chapter 4: Application Configuration

This chapter describes how to create a tenant application and what configuration files are used by the CompuCom RDF. All the files outlined in this section are the configuration files needed to control your Amazon Connect application. In other words, there is no need to update any of the base RDF Contact Flows in Amazon Connect. You can simply upload these application configuration files to the S3 Bucket and the changes will take effect immediately. See section "RDF Lambda Cache" in Chapter 9 to understand how caching has been implemented on the framework.

config.csv

The *config.csv* file is a tenant specific configuration file used to define your application's default settings. This is the **initial** configuration file that the CompuCom RDF uses that defines all other configuration files used in your application such as Callflow(s), TTS, Holiday, and System Administration files. All other parameters defined in this file control the global behaviour of your application for error handling and global treatment of DTMF keys. These global settings alleviate the need to continually define repetitive settings throughout your call flow. However, these settings can also be overridden at anytime in your call flow as needed. For example, in the configuration sample below, since we set option 9 globally to return a caller to a previous menu, there is no need to define option 9 elsewhere in the application unless you want option 9 to override the global behaviour at a specific menu. This also applies to option * used for repeat.

	A	B
1	KEY	VALUE
2	tenant	sample
3	callFlow	sample-callflow.csv
4	ttsFile	sample-tts.csv
5	holidayFile	sample-holidays.csv
6	sysAdminFile	sample-sysadmin.csv
7	connectFlowType	INBOUND
8	defaultDNIS	IVR100
9	defaultQueue	BasicQueue
10	language	en-US
11	globalMenuAttempts	2
12	globalInvalidPrompt	tts:InvalidInput1|tts:InvalidInput2
13	globalTimeoutPrompt	tts:NoInput1|tts:NoInput2
14	globalMaxtriesPrompt	tts:MaxInputXfer
15	globalMaxtriesAction	PQ100
16	globalMenuRepeat	*
17	globalMenuPrevious	9
18	globalZeroPrompt	agent
19	globalInputTerm	#
20	speechAnalytics	off
21	chatAnalytics	off

The following table provides a description of all the config.csv parameters used in the application:

PARAMETER	DESCRIPTION	EXAMPLE
tenant	**MANDATORY –** Defines the tenant name used for your application. This is also the name used for the sub-folder in your **prompts/** folder.	sample
callFlow	**MANDATORY –** Defines the main callflow csv file that will be used in your application. A tenant can also have multiple sub callflows. Please review the SUB Block Type in Chapter 5 for more details.	sample-callflow.csv
ttsFile	**MANDATORY –** Defines the TTS csv file that will be used in your application. The TTS configuration files defines all the TTS scripts used in your application. The TTS configuration file supports multiple languages.	sample-tts.csv
holidayFile	**OPTIONAL –** Defines the Holidays configuration for your application. It lists all the holiday dates observed by your tenant application.	sample-holidays.csv
connectFlowType	**MANDATORY –** Defines whether your application is being used for Inbound or Outbound calls.	INBOUND/OUTBOUND
defaultDNIS	**MANDATORY –** Defines the first BLOCK ID to execute in your callflow csv file if a specific DNIS record is not found in your call flow. See the DNIS Block Type for more details in Chapter 5 on how you can route to different starting blocks based on DNIS.	IVR100
language	**MANDATORY –** Defines the default language for your application. This will usually be set to en-US.	en-US / fr-CA
globalMenuAttempts	**MANDATORY –** Defines the maximum number of re-attempts for each Menu or Input Block in your application.	2
globalInvalidPrompt	**MANDATORY –** Defines the default prompts used for invalid menu attempts by a caller. You can define different prompts to be played based on each invalid attempt up to the maximum number of attempts defined by	tts:InvalidInput1|tts:InvalidInput2

	globalMenuAttempts. Each prompt is separated by a "|" pipe delimiter. See the PLAY Block Type in Chapter 5 for more details on usage. These prompts are non-bargeable.	
globalTimeoutPrompt	**MANDATORY –** Defines the default prompts used for no entry attempts by a caller. You can define different prompts to be played based on each no entry attempt up to the maximum number of attempts defined by globalMenuAttempts. Each prompt is separated by a "|" pipe delimiter. See the PLAY Block Type in Chapter 5 for more details on usage. These prompts are non-bargeable.	tts:NoInput1|tts:NoInput2
globalMaxPrompt	**MANDATORY –** Defines the final prompt played if a caller reaches the maximum number of attempts defined by globalMenuAttempts. This is a single prompt only and is non-bargeable.	tts:MaxInputXfer
globalMaxAction	**MANDATORY –** Defines the Block ID to execute if a caller reaches the maximum number of attempts defined by globalMenuAttempts. This usually goes to the starting block of a Pre-Queue flow or a graceful disconnect block.	PQ100
globalMenuRepeat	**OPTIONAL –** Defines a global DTMF key used to allow a caller to repeat any Menu within your application. This global setting can be overridden if the key is defined locally for a Menu.	*
globalMenuPrevious	**OPTIONAL –** Defines a global DTMF key used to allow a caller to go to a previous menu. The CompuCom RDF tracks a caller as they traverse through different menu's. If a global setting is defined but there's a specific menu that you don't want a caller to be able to return to, override the previous menu key for that Menu. An example of this would be for the Language Menu.	9
globalZeroPrompt	**OPTIONAL –** Used to suppress an option Zero prompt if a caller is calling outside of business hours. If this prompt is defined within a Menu block, the prompt will only play during business hours and suppressed outside of business hours.	agent

globalInputTerm	**OPTIONAL –** Defines the key used to terminate variable length input from a caller.	#
speechAnalytics	**OPTIONAL –** Determines whether speechAnalytics should be turned **off** or set to **real** time or **post** time analytics. If not specified, the default value is set to **off**.	off/real/post
chatAnalytics	**OPTIONAL –** Determines whether chatAnalytics should be turned **off** or **on**. If not specified, the default value is set to **off**.	off/on

queues.csv

The *queues.csv* file is a tenant-specific configuration file that defines all of the Queue Names and Queue ID's used in your application. This configuration file allows a developer to reference Queues by Name within their applications without having to set the Queue ID that is required by Amazon Connect. Refer to Block Type SETQUEUE in chapter 5 for additional details.

	A	B
1	KEY	VALUE
2	BasicQueue	d9377d03-2e41-494b-923b-e321d94f25ea
3		

<tenant>-callflow.csv

The *<tenant>-callflow.csv* file is the main callflow application file used by your tenant. Replace <tenant> with the tenant name defined in config.csv. You can have multiple callflow files developed as sub call flow applications. To call a sub call flow, you can use the SUB Call Flow Block Type as described in Chapter 5. Refer to Chapter 5 for Application Development details on how to configure your call flow blocks.

	A	B	C	D
1	BLOCK	TYPE	PARAM	ACTION
2	IVR100	PLAY	wav:greeting	IVR125
3	IVR125	MENU	wav:language	2:IVR125F|9:IVR150|*:IVR150|timeout:IVR150|invalid:IVR150
4	IVR125F	LANGUGE	fr-CA	IVR150
5	IVR150	PLAY	eval:util.getSysAdmin("BM01")	IVR151
6	IVR151	PLAY	eval:util.getSysAdmin("BM02")	IVR152
7	IVR152	PLAY	eval:util.getSysAdmin("BM03")	IVR175
8	IVR175	SETQUEUE	en-US:BasicQueue|fr-CA:BasicQueue	IVR200
9	IVR200	MENU	tts:mainmenu|tts:agent	1:IVR300|2:IVR400|3:IVR500|0:PQ100
10	IVR300	MENU	tts:promptsmenu|tts:agent	1:IVR310|2:IVR320|3:IVR330|4:IVR340|0:PQ100
11	IVR310	INPUT	tts:getinput	var:my_variable|length:5|action:IVR311|seconds:3
12	IVR311	PLAY	tts:ascardinal|cardinal:session["my_variable"]	IVR312

<tenant>-tts.csv

The *<tenant>-tts.csv* file is the application's TTS definition file. Replace <tenant> with the tenant name defined in config.csv. This file contains the script labels and verbiage for your application scripts used by Amazon Polly to synthesize your IVR application prompts in multiple languages. The TTS entries in this configuration files also supports SSML tags. To see a list of supported SSML tags, please refer to the following:

https://docs.aws.amazon.com/connect/latest/adminguide/supported-ssml-tags.html

IMPORTANT: When using SSML tags for your application scripts, DO NOT include the opening and closing <speak> and </speak> tags as the CompuCom RDF automatically inserts these tags at runtime.

	A	B	C
1	LABEL	en-US	fr-CA
2	greeting	Thank you for calling the Compucom Rapid Development Framework sample application.	Merci d'avoir appelé Compucom - Rapid Development Framework, exemple d'application.
3	language	For service in English, press 1 or stay on the line. Pour obtenir des renseignements en français, faites le deux	
4	BM01	Broadcast Message 1. You can enable and disable this message using the IVR Administration line.	Message de diffusion 1. Vous pouvez activer et désactiver ce message à l'aide de la ligne d'administration IVR.
5	BM02	Broadcast message 2. While you test out some of the features of this sample application, you can always press star to repeat a menu, or press 9 to return to the previous menu.	Il s'agit du message de diffusion 2. Pendant que vous testez certaines des fonctionnalités de cet exemple d'application, vous pouvez toujours appuyer sur étoile pour répéter un menu ou appuyer sur 9 pour revenir au menu précédent.

The LABEL column is used to reference the application scripts in your call flows. The different language columns define the verbiage for each Script Label for the appropriate language. See PLAY Block Type in Chapter 5 for more details.

<tenant>-holidays.csv

The *<tenant>-holidays.csv* file is the application's holidays definition file. Replace <tenant> with the tenant name defined in config.csv. This configuration file lists all the holidays observed by your tenant. Currently, each tenant has its own holiday file. However, holiday definition files can be centralized and shared by a common S3 bucket. The format for DATE should be in M/Y/DDDD format (do not include preceding 0's when defining Month's or Day's).

	A	B
1	DATE	DESCRIPTION
2	12/25/2022	Christmas Day
3	12/26/2022	Boxing Day
4	12/27/2022	Christmas Day
5	1/1/2023	New Years Day
6	1/2/2023	New Years Day
7	2/20/2023	Family Day
8	4/7/2023	Good Friday
9	5/22/2023	Victoria Day
10	7/1/2023	Canada Day
11	8/7/2023	Civic Holiday
12	9/4/2023	Labour Day
13	10/9/2023	Thanksgiving Day
14	12/25/2023	Christmas Day
15	12/26/2023	Boxing Day
16	1/1/2024	New Years Day
17		

<tenant>-sysadmin.csv

The *<tenant>-sysadmin.csv* file is the application's System Administration file. Replace <tenant> with the tenant name defined in config.csv. This configuration file is used to define your application's Broadcast Messages and Dynamic Menus. For Broadcast Messages, refer to the PLAY Block **eval** type in Chapter 5. For Dynamic Menus, refer to the MENU Block **eval** ACTION parameters in Chapter 5. Also review Chapter 6 to learn how these Broadcast Messages and Dynamic Menus can be controlled by the IVR System Administration Line.

	A	B
1	KEY	VALUE
2	APPCODE	1234
3	VM	FALSE
4	BM01	tts:BM01
5	BM02	tts:BM02
6	BM03	tts:BM03
7		

Chapter 5: Application Development

This chapter describes all the different building blocks required to develop an application with the CompuCom RDF. The foundation of the framework is driven by CSV files that can be created and modified using a simple CSV Editor such as Microsoft Excel. When saving a csv file, please ensure to select the *"CSV UTF-8 (Comma delimited)(*.csv)"* option shown below:

CALL FLOWS

The Call Flow CSV file consists of 4 columns – **BLOCK, TYPE, PARAM,** and **ACTION** shown here:

	A	B	C	D
1	BLOCK	TYPE	PARAM	ACTION
2				
3				
4				

BLOCK

The BLOCK column contains unique Block ID's that identifies each of the specific call flow blocks of an application. The BLOCK names are user-defined and are usually based on the Call Flow labels of a Call Flow Design Document. The BLOCK name is also used to capture Caller Dispositions (see Chapter 9). When used with a DNIS Block Type, the BLOCK name should contain the DNIS of the dialed phone number. By associating multiple DNIS' to the same call flow, different ACTION's (or caller treatment) can be taken based on the dialed phone number.

TYPE

The TYPE column is used to describe the function of a specific call flow block. Each Block TYPE provides different functions as outlined in section **"Block Types"**.

PARAM

The PARAM column is used to set parameters specific to a Block TYPE. The parameters used for each Block TYPE are outlined in section **"Block Types"**.

ACTION

The ACTION column is used to set the next set of actions based on the Block TYPE. The action parameters used for each Block Type are outlined in section **"Block Types"**.

Session Attributes

Session attributes are like Contact Attributes used in Amazon Connect Contact Flows. Unlike Contact attributes, Session attributes are used within the CompuCom RDF namespace and does not get saved within the Contact Trace Record (CTR). Session attributes can contain temporary storage to store variables within your application that you don't want to include within your Contact Trace Records or reports. To save attributes to a Contact Trace Record, use the USERDATA Block Type as described in Chapter 5.

Session attributes can also be used in your applications for computational expressions or logical evaluations such as a loop counter, arithmetic, and conditional statements. You can store user input, save data from a backend web service, and use the contents of these attributes throughout your applications.

The format used to reference a Session attribute is:

session['attribute_name'] = attribute_value

Where, **attribute_name** is a unique name for your attribute, and **attribute_value** can contain a fixed value or the name of another session attribute. The session attribute can contain any Python Built-in Data Types.

Python Built-in Data Types

Session attributes can store data of different types, and different types can do different things. Python has the following data types built-in by default, in these categories:

Text Type:	`str`
Numeric Types:	`int, float, complex`
Sequence Types:	`list, tuple, range`
Mapping Type:	`dict`
Set Types:	`set, frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes, bytearray, memoryview`
None Type:	`NoneType`

There is no limit to the number of Session attributes you can use in your applications.

BLOCK TYPES

PLAY

The PLAY block is used to play a series of application scripts in different formats. The list of formats is extensive and the values can be either be a fixed value or a Session variable.

TYPE	PARAM	ACTION
PLAY	type:value|type:value	nextBlock

PARAM

Defines a series of application scripts in the format **type:value** separated by the pipe "|" delimiter.

type	**wav** – play a wav file from your application's S3 bucket. **tts** – play tts scripts referenced in your <tenant>-tts.csv. **var** – play the contents of a session variable. **eval** – play a value returned from a Python expression or pre-built function. **text** – play free text. **currency** – play currency in the format of <dollarvalue>.<centsvalue> (i.e., 123.45) **characters or spell-out –** spells out each letter of the text, as in a-b-c. **cardinal or number –** Interprets the numerical text as a cardinal number, as in 1,234. **ordinal – i**nterprets the numerical text as an ordinal number, as in 1,234th. **digits –** spells out each digit individually, as in 1-2-3-4. **fraction –** interprets the numerical text as a fraction. This works for both common fractions such as 3/20, and mixed fractions, such as 2 ½. See below for more information. **unit – i**nterprets a numerical text as a measurement. The value should be either a number or a fraction followed by a unit with no space in between as in 1/2inch, or by just a unit, as in 1meter. **date –** interprets the text as a date. The format must be yyyymmdd you can make Amazon Polly skip parts of the date using question marks.

	time – interprets the numerical text as duration, in minutes and seconds, as in 1'21". **address** – interprets the text as part of a street address. **expletive** – "beeps out" the content included within the tag. **telephone –** interprets the numerical text as a 7-digit or 10-digit telephone number, as in 2025551212. You can also use this value for handle telephone extensions, as in 2025551212x345.
value	A fixed value or a session attribute

ACTION

Defines the Next BLOCK to execute in the call flow.

nextBlock	The next BLOCK to execute.

MENU

The MENU block is used to play a series of application scripts in different formats and allows a caller to select a DTMF menu option. The caller will navigate to the next BLOCK based on the DTMF menu option selected.

TYPE	PARAM	ACTION
MENU	type:value|type:value	option:nextBlock|option:nextBlock

PARAM

Defines a series of application scripts in the format **type:value** separated by the pipe "|" delimiter.

type	**wav** – play a wav file from your application's S3 bucket. **tts** – play tts scripts referenced in your <tenant>-tts.csv. **var** – play the contents of a session variable. **eval** – play a value returned from a Python expression or pre-built function. **text** – play free text. **currency** – play currency in the format of <dollarvalue>.<centsvalue> (i.e., 123.45) **characters or spell-out –** spells out each letter of the text, as in a-b-c.

	cardinal or number – Interprets the numerical text as a cardinal number, as in 1,234. **ordinal – i**nterprets the numerical text as an ordinal number, as in 1,234th. **digits –** spells out each digit individually, as in 1-2-3-4. **fraction –** interprets the numerical text as a fraction. This works for both common fractions such as 3/20, and mixed fractions, such as 2 ½. See below for more information. **unit – i**nterprets a numerical text as a measurement. The value should be either a number or a fraction followed by a unit with no space in between as in 1/2inch, or by just a unit, as in 1meter. **date –** interprets the text as a date. The format must be yyyymmdd you can make Amazon Polly skip parts of the date using question marks. **time –** interprets the numerical text as duration, in minutes and seconds, as in 1'21". **address** – interprets the text as part of a street address. **expletive** – "beeps out" the content included within the tag. **telephone –** interprets the numerical text as a 7-digit or 10-digit telephone number, as in 2025551212. You can also use this value for handle telephone extensions, as in 2025551212x345.
value	A fixed value or a session attribute

ACTION

Defines a set of actions based on a caller's response in the format **option:nextBlock**. Actions are separated by the pipe "|" delimiter.

option	**barge** (optional) – Sets barge-in to **true** or **false**. When not specified, default is true. **0-9, *, #** (optional)**–** Sets the action for a DTMF option. If a DTMF option is not defined, it will be treated as invalid input or behave based on global settings defined in config.csv (i.e., Option * for repeat or Option 9 for previous menu). **timeout** (optional) – Sets the next action if a caller does not make a menu selection (i.e., No Input). If timeout is not defined, it will inherit the global settings of globalTimeoutPrompt, globalMaxtriesPrompt, and globalMaxtriesAction as defined in config.csv. **invalid** (optional) **–** Sets the next action if a caller makes an invalid selection (i.e., Invalid Input). If invalid is not defined, it will inherit the global settings of

	globalInvalidPrompt, globalMaxtriesPrompt, and globalMaxtriesAction as defined in config.csv. **maxaction** (optional) – Sets the next action if a caller reaches the maximum number of re-tries as defined by globalMenuAttempts. If maxaction is not defined, it will inherit the global settings of globalMaxtriesAction as defined in config.csv.
nextBlock	The next BLOCK to execute.

INPUT

The INPUT block is used to play a series of application scripts in different formats and allows a caller to input a DTMF string that gets saved to a session attribute.

TYPE	PARAM	ACTION
INPUT	type:value|type:value	param:value|param:value

PARAM

Defines a series of application scripts in the format **type:value** separated by the pipe "|" delimiter.

type	**wav** – play a wav file from your application's S3 bucket. **tts** – play tts scripts referenced in your <tenant>-tts.csv. **var** – play the contents of a session variable. **eval** – play a value returned from a Python expression or pre-built function. **text** – play free text. **currency** – play currency in the format of <dollarvalue>.<centsvalue> (i.e., 123.45) **characters or spell-out –** spells out each letter of the text, as in a-b-c. **cardinal or number –** Interprets the numerical text as a cardinal number, as in 1,234. **ordinal – i**nterprets the numerical text as an ordinal number, as in 1,234th. **digits –** spells out each digit individually, as in 1-2-3-4. **fraction –** interprets the numerical text as a fraction. This works for both common fractions such as 3/20, and mixed fractions, such as 2 ½. See below for more information.

	unit – interprets a numerical text as a measurement. The value should be either a number or a fraction followed by a unit with no space in between as in 1/2inch, or by just a unit, as in 1meter. **date –** interprets the text as a date. The format must be yyyymmdd you can make Amazon Polly skip parts of the date using question marks. **time –** interprets the numerical text as duration, in minutes and seconds, as in 1'21". **address** – interprets the text as part of a street address. **expletive** – "beeps out" the content included within the tag. **telephone –** interprets the numerical text as a 7-digit or 10-digit telephone number, as in 2025551212. You can also use this value for handle telephone extensions, as in 2025551212x345.
value	A fixed value or a session attribute

ACTION

Defines a set of action parameters to define the expected DTMF INPUT. Action parameters are separated by the pipe "|" delimiter.

param	**length** (mandatory)– Defines the expected length of the DTMF input string. **action** (mandatory) **–** Defines the next Block to execute when the caller successfully enters the expected number of DTMF digits based on length. **var** (mandatory) **–** Defines the session attribute to save the caller's DTMF input. **barge** (optional) – Sets barge-in to **true** or **false**. When not specified, default is true. **seconds** (optional) – Defines the number of seconds to wait between DTMF input (i.e., interdigit timeout value). If not set, default value is 10. **timeout** (optional) – Sets the next action if a caller does not input any DTMF (i.e., No Input). If timeout is not defined, it will inherit the global settings of globalTimeoutPrompt, globalMaxtriesPrompt, and globalMaxtriesAction as defined in config.csv. **invalid** (optional) **–** Sets the next action if a caller makes an invalid selection (i.e., Invalid Input). If invalid is not defined, it will inherit the global settings of globalInvalidPrompt, globalMaxtriesPrompt, and globalMaxtriesAction as defined in config.csv.

	maxaction (optional) **–** Sets the next action if a caller reaches the maximum number of re-tries as defined by globalMenuAttempts. If maxaction is not defined, it will inherit the global settings of globalMaxtriesAction as defined in config.csv. **0-9, *, #** (optional) **–** Sets the action for the first DTMF key entered by a caller. If not defined, the first digit will be included within the full DTMF string entered by a caller. **IMPORTANT:** Do not set this option if the first DTMF digit is a valid value for the remaining input.
value	The next BLOCK to execute or as defined per INPUT param settings above.

LANGUAGE

The LANGUAGE block is used to set the language. Once set, all prompts that follow will play in the appropriate language for both audio wav files and TTS.

TYPE	PARAM	ACTION
LANGUAGE	value	nextBlock

PARAM

Defines the ISO standard language code.

value	ISO standard language code (en-US, fr-CA)

ACTION

Defines the Next BLOCK to execute in the call flow.

nextBlock	The next BLOCK to execute or as defined per INPUT param settings above.

SUB

The SUB block is used to invoke another call flow in your application. Sub call flows are reusable and repeatable sections of a call flow to create common functions. For example, you can create a Credit Card payment flow and use this as a SUB call flow reused by other applications or tenants. **IMPORTANT:** When creating a SUB call flow, the starting BLOCK must be set to “START” and the final action should be set to “RETURN” or “DISCONNECT”.

TYPE	PARAM	ACTION
SUB	value	nextBlock

PARAM

Defines the sub call flow to invoke.

value	The name of the Sub callflow csv file. **NOTE:** You do not need to include the file extension.

ACTION

Defines the Next BLOCK to execute when the Sub Call flow returns.

nextBlock	The next BLOCK to execute.

EXEC

The EXEC block provides access to a dynamically generated Python expression such as assigning a value to a session attribute, arithmetic, incrementing or decrementing a counter, or to invoke any pre-built functions. The EXEC block does not return a result.

TYPE	PARAM	ACTION
EXEC	expression	nextBlock

PARAM

Used to define a Python expression to execute.

expression	A valid Python expression. (i.e., session[“counter”] += 1)

ACTION

Defines the Next BLOCK to execute.

nextBlock	The next BLOCK to execute.

EVAL

The EVAL block provides access to a dynamically generated Python expression, such as evaluating the results of a session attribute, conditional statement, or pre-built function. The EVAL block evaluates the expression and saves the result in a **session[“eval_response”]** attribute and branches based on whether the result was true or false.

TYPE	PARAM	ACTION
EVAL	expression	true:nextBlock|false:nextBlock

PARAM

Used to define a Python expression to execute.

expression	A valid Python expression. (i.e., session[“counter”] == 3)

ACTION

Defines the Next BLOCK to execute.

true	The next BLOCK if the evaluated expression is True.
false	The next BLOCK if the evaluated expression is False.
nextBlock	The next BLOCK to execute based on true or false.

DNIS

The DNIS block is used to provide different caller treatment based on the dialed phone number. Different ACTION’s can be set to execute different BLOCK’s based on DNIS. **NOTE:** the BLOCK column for DNIS should contain the dialed phone number.

TYPE	PARAM	ACTION
DNIS	expression (optional)	nextBlock

PARAM

Like the EXEC block, the PARAM field can be used to include a dynamically generated Python expression. This is useful for initializing any session attributes you may want to initialize based on DNIS.

expression	A valid Python expression. (i.e., session[“app”] = “custom”)

ACTION

Defines the Next BLOCK to execute in the call flow.

nextBlock	The next BLOCK to execute.

CASE

The CASE block is used as a conditional switch statement to route calls based on the evaluation of a session attribute or Python expression.

TYPE	PARAM	ACTION
CASE	expression	result:nextBlock|result:nextBlock

PARAM

Like the EXEC block, the PARAM field can be used to define a Python expression.

expression	A valid Python expression. (i.e., session["result"])

ACTION

Defines the Next BLOCK to execute in the call flow based on the result from expression.

result	The expected results as evaluated from the Python expression. For example, when calling a web service, you may want to handle a caller based on the HTTP response status codes: 200:IVR500|400:IVR600|404:IVR700|else:PQ100 Call flow will take the **else** path if none of the conditions are met.
nextBlock	The next BLOCK to execute based on result.

USERDATA

The USERDATA block is used to save a value to the **userdata** Contact attribute. Since all session attributes belong to the CompuCom RDF namespace, session attributes are never saved to the CTR. The USERDATA block can be used to save one or more values as a Python list[] or dict[] data type. The userdata Contact attribute will be saved in the CTR.

TYPE	PARAM	ACTION
USERDATA	expression	nextBlock

PARAM

Like the EVAL block, the PARAM field can be used to define a Python expression.

expression	A valid Python expression. (i.e., session["account_number"])

ACTION

Defines the Next BLOCK to execute in the call flow.

nextBlock	The next BLOCK to execute based on result.

SETQUEUE

The SETQUEUE block is used to set the working queue based on a caller's language selection. It is best to call this block after a language selection has been made.

TYPE	PARAM	ACTION
SETQUEUE	language:queue|language:queue	nextBlock

PARAM

The PARAM field is used to set the working queue based on language.

language	**en-US** – Set the working queue if the caller's language selection is English. **fr-CA** – Set the working queue if the caller's language selection is French.
queue	The queue name as defined in queues.csv. By defining the queue name, CompuCom RDF can obtain the Queue ID and ARN needed by Amazon Connect.

ACTION

Defines the Next BLOCK to execute in the call flow.

nextBlock	The next BLOCK to execute.

QUEUE

The QUEUE block is used to transfer a caller to Queue. By calling this block, control is transferred to **Connect-RDF-Queue** Contact Flow.

TYPE	PARAM	ACTION
QUEUE	(NOT USED)	nextBlock

PARAM

Not Applicable. PARAM is not required for the QUEUE Block.

ACTION

Defines the Next BLOCK to execute in the call flow.

nextBlock	The first BLOCK to execute when caller enters the Queue flow.

XFER

The XFER block is used to transfer a caller to an external phone number.

TYPE	PARAM	ACTION
XFER	phone	nextBlock

PARAM

The PARAM field is used to set the external phone number to transfer to.

phone	The 11-digit external phone number to transfer. You must include a preceding "1" with the 10-digit phone number.

ACTION

Defines the Next BLOCK to execute in the call flow.

nextBlock	The next BLOCK to execute.

REST

The REST block is used to interact with a RESTful web service. This is in preview and is subject to change.

TYPE	PARAM	ACTION
REST	api:url|operation:type|json:data	true:nextBlock|false:nextBlock

PARAM

The PARAM field is used …

api	**url** – defines the API URL
operation	**type –** defines the REST API operation (get, post, put, patch, delete)
json	**data –** defines the input data in json format.

ACTION

Defines the Next BLOCK to execute in the call flow.

true	The next BLOCK if the SOAP web service is successful.
false	The next BLOCK if the SOAP web service fails.
nextBlock	The next BLOCK to execute based on true or false.
nextBlock	The next BLOCK to execute.

SOAP

The SOAP block is used to interact with a SOAP web service.

TYPE	PARAM	ACTION
SOAP	wsdl:url|operation:name	true:nextBlock|false:nextBlock

PARAM

The PARAM field is used …

wsdl	**url –** Defines the WSDL URL.
operation	**name** – Enter the operation to invoke as defined by the WSDL with appropriate parameters.

ACTION

Defines the Next BLOCK to execute.

true	The next BLOCK if the SOAP web service is successful.
false	The next BLOCK if the SOAP web service fails.
nextBlock	The next BLOCK to execute based on true or false.

RECORD

The RECORD block is used to play a series of application scripts in different formats and allows a caller to record a message and save the recording to an S3 Bucket.

TYPE	PARAM	ACTION
RECORD	type:value|type:value	nextBlock

PARAM

Defines a series of application scripts in the format **type:value** separated by the pipe "|" delimiter.

type	**wav** – play a wav file from your application's S3 bucket. **tts** – play tts scripts referenced in your <tenant>-tts.csv. **var** – play the contents of a session variable. **eval** – play a value returned from a Python expression or pre-built function. **text** – play free text. **currency** – play currency in the format of <dollarvalue>.<centsvalue> (i.e., 123.45) **characters or spell-out –** spells out each letter of the text, as in a-b-c. **cardinal or number –** Interprets the numerical text as a cardinal number, as in 1,234. **ordinal – i**nterprets the numerical text as an ordinal number, as in 1,234th. **digits –** spells out each digit individually, as in 1-2-3-4. **fraction –** interprets the numerical text as a fraction. This works for both common fractions such as 3/20, and mixed fractions, such as 2 ½. See below for more information. **unit – i**nterprets a numerical text as a measurement. The value should be either a number or a fraction followed by a unit with no space in between as in 1/2inch, or by just a unit, as in 1meter. **date –** interprets the text as a date. The format must be yyyymmdd you can make Amazon Polly skip parts of the date using question marks. **time –** interprets the numerical text as duration, in minutes and seconds, as in 1'21". **address** – interprets the text as part of a street address. **expletive** – "beeps out" the content included within the tag. **telephone –** interprets the numerical text as a 7-digit or 10-digit telephone number, as in 2025551212. You can also use this value for handle telephone extensions, as in 2025551212x345.

value	A fixed value or a session attribute

ACTION

Defines the Next BLOCK to execute in the call flow.

nextBlock	The next BLOCK to execute.

Chapter 6: System Administration

The CompuCom RDF contains a pre-built IVR System Administration module that can be re-used and "plugged-in" to other tenant applications to allow administrators to enable and disable Broadcast Messages and Dynamic Menu options. From the callflows downloaded in Chapter 4, a **sysadmin-callflow.csv** module is included that interacts with the **<tenant>-sysadmin.csv** to manage these settings. The <tenant>-sysadmin.csv file can be used to add other configurable parameters for your application.

The following is a sample **<tenant>-sysadmin.csv** file:

	A	B
1	KEY	VALUE
2	APPCODE	1234
3	VM	FALSE
4	BM01	tts:BM01
5	BM02	tts:BM02
6	BM03	tts:BM03
7	DM100	1:IVR310|2:IVR320|3x:IVR330|4x:IVR340|0:PQ100

Broadcast Messages

Broadcast Messages can be used in a PLAY block with the **eval** type to call the **util.getSysAdmin()** pre-built function. The following are example broadcast messages set in **<tenant>-callflow.csv**:

	A	B	C	D
1	BLOCK	TYPE	PARAM	ACTION
5	IVR150	PLAY	eval:util.getSysAdmin("BM01")	IVR151
6	IVR151	PLAY	eval:util.getSysAdmin("BM02")	IVR152
7	IVR152	PLAY	eval:util.getSysAdmin("BM03")	IVR175

The TTS or WAV label for your broadcast message scripts should match the Broadcast Message ID (or KEY). For example, Broadcast Message BM01 should have a matching TTS labeled BM01, or a wav file named BM01.wav. The following are TTS labels from **<tenant>-tts.csv**:

	A	B	C
1	LABEL	en-US	fr-CA
4	BM01	Broadcast Message 1. You can enable and disable broadcast messages using the IVR Administration line.	Message de diffusion 1. Vous pouvez activer et désactiver ce message à l'aide de la ligne d'administration IVR.
5	BM02	Broadcast message 2. While you test out some of the features of this sample application, you can always press star to repeat a menu, or press 9 to return to the previous menu.	Il s'agit du message de diffusion 2. Pendant que vous testez certaines des fonctionnalités de cet exemple d'application, vous pouvez toujours appuyer sur étoile pour répéter un menu ou appuyer sur 9 pour revenir au menu précédent.
6	BM03	Broadcast Message 3. You should also know that the default invalid and timeout retries is set to 2.	This is Broadcast Message 3. You should also know that the default invalid and timeout retries is set to 2.

Dynamic Menus

Dynamic Menus can be used in a MENU block with the eval type to call the **util.getDynamicPrompts()** and **util.getDynamicMenu()** pre-built functions. The following is an example dynamic menu set in **<tenant>-callflow.csv**:

	A	B	C	D
1	BLOCK	TYPE	PARAM	ACTION
10	IVR300	MENU	eval:util.getDynamicPrompt("tts","DM100")	eval:util.getDynamicMenu("DM100")

util.getDynamicPrompts(type, id) – where **type** is the application script type to play (see PLAY block in Chapter 5), and **id** is the Dynamic Menu ID to reference in **<tenant>-sysAdmin.csv**.

util.getDynamicMenu(id) – where **id** is the Dynamic Menu ID to reference in **<tenant>-sysAdmin.csv**.

The TTS or WAV labels for your Dynamic Menu should have an introductory script plus a corresponding script for each dynamic menu option. For example, Dynamic Menu DM100 with 5 dynamic options (1-4, and 0), should have matching TTS entries labeled DM100 (introductory menu script), DM100-1 (script for option 1), DM100-2 (script for option 2), DM100-3 (script for option 3), DM100-4 (script for option 4), and DM100-0 (script for option 0). The **util.getDynamicPrompts()** pre-built function will play always play the introductory script and only the scripts where the corresponding option is enabled.

	A	B	C
1	LABEL	en-US	fr-CA
8	DM100	This is a Dynamic Menu. You can enable and disable menu options using the IVR Administration Line	Il s'agit d'un menu dynamique. Vous pouvez activer et désactiver les options de menu à l'aide de la ligne d'administration IVR.
9	DM100-1	To play a numerical value in different formats, press 1	Pour lire une valeur numérique dans différents formats, appuyez sur 1
10	DM100-2	To play a date, press 2	Pour écouter une date, appuyez sur 2
11	DM100-3	To play a telephone number, press 3	Pour écouter un numéro de téléphone, appuyez sur 3
12	DM100-4	To play an address, press 4	Pour lire une adresse, appuyez sur 4
13	DM100-0	To speak with a customer service representative, press 0	Pour parler à un représentant du service client, appuyez sur 0

A Dynamic Menu with all options enabled in <tenant>-sysAdmin.csv:

	A	B
1	KEY	VALUE
7	DM100	1:IVR310|2:IVR320|3:IVR330|4:IVR340|0:PQ100

A Dynamic Menu option is disabled by simply marking the option with an ‘x’ as shown below. This can be accomplished manually or via the IVR System Administration module

	A	B
1	KEY	VALUE
7	DM100	1:IVR310|2:IVR320|3x:IVR330|4x:IVR340|0:PQ100

Chapter 7: Utility Functions

Queue Metrics

The CompuCom RDF leverages Amazon Connect's ***get_current_metric_data()*** API to obtain Queue metrics such as AGENTS_ONLINE, AGENTS_STAFFED, AGENTS_AVAILABLE, SLOTS_AVAILABLE, CONTACTS_IN_QUEUE, and OLDEST_CONTACT_AGE. This allows the CompuCom RDF to implement custom routing logic and calculations using these queue metrics. Review the following for more details:

https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/connect/client/get_current_metric_data.html

To overcome Amazon Connect's API throttling limits, exponential back off and jitter was implemented to retry failed API calls when there are too many concurrent requests. Review the following for more details:

https://aws.amazon.com/blogs/architecture/exponential-backoff-and-jitter/

The following are pre-built functions that can be called within an EVAL Block:

queues.agentsOnline(): CHECK AGENTS ONLINE

queues.agentsStaffed(): CHECK AGENTS STAFFED

queues.agentsAvailable(): CHECK AGENTS AVAILABLE

queues.slotsAvailable(): CHECK SLOTS AVAILABLE

queues.withinQueueCapacity(threshold): CHECK QUEUE CAPACITY

queues.withinWaitCapacity(threshold): CHECK WAIT THRESHOLD

DynamoDB

The CompuCom RDF can replace CSV configuration files with DynamoDB tables. However, updating and maintaining DB tables adds a layer of complexity when importing and exporting data. Therefore, CSV files were used to implement all configuration settings. To utilize a DynamoDB table, follow the instructions in Chapter 2 to ensure the Cross-Account IAM role of your tenant contains the proper permissions to access DynamoDB tables.

From CompuCom RDF, you can call the ***readDB(tableName, Key, Type)*** built-in function to read the contents of a simple DynamoDB table with Key/Value pairs, where *tableName* is the DynamoDB table name, *Key* is the search key in the

table, and *Type* defines the data type of the value stored in the table (i.e. “S” for String or “BOOL” for Boolean). Review the following for more information on DynamoDB Data Types:

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.NamingRulesDataTypes.html#HowItWorks.DataTypes

Clear Previous Menu

The CompuCom RDF tracks all the Menu’s that a caller accesses within a *MenuTracker[]* List. This List is mainly used to provide the ability for a caller to return to a previous menu. Previous Menu’s can span across different sub call flows. There may be times in your call flow design that requires a particular menu to be removed from the MenuTracker. For example, a confirmation Menu might not be appropriate in some cases to return. In this case, you can call ***util.clearPreviousMenu(session)*** built-in function to remove the most recent Menu from the MenuTracker[] List. It’s also important to note that the MenuTracker will not add a Menu that already exists in the MenuTracker[] List. This is to avoid call flow designs where callers are explicitly sent back to a Menu that has already been accessed.

Contact Dispositions

The CompuCom RDF tracks a caller’s disposition as they traverse through a call flow using the Block ID’s created in your application. It will also track invalid and timeout outcomes. A **contactDisposition** contact attribute is used in Amazon Connect to save the caller dispositions within the CTR. An example is shown below:

contactDisposition	IVR100,IVR125,IVR150,IVR151,IVR152,IVR175,IVR200,IVR300,IVR200,IVR40

You can also create your own custom caller dispositions to get added to the contactDispostion contact attribute by calling the following pre-built function:

session['contactDisposition'] = util.addDisposition(session['contactDisposition'], new_dispostion)

Where, new_disposition is the new caller disposition to add to the contactDisposition contact attribute.

Chapter 8: Helpful Hints

This chapter provides helpful hints on some of the underlying components of the CompuCom RDF and additional configuration options.

RDF Lambda Cache

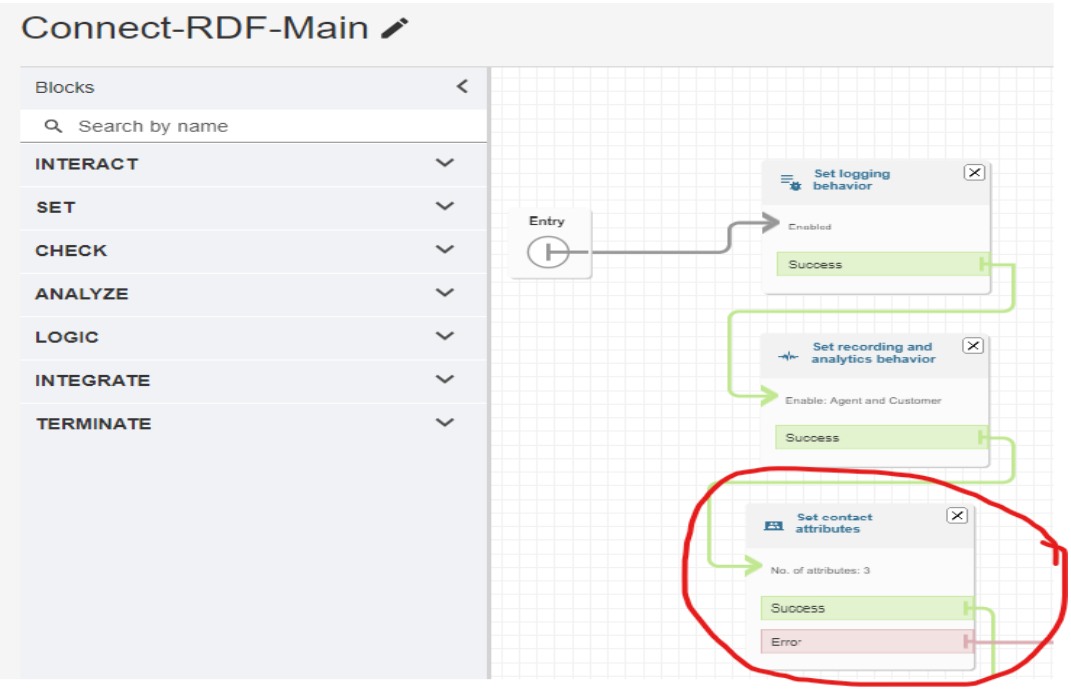
The CompuCom RDF uses temporary storage to save application configuration files during AWS Lambda runtime to avoid reading files from S3 at every invocation. Each Lambda execution environment provides between 512 MB and 10,240 MB, in 1-MB increments of disk space in the /tmp directory. The /tmp directory content is preserved for the lifetime of the execution environment, providing a transient cache that can be used for multiple invocations to optimize performance. The CompuCom RDF checks if the cache has the data required, otherwise it will fetch the required application configuration files from S3 and saves them within the /tmp directory for subsequent invocations. The /tmp directory is equivalent to a local hard disk, providing fast throughput. Operations like makedirs, listdir, etc. on this folder using the 'os' Python package can be performed just like you'd perform on a local hard disk.

At the start of every call, the CompuCom RDF will check to see if an updated configuration file exists in S3. If a new version of an application configuration file exists, the /tmp cache directory will be updated with the new configuration file. Otherwise, it will continue to use the configuration file saved in cache.

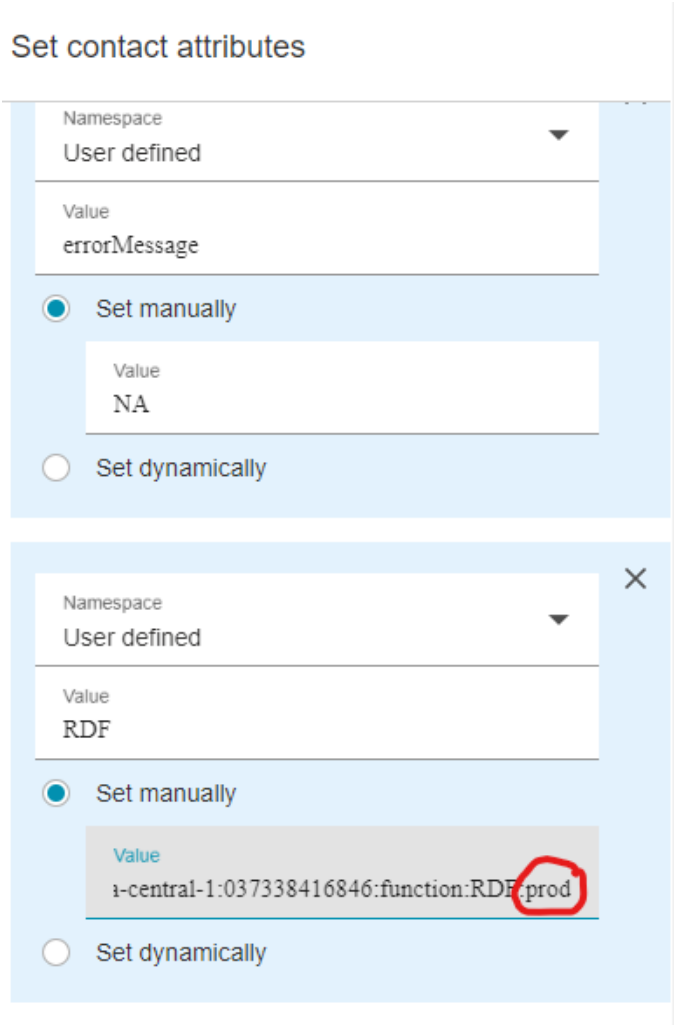
It is important to note that the /tmp directory is private to each Lambda instance/execution context and it's not shared among all Lambda instances. That means if a request is handled by a different execution context, the data won't be present there and will need to be fetched in S3.

CompuCom RDF – Pointing to DEV/UAT/PROD

The CompuCom RDF can be pointed to different instances for DEV, UAT and PROD by updating the third block of the **Connect-RDF-Main** Contact Flow:



Open the Set contact attributes block in Connect-RDF-Main, and update the RDF attribute to point to either dev, uat, or prod, as shown below:



Quick Connects

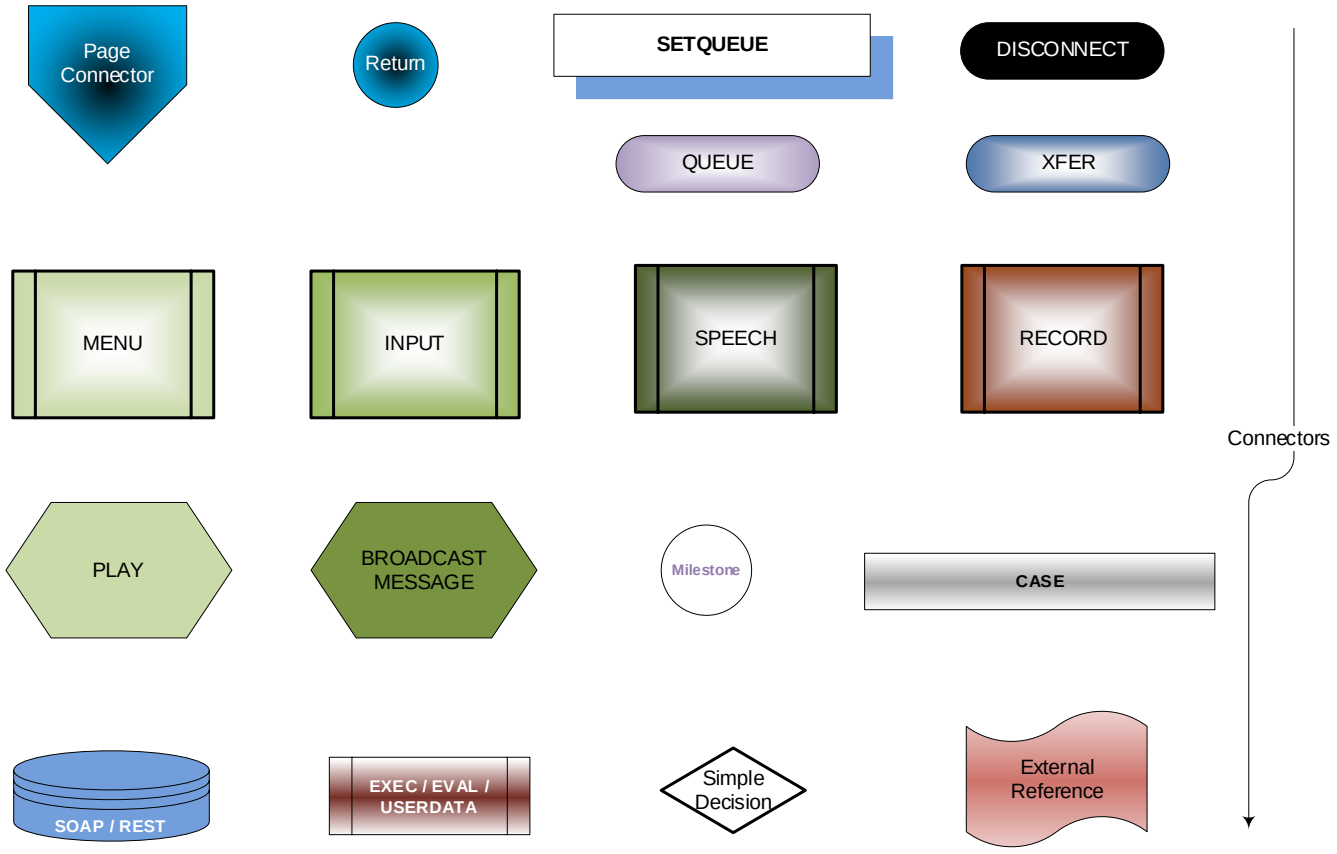
Quick Connects can be used to allow agents to transfer callers to another queue. When a Quick Connect is answered by the CompuCom RDF, the **Connect-RDF-Agent** Contact Flow takes control of the application. The first block that a Quick Connect call will execute is the Block ID defined for **"globalMaxtriesAction"** in config.csv. The globalMaxtriesAction block ID is used since it defines the beginning of the pre-queue call flow. To determine if your application was reached by a Quick Connect, check for **session['function'] == 'agent'**. If this condition is true, you can design your application to interact with a Quick Connect contact. An example is shown below that allows an agent to send a caller back to the Pre-Queue flow:

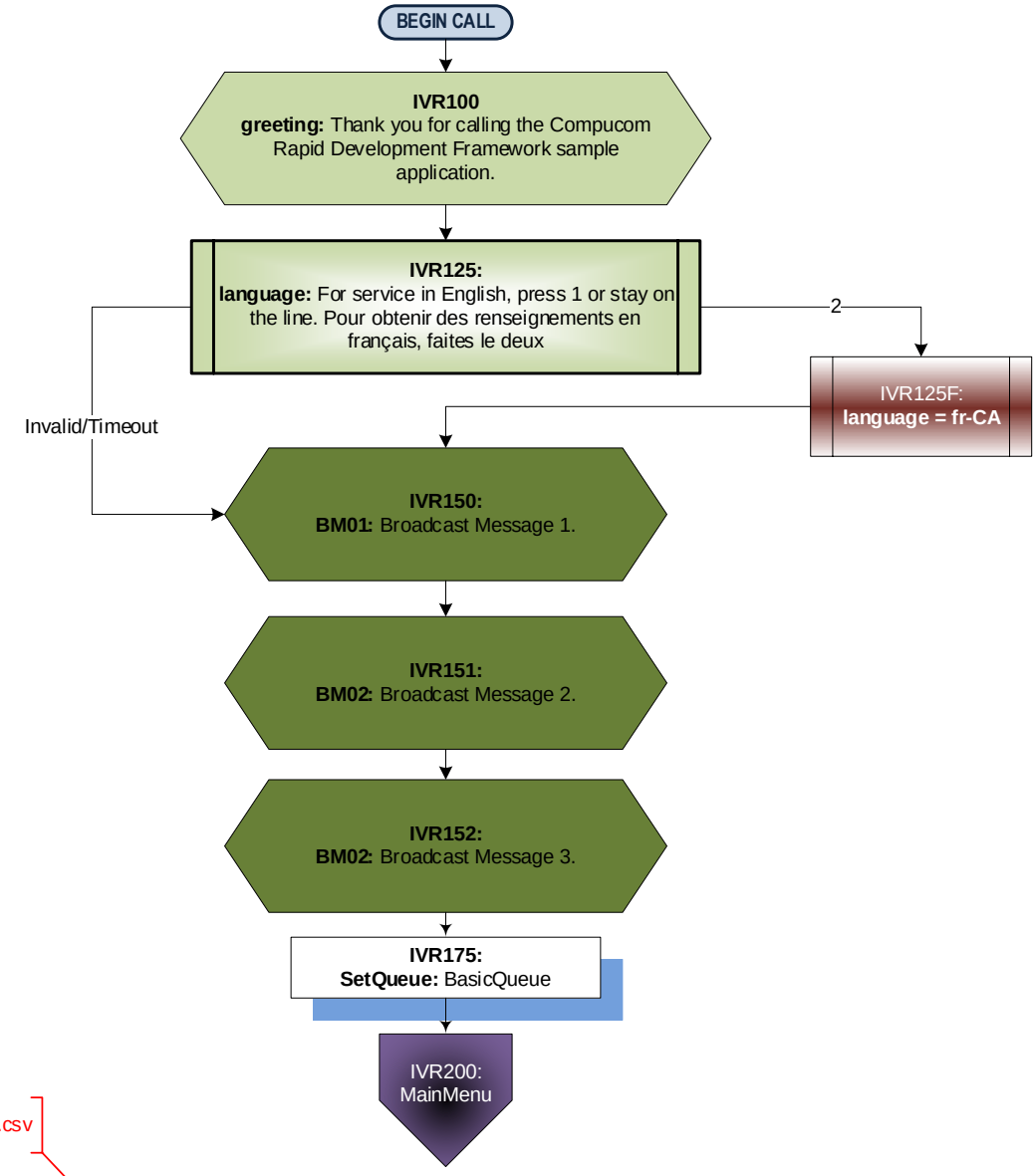
	A	B	C	D
1	BLOCK	TYPE	PARAM	ACTION
25	PQ100	EXEC	session['queuemetrics'] = queues.getQueueMetrics(session['q	PQ110
26	PQ110	EVAL	session['function']=='agent'	true:PQ125|false:PQ175
27	PQ125	EVAL	session['queuemetrics']['agentsOnline'] > 0	true:PQ150|false:PQ175
28	PQ150	PLAY	text:The queue is open and staffed. Please transfer the caller r	PQ175

Chapter 9: Sample Application

This chapter provides a walkthrough of the sample application that was deployed in **Chapter 3 – Hello World!** For additional clarity, the call flow diagrams include the corresponding configuration entries in ***sample-callflow.csv.***

The shapes used in the call flow diagrams are as follows:



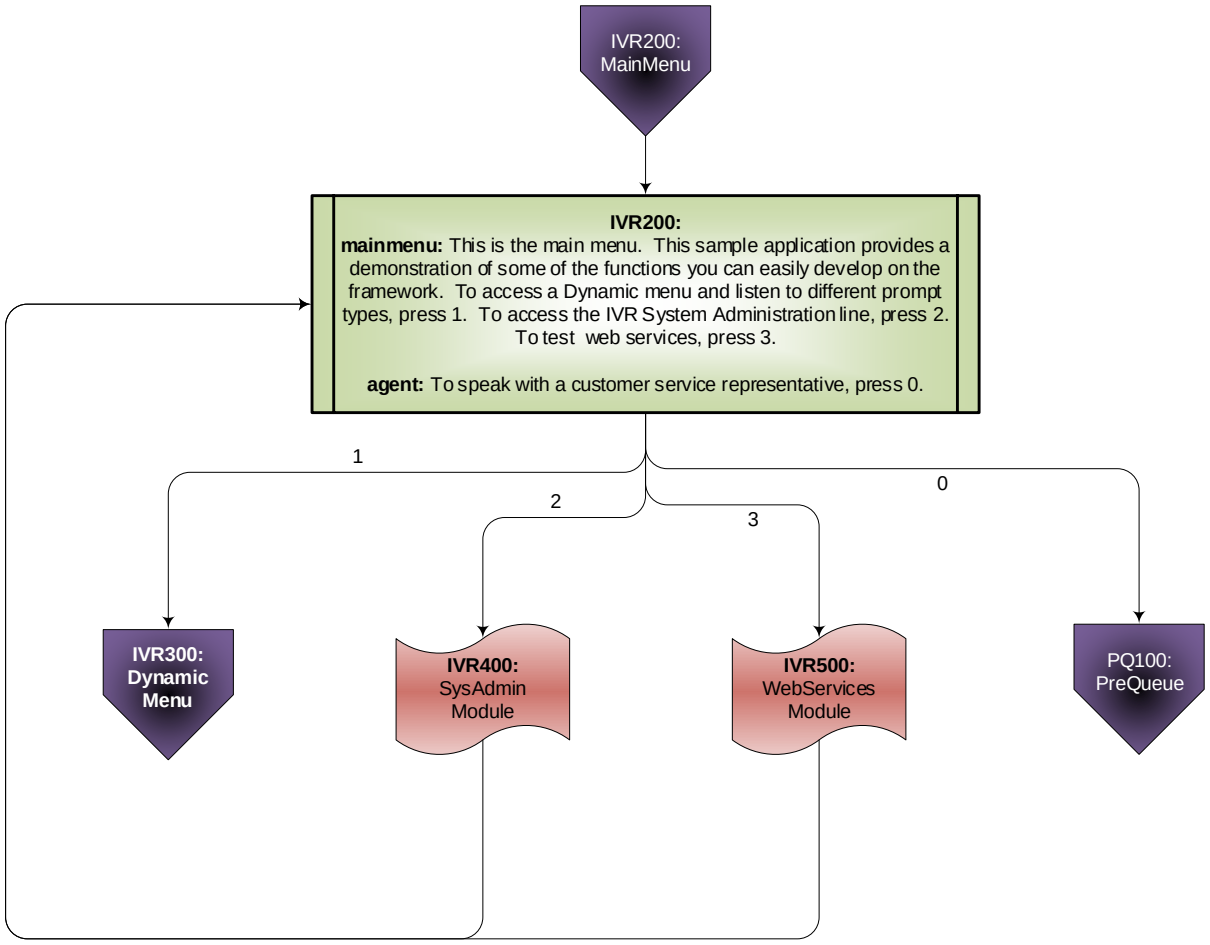


sample-callflow.csv

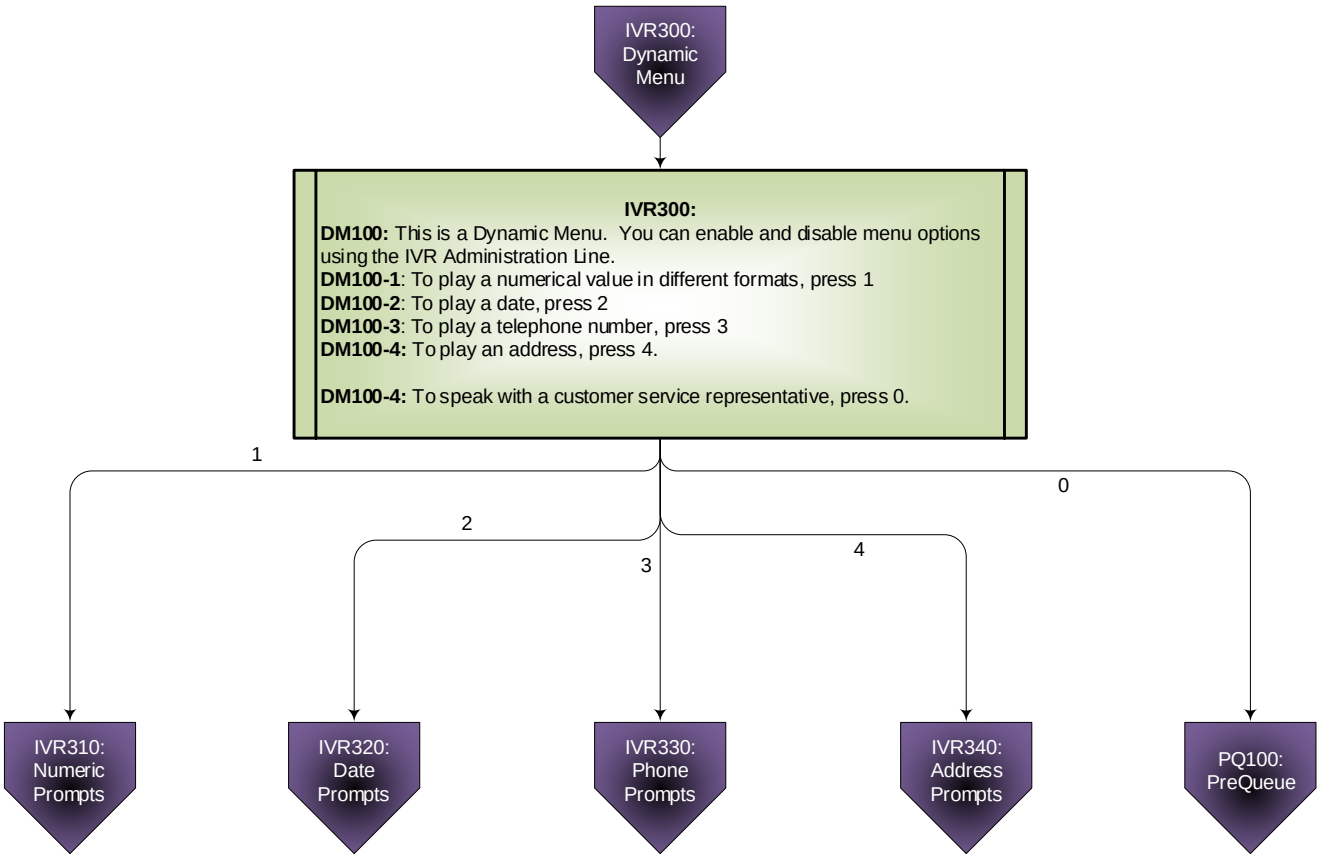
	A	B	C	D
1	BLOCK	TYPE	PARAM	ACTION
2	IVR100	PLAY	wav:greeting	IVR125
3	IVR125	MENU	wav:language	2:IVR125F|9:IVR150|*:IVR150|timeout:IVR150|invalid:IVR150
4	IVR125F	LANGUAGE	fr-CA	IVR150
5	IVR150	PLAY	eval:util.getSysAdmin("BM01")	IVR151
6	IVR151	PLAY	eval:util.getSysAdmin("BM02")	IVR152
7	IVR152	PLAY	eval:util.getSysAdmin("BM03")	IVR175
8	IVR175	SETQUEUE	en-US:BasicQueue|fr-CA:BasicQueue	IVR200

sample-sysadmin.csv

	A	B
1	KEY	VALUE
4	BM01	tts:BM01
5	BM02	tts:BM02
6	BM03	tts:BM03



	A	B	C	D
1	BLOCK	TYPE	PARAM	ACTION
54	IVR200	MENU	tts:mainmenu|tts:agent	1:IVR300|2:IVR400|3:IVR500|0:PQ100
55	IVR300	MENU	eval:util.getDynamicPrompt("tts","DM100")	eval:util.getDynamicMenu("DM100")
56	IVR400	SUB	sysadmin-callflow	IVR200
57	IVR500	SUB	ws-callflow	IVR200
58	PQ100	EXEC	session['queuemetrics'] = queues.getQueueMetrics(session['q	PQ110



From sample-callflow.csv

	A	B	C	D
1	BLOCK	TYPE	PARAM	ACTION
55	IVR300	MENU	eval:util.getDynamicPrompt("tts","DM100")	eval:util.getDynamicMenu("DM100")

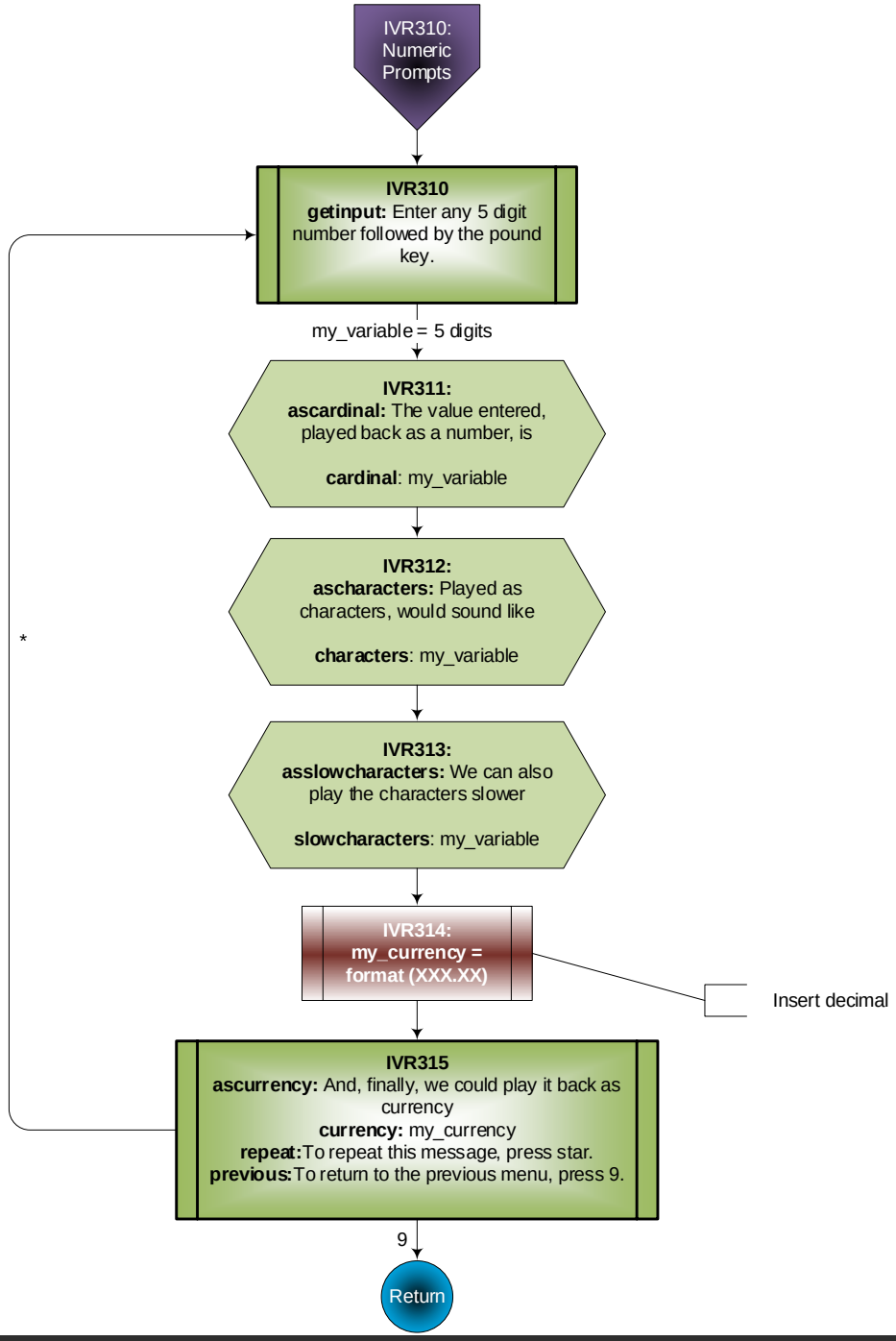
sample-sysadmin.csv – all options enabled

	A	B
1	KEY	VALUE
7	DM100	1:IVR310|2:IVR320|3:IVR330|4:IVR340|0:PQ100

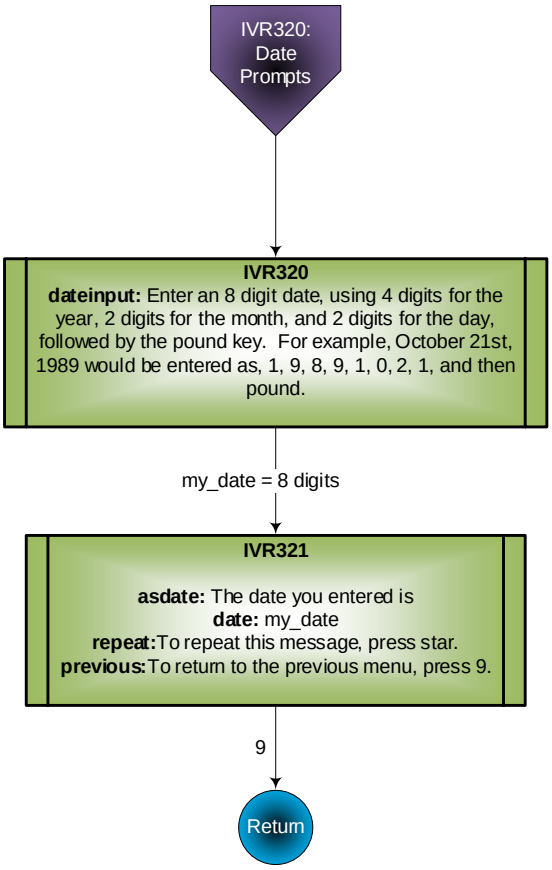
sample-sysadmin.csv – options 3 & 4 disabled

	A	B
1	KEY	VALUE
7	DM100	1:IVR310|2:IVR320|3x:IVR330|4x:IVR340|0:PQ100

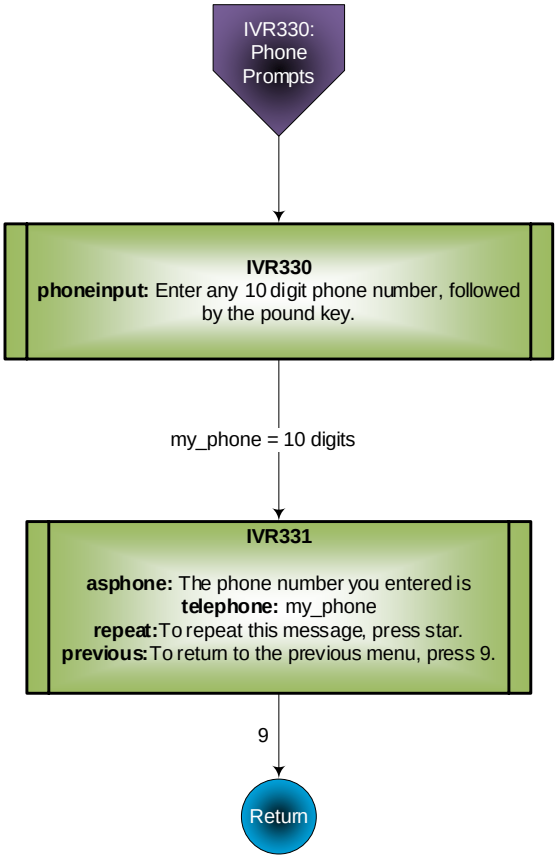
NUMERICPROMPTS



	A	B	C	D
1	BLOCK	TYPE	PARAM	ACTION
11	IVR310	INPUT	tts:getinput	var:my_variable|length:5|action:IVR311|seconds:3
12	IVR311	PLAY	tts:ascardinal|cardinal:session["my_variable"]	IVR312
13	IVR312	PLAY	tts:ascharacters|characters:session["my_variable"]	IVR313
14	IVR313	PLAY	tts:asslowcharacters|slowcharacters:session["my_variable"]	IVR314
15	IVR314	EXEC	session['my_currency'] = session['my_variable'][:3] + '.' + session['my_variable'][3:]	IVR315
16	IVR315	MENU	tts:ascurrency|currency:session["my_currency"]|tts:repeat|tts:previous	*:IVR311

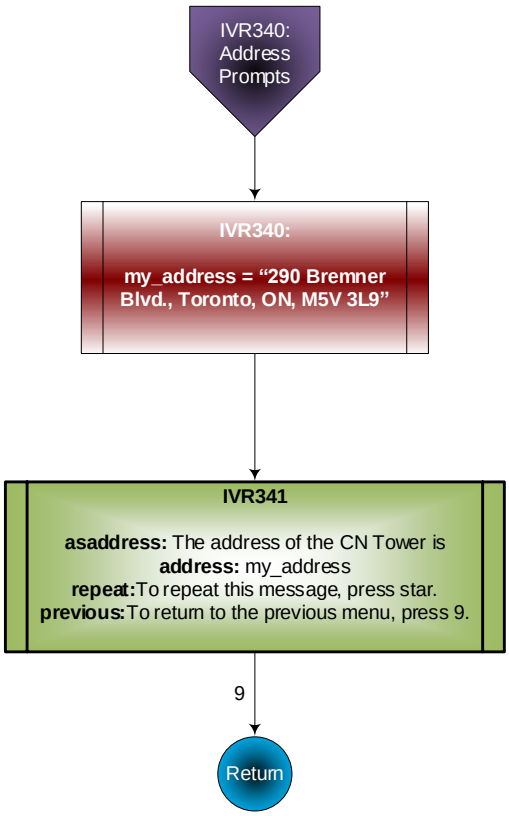


	A	B	C	D
1	BLOCK	TYPE	PARAM	ACTION
17	IVR320	INPUT	tts:dateinput	var:my_date|length:8|action:IVR321|seconds:3
18	IVR321	MENU	tts:asdate|date:session["my_date"]|tts:repeat|tts:previous	

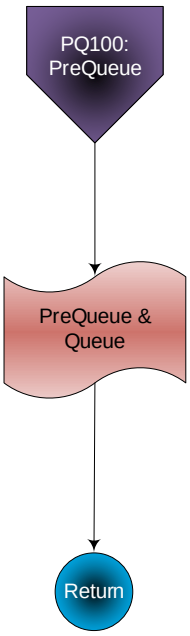


	A	B	C	D
1	BLOCK	TYPE	PARAM	ACTION
19	IVR330	INPUT	tts:phoneinput	var:my_phone|length:10|action:IVR331|seconds:3
20	IVR331	MENU	tts:asphone|telephone:session["my_phone"]|tts:repeat|tts:previous	

ADDRESSPROMPTS



	A	B	C	D
1	BLOCK	TYPE	PARAM	ACTION
21	IVR340	EXEC	session["my_address"] = "290 Bremner Blvd, Toronto, ON M5V 3L9"	IVR341
22	IVR341	MENU	tts:asaddress|address:session["my_address"]|tts:repeat|tts:previous	


PQ100:
PreQueue
PreQueue &
Queue
Return

Chapter 10: Support

For Additional Support, contact CompuCom at:

Julius Malixi
Lead IVR Consultant
CompuCom Canada
(416) 848-0094
julius.malixi@compucom.com