Brekeke PBX

Version 3

IVR Developer's Guide

Brekeke Software, Inc.

Version

Brekeke PBX version 3 IVR Script Developer's Guide

Disclaimer

Brekeke Software, Inc. reserves the right to change any information found in this document without any written notice to the user.

1. Introduction

This document explains the configuration of Brekeke PBX's IVR. Using Brekeke PBX with IVR option, system administrator can customize IVR behavior however they like, either by designing IVR flows with default [IVR Designer] modules or by creating their own JavaScript modules. To use IVR Script or the IVR Designer feature, IVR option is required to be added Brekeke PBX license.

2. Setting Up IVR Extensions

2.1. IVR Flow Extension

Name	Default Value	Description
Extension		Extension Number
Type	Flow	IVR type
Description		Extension description
Flow Name		The name of the flow that is created in [IVR Designer]
Properties		Multiple properties can be set. The setting in this field will overwrite the properties value of modules in this flow. Format: <module_a>.<property1>=<value> <module_a>.<property2>=<value> <module_b>.<property1>=<value> <module_b>.<property2>=<value> For example: In the flow, there is a "prompt" module named prom. Its property named "stop_by_dtmf" can be set from the flow extension [Parameter] field as: `prom.stop_by_dtmf=yes`

Name	Default Value	Description
Auto Answer	yes	Determines whether or not an extension should answer incoming calls. If this is set to no, the Answer module must be used in the flow.
Sound Files		Upload a customized sound file

2.1.1. IVR Flow Sample

The example below shows the steps to create an IVR flow with default module templates from [IVR Designer], as well as how to send calls through the created IVR flow.

Create an IVR flow to make Brekeke PBX perform the following actions:

1) Send 180 reply to the incoming call.
2) Answer the call.
3) Play voice prompt asking caller to input transfer extension, then wait for input.
   If there is no input within the DTMF timeout, play voice prompt again;
   otherwise, go to next module.
4) Transfer the call to the input user extension.
5) If the call is answered within transfer timeout, the flow ends.
6) If not, the flow will go back to Step 3.

2.1.2. Creating an IVR Flow

1) From Brekeke PBX Admintool > **[IVR Designer]**, click on setting icon and choose **[New Flow]** to create a new flow named "**test.**"
   Flow workspace will be opened with name tab "test."
2) Click on **[Basic]** folder on left side of flow workspace to show default IVR module templates.
3) Select and drag template **[Response 18x]** to flow "test" workspace.
   - Input a module name, such as "**18x**", and click [OK].
   - Double-click on module "18x" and select [Properties] tab and set:
     [Response Code] 180
     [Add SDP] yes
   - Click [OK].
4) Select and drag template **[Answer]** to flow "test" workspace.
   - Input a module name, such as "**ans,**" and click [OK].

5) Connect modules “18x” and “ans.”

   Connect two modules: Drag line from the bottom of one module to the top of the other.

   Disconnect two modules: Click on the bottom of the module where connection line starts.

6) Select and drag template **[DTMF Input]** to flow “test” workspace.
   - Input a module name, such as “**input**” and click [OK].
   - Double-click on module “input,” select [Properties] tab and set:

     [Prompt] {inputprompt}

     Uploaded voice prompt named as “inputprompt” from [Voice Prompts]. For details about prompt settings, check method “PlayAndInput” in the following section.

     [Max DTMF Length] 10

     A number greater than or equal to the number of Brekeke PBX user extension digits

     [Time out] 10,000

     Time length waiting for input in ms

     [Terminator DTMF] #

     Character to stop detecting DTMF input
   - Click [OK].

7) Connect modules “ans” and “input.”

8) Select and drag template **[Transfer]** to flow “test” workspace.
   - Input a module name, such as “**xfer**,” and click [OK].
   - Double-click on module “xfer,” select [Properties] tab and set:

     [Number] [input]

     Get the input extension number from last module “input.” For details about setting and format, check method “replaceWithPropertiesAndResults” in the following section.

     [Time out] <input.timeout>

     Time out for transfer in ms. Get last module “input” timeout property setting. For details about setting and format, check method “replaceWithPropertiesAndResults” in the following section.
   - Click [OK].

9) Connect modules “input” and “xfer.”

10) Select and drag template **[Disconnect]** to flow “test” workspace.
    - Input a module name, such as “**end**” and click [OK].

11) Connect module “xfer” -> “Succeeded” output with module “end.”

12) Connect module “xfer” -> “Failed” output with module “input.”

13) Save the Flow “test” with either of the following options:
    - Right-click in flow workspace and choose “Save Flow.”
    - Click on setting icon and choose “Save Flow.”

2.1.3. Creating an IVR Flow Extension

In this part, an IVR flow extension is set up and calls the flow "test" created above.

1) From Brekeke PBX Admintool > [Extensions] -> [IVR], create a new IVR extension.
2) Set the following fields:

   [Extension] 2020

   [Type] Flow

   [Flow Name] test

   [Auto Answer] no

   If set to "yes," modules "Response 18x" and "Answer" are not needed in flow.
3) Save the settings.

2.1.4. Making a Call

1) Call to extension 2020.
2) Brekeke PBX will act as designed in Section "IVR Flow Sample."

2.2. IVR Script Extension

Name	Default Value	Description
Extension		Extension Number
Type	Script	IVR type
Description		Extension description
Note		Note name in which IVR JavaScript function is saved
Function		The function name saved in Note
Parameter		Only one parameter can be set in this field. The value set in this field can be passed to function in Notes as the second parameter in function, such as: `function guidance( ivr, param ) { // actions }`

Name	Default Value	Description
Auto Answer	yes	Determines whether extension answers the incoming call. If set to “no,” method answer() must be used in JavaScript
Sound Files		Upload customized sound file.

2.2.1. Example

1) From Brekeke PBX Admintool > [Options] > [Notes], create a Note named "test."
2) Put the following sample JavaScript in the text field of Note "test" and save it:

```
function f_1(ivr){
 ivr.play("(C:\\<path_to_the_sound_file>\\<file_name>.ul)");
 ivr.dropcall();
 }
```

3) From Brekeke PBX Admintool > [Extensions] > [IVR], create a Script type IVR extension, such as 2021.
4) Set IVR Script extension 2021 as follows:
   Note: Test (the Note created above)
   Function: f_1 (function name in Note)
   Parameter: No setting for this example
   Auto Answer: Yes
5) Making a call to IVR script extension
   Brekeke PBX will look for and play the sound file at the location defined in ivr.play() in step 2, then drop the call.

3. Classes

3.1. IVR Class

This class implements a basic IVR feature. An instance of this class will be created when IVR starts and will be passed to an IVR Script function as the first argument.

The ivr methods can be called in format: ivr.<method_name>, such as ivr.answer();

3.1.1. Methods

void answer()

Description: Answer the call

This method is needed when extension [Auto Answer] field is set to ”no.”

void cancelTransfer()

Description: Back to the original party

String clearDTMFBuffer()

Description: Get DTMF signals from the signal buffer

Return: DTMF characters

boolean connected()

Description: Check if the session is connected

Return: “true” if the session has been established; “false” if not connected

void dropcall()

Description: Drop current call

void exec(String note, String function, String param)

Description: Execute another JavaScript function in another note

Parameters:

note: Note name

function: Function name

param: Only one parameter

ConnectionManager getDb(String name)

Description: Return the ConnectionManager object that is for JDBC connection pool

For details, please refer to ConnectionManager class.

Return: The ConnectionManager object

FlowRunner getFlowRunner()

Description: Return the FlowRunner object associated with the current IVR

For details, please refer to FlowRunner class.

Return: FlowRunner object associated with the current IVR, or “null” if the IVR type is not Flow

String getLanguage()

Description: Get a language code of the current language

Return: Current language code

"en" for English, "ja" for Japanese or another language code if you have localized Brekeke PBX into other languages.

org.apache.log4j.Logger getLogger()

Description: Return the Log4j Logger object for debug logging

Please refer to Log4j website: http://logging.apache.org/log4j/index.html. The default output will be saved in the webapps/pbx/WEB-INF/work/pbx/logs folder.

Return: Log4j Logger object

String getMyNumber()

Description: Get phone number that is used for current call

Return: The user’s phone number

Object getObject(String objectName)

Description: Get The object bound with the specified objectName in this session

Return: The object with the specified objectName

String getOtherNumber()

Description: Get phone number of the other party on current call

Return: The other party’s phone number

String getParameter()

Description: Get a parameter of this call, the parameter can be passed in format ivr<ivr_extension>*<parameter>

Return: The parameter of the call in string

Example:

If there is an IVR flow extension named 1000, the parameter 1234 can be passed in format: ivr1000*1234

The return value of `ivr.getParameter()` will be 1234.

String getProperty(String key)

Description: Retrieve a property value from Brekeke PBX

The parameters may be set at [Options] > [Advanced].

Parameters:

key: A string of a property name

Return: A string containing the property value

String getTempDir()

Description: Returns the pathname string of the temporary folder

Files under this folder will be deleted when restarting the PBX service.

Return: The temporary folder pathname

String getTenant()

Description: Returns the tenant name as string from which IVR script is called

Return: Tenant name if it is Multi-tenant Brekeke PBX; otherwise, return a hyphen: **-**

String getUserProperty(String key)

Description: Retrieve a user property value

Parameters:

key: A string of property name

Return: A string containing the property value

Boolean isMultitenant()

Description: Whether or not it is Multi-tenant Brekeke PBX

Return: “true” if Brekeke PBX is the multi-tenant edition; otherwise “false”

String play(String playlist)

Description: Play a sequence of sound files

Parameters:

playlist = *play-resource

play-resource = dtmf-character / prompt / voice-lib / ulaw-file

dtmf-character = DIGIT / "A" / "B" / "C" / "D" / "*" / "#"

prompt = "{" prompt-name "}"

voice-lib = "{" voice-lib-name ":" voice-lib-param "}"

voice-lib-name = "name" / "date" / "time" / "number"

voice-lib-param = 1*(ALPHA / DIGIT)

ulaw-file = "(" fullpath-ulaw-file ")"

Return: DTMF character, if a button was pressed

Example:

```
ivr.play("{ring}1234");
```

First play the file named “ring,” which is uploaded to Brekeke PBX from [VoicePrompts], then play 1234 as DTMF.

String play(string playlist, boolean ignoreDTMF)

Description: Play a sequence of sound files

Parameters:

playlist = *play-resource

play-resource = dtmf-character / prompt / voice-lib / ulaw-file

dtmf-character = DIGIT / "A" / "B" / "C" / "D" / "*" / "#"

prompt = "{" prompt-name "}"

voice-lib = "{" voice-lib-name ":" voice-lib-param "}"

voice-lib-name = "name" / "date" / "time" / "number"

voice-lib-param = 1*(ALPHA / DIGIT)

ulaw-file = "(" fullpath-ulaw-file ")"

ignoreDTMF: true or false; Stop playing with DTMF

Return: DTMF characters if button was pressed; always empty if “ignoreDTMF” is set to “true”

String playAndInput(String playlist, int maxDtmfLength, int timeout, String terminateDtmf, boolean removeTerm)

Description: Play a sequence of sound files and retrieve DTMF signals

Parameters:

playlist = *play-resource

play-resource = dtmf-character / prompt / voice-lib / ulaw-file

dtmf-character = DIGIT / "A" / "B" / "C" / "D" / "*" / "#"

prompt = "{" prompt-name "}"

voice-lib = "{" voice-lib-name ":" voice-lib-param "}"

voice-lib-name = "name" / "date" / "time" / "number"

voice-lib-param = 1*(ALPHA / DIGIT)

ulaw-file = "(" fullpath-ulaw-file ")"

maxDtmfLength:

Maximum length of DTMF signals

timeout:

Length of time (milliseconds) for detecting DTMF signals

terminateDtmf:

DTMF characters used to terminate to retrieve DTMF input

removeTerm:

If true, remove terminate characters from the return value

Return: Detected DTMF signals

The return value will include terminateDtmf character input if parameter “removeTerm” is set to false.

void record(String file, int timeout, String terminateDtmf)

Description: Record sound data

Parameters:

file: File name to store the recorded sound

timeout: Length of record time (milliseconds)

terminateDtmf: DTMF characters input to terminate recording

public void recordVoicemail(String user, int timeout, String terminateDtmf, Properties prop)

Description: Record sound data and store it as a voicemail file

Parameters:

user: Owner of the voicemail inbox where sound data will be stored

timeout: Length of record time (milliseconds)

terminateDtmf: DTMF characters used to terminate recording

prop: Properties object that will be stored as voicemail properties

public void recordPrompt(String lang, String name, int timeout, String terminateDtmf, String filedesc)

Description: Record sound data and store as a prompt file

Parameters:

lang: Language code ("en": English, "ja": Japanese) or “common” if language is not specified

timeout: Length of record time (milliseconds)

terminateDtmf: DTMF characters used to terminate recording

filedesc: Description of the prompt file

void response18x(int rescode)

Description: Send 18x response without SDP

This method can be called before method answer().

Parameters:

rescode: Response code; set 180 or 183

void response18x(int rescode, boolean bSDP)

Description: Send 18x response

This method can be called before method answer().

Parameters:

rescode: Response code; set 180 or 183.

bSDP: Add SDP or not to the 18x response

void setLanguage(String lang)

Description: Set a language code for current language

Parameters:

lang: Language code, such as "en," "ja" or another language code if you have localized Brekeke PBX into other languages.

void setObject(String objectName, Object anObject)

Description: Bind an object to this session with the specified objectName

Parameters:

objectName: a string represent the name of an object

anObject: an object to be bound

boolean transfer(String number, int timeout)

Description: Start attended transfer

Parameters:

number: Destination phone number

timeout: Length of time to ring the transfer recipient destination (milliseconds)

Return: “true” if connection made to transfer destination; otherwise, “false”

3.2. ConnectionManager Class

This class implements a JDBC connection pool. An instance of this class can be obtained from the IVR.getDb() method. The configuration for the JDBC connection may be set at [Options] > [Advanced], as below:

```
db1.driver=com.mysql.jdbc.Driver
db1.url=jdbc:mysql://localhost:3306/dbname?useUnicode=true&charac
terEncoding=UTF-8
db1.user=root
db1.password=root
```

Example of how to use ConnectionManager class:

```
function sampleDbAccess( ivr ){
    var db = ivr.getDb( 'db1' );
    var con = db.getConnection();
    var st;
    var rs;
    try{
        var tel = ivr.getOtherNumber();
        var sql = "SELECT member_id FROM membertalbe WHERE tel='"
+ tel + "'";
        con.setAutoCommit( true );
        st = con.createStatement();
        rs = st.executeQuery( sql );
        var memberid = "";
        if( rs.next() ){
           memberid  = rs.getString( 1 );
        }
        if( memberid.length < 10 ){ //she/he is not a member.
            ivr.dropCall();
            return;
        }
        anotherMethod( ivr, cid );
            // call another method to process something.
    }finally{
            db.close( st );
            db.close( rs );
            db.close( con );
    }
```

3.2.1. Methods

void close(java.sql.Connection con)

Description: Call the con.close() method if not null

The physical connection might not be closed and could be recycled if the given connection is a pooled connection.

Parameters:

con: JDBC connection

void close(java.sql.Statement st)

Description: Call the st.close() method if not null.

Parameters:

st: JDBC Statement object

void close(java.sql.ResultSet rs)

Description: Call the rs.close() method, if not null

Parameters:

rs: JDBC ResultSet object

java.sql.Connection getConnection()

Description: Gets a JDBC connection from the connection pool

Return: A JDBC connection

3.3. FlowRunner Class

This class provides access to the current Flow and Module. This class is only available when IVR type is “Flow.”

The FlowRunner methods can be called in format: ivr.getFlowRunner().<method_name>

Such as:

```
var fr = ivr.getFlowRunner();
fr.getResult(aModule);
```

3.3.1. Methods

Void exec(String flow, String[] params)

Description: Execute another flow

Parameters:

flow: Flow name

params: An array of parameters

String getModuleProperty(String module, String key)

Description: Retrieves a property of the module

module: Module name

key: Key to the value

Return: A property value

String getResult(String module)

Description: Retrieves a result value of the module

Parameters:

module: Module name; if null, returns the result of the current module

Return: The result value

void setModuleProperty(String module, String key, String value)

Description: Set a property of the module

Parameters:

module: Module name

key: The key to be placed into this property of the module

value: The value corresponding to the key

void setResult(String res)

Description: Set the result of the module

Parameters:

res: Result of the module

String replaceWithPropertiesAndResults(String str)

Description: Replace a string value with a module property value or results

Parameters:

str: Source string

When replacing current variable with a module result, insert module name in square brackets [].

When replacing current variable with a module property value, insert module name and property name in angle brackets <> and separate module name and property name by period "**.**"

Return: A string of replaced results

4. IVR Designer

With IVR Designer, Brekeke PBX administrators can create graphic IVR flows to perform customized IVR behavior. Either the customized IVR flows can be based on basic module templates, or Brekeke PBX administrators can create their own module templates depending on their needs.

4.1. Settings

Name	Description
New Flow	Create a new flow
Open Flows	Show flow list Select flow(s) and open them in workspace
Save Flow	Save current flow
Copy Flow	Copy current flow
Close Flow	Close current flow
Delete Flows	Show flow list Select flow(s) to delete
Flow Setting	Show current flow settings
Full Screen	Show IVR Designer workspace in full screen
Save Module Template	Save self-defined module templates
Save All	Save all unsaved data
Export	Show Module and Flow list Select and export to .bivr file
Import	Import saved flows and modules

4.2. Basic Module Templates

Clicking on the [Basic] icon in the [IVR Designer] left-hand panel will reveal a list of pre-defined module templates. Double-clicking on each basic module template will show the Module Template Setting.

Basic Module Templates	Description
Prompt	Play pre-recorded sound file(s) For details, check Method play(string playlist, boolean ignoreDTMF).
Voice Rec (File)	Play pre-recorded sound file(s) and record voice, which is stored as a file at the specified location For details, check method play(string playlist, boolean ignoreDTMF).
Voice Rec (Voicemail)	Play pre-recorded sound file(s) and record voice, which is stored as a voicemail file under the specified PBX user For details, check method recordVoicemail.
Voice Rec (Prompt)	Play pre-recorded sound file(s) and record voice, which is stored as a prompt file at the [Voice Prompt] list For details, check method recordPrompt.
Disconnect	Drop current call For details, check method dropcall().
Answer	Answer the call For details, check method answer()".
Response 18x	Send 18x response before answer the call For details, check method response18x.
DTMF Input	Play sound file and retrieve DTMF input signals For details, check method playAndInput.
Script	Execute JavaScript code
Jump to Flow	Jump to another flow
SQL Query	Query Database with SQL Statement For details, check Section "ConnectionManager class."
SOAP	SOAP Web Service
Call Transfer	Start attended/blind transfer For details, check method transfer.

Basic Module Templates	Description
Cancel Call Transfer	Cancel transfer and go back to conversation with original party For details, check method cancelTransfer.
Reject	During receiving process, reject a incoming call.
Retry Counter	Count the number of processing that go through itself and determine if the number reach a specific value that is set beforehand.
Reset Retry Counter	Clear retry counter.

4.3. CCS Module Templates (Require Contact Center license)

Basic Module Templates	Description
Call Transfer from Queue	Transfer a customer's call to another destination when the call is placed in a queue.
Position in Queue	Check an order of a call in queue.
Wait for Silence (AMD)	When an Answering Machine is detected, the system will wait to play the guidance till recording starts. System will be able to leave a voicemail when prompt is set after this module.

4.4. Create New Module Templates

Brekeke PBX administrators can create and save their own IVR module templates from [IVR Designer]. These self-defined module templates can be used in any flow in the same way as the default basic module templates.

Select and right-click on [Basic] icon and create a new folder to save self-defined module templates. Select and right-click on the new-created folder to create new module templates or edit current folder.

Module Template Settings > Basic Settings

Name	Description
Type name	Module template name
Description	Module template description
Color	Choose Module template icon color
Icon	Choose Module template icon from a list of default icons
Execute Type	The execution type of module template Options: Flow/Script
Execute Type – Flow **Flow Name**	The name of executed flow
Execute Type – Script **Script**	Text area for JavaScript code

Module Template Settings > Properties

Name	Description
Add Property	Add a property to module template
Delete Property	Delete selected property from module template
Name	Property variable name
Label	Property variable label
Type	The type of property variable Options: - Text - Prompt - Yes/No - Selection - Text Area
Variable	Default value of property variable If no default value, leave it blank.
Rule	To define what values can be set to the property; Regular expressions can be used
Comment	Property description

Module Template Settings > Jump

Name	Description
Add Jump	Add a jump condition Depending on the result of module template, add one or more jump conditions.
Delete Jump	Delete a jump condition
Matching Method	Match the module result with either exact value or a pattern defined with a regular expression Options: - Match - Regular expression

Name	Description
Modify Matching Method	If “true,” user can change matching method selection. If “false,” user cannot change the method defined in [Matching Method]. Options: - True - False
Condition Modify	If “true,” user can define their own conditions depending on their needs. If “false,” only default condition can be used and user cannot change the condition settings. Options: - True - False
Condition	Conditions of module result for jumping to the next module
Label	Condition label
Comment	Condition description

Module Template Settings > Icon Manager

Name	Description
Add Icon	Add an icon to list
Delete Icons	Delete selected icon(s) from list
Close	Close icon manager table