Integration of IVR Framework with MOTECH Platform

# Overview

The latest version of **Motech (Version 0.25)** has been developed in such a way that it integrates with IVR frameworks, which are VXML/CCXML compatible.

VXML/CCXML compatible IVR frameworks can be integrated with Motech through the IVR module provided by MOTECH. With that any task can be scheduled for providing Outbound and inbound call features.

# MOTECH IVR Architecture

Motech IVR

domain

* CallDetailRecord.java
* CallDirection.java
* CallStatus.java
* Config.java

Motech IVR Module

event

* EventParams.java
* MotechEventHandler.java

repository

CallDetailRecordDataService.java

ConfigDataService.java



service

**MOTECH IVR Module**

* OutboundCallService.java
* OutboundCallServiceImpl.java

## Package Information

**Domain**

* Contains Config class which holds IVR provider configuration, represents how the IVR module interacts with an IVR provider.
* Contains CallDetailRecord class which holds all the details regarding the IVR call such as Config object, Timestamp, Call Direction, and Call Status etc.
* Contains Java Enums as Call Status, Call Direction, HttpMethod

**Event**

* Contains final Classes as EventParams and EventSubjects which holds Possible Event payloads and event subjects.
* Contains MotechEventHandler class which listens to the ivr\_initiate\_call MotechEvent and calls initiateCall.

**Repository**

* Contains two classes CallDetailRecordDataService and ConfigDataService which contains MDS generated CallDetailRecord and ConfigDataService database queries.

**Service**

* Contains an interface OutboundCallService which is used to initiate an outgoing call and its implementation class OutboundCallServiceImpl which generates & sends an HTTP request to an IVR provider to trigger an outbound call.

# VXML/CCXML Compliant IVR Integration with MOTECH

## Voxeo IVR Integration (For Inbound Calls)

Voxeo is a generic VXML/CCXML provider. Voxeo is the most widely used international IVR service provider and Motech IVR integrates with quite well with Voxeo.

Voxeo IVR framework accept phone calls from any user, prompt for and record a code, and then depending on the user input trigger a response (maybe a call back to the Caller or a Sms to the Callers number) through the Tasks module of Motech.

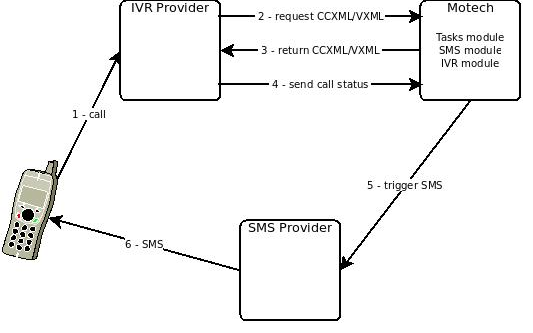
For the IVR provider to send back call status, a Config record for that IVR provider need to be created in Motech database.

Only VXML/CCXML file can be used that prompts the user for a code and then sends it to Motech Interface.

VXML/CCXML script sends user input to Motech interface (at the call status URL for the voxeo config) as a parameter using the VXML <data> element. Since user input is not a standard property, it will be added to the CallDetailRecord‘s class providerExtraData map property. Note that the call status, the caller id and the session id are sent as the callStatus, from and providerCallId parameters.

**Task is to be Created:-**

We need to create a task which will be triggered through an IVR callstatus. When the call status is ANSWERED then task is triggered to send a Call/SMS to the original caller with the desired response.



## Voxeo IVR Server Integration (For Outbound Calls)

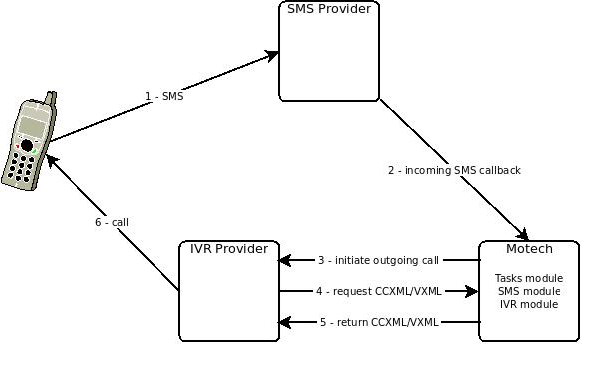
For the IVR provider to initiate a call and send back a call status to MOTECH server, we need to create a Config record for that IVR provider in the database as done for inbound call.

Note that it’s a bit different than the one created above in the IVR - Incoming Calls, as we need to tell the IVR module how to reach the IVR provider. We can achieve this by settings the outgoingCallUriTemplate and outgoingCallMethod properties.

**Task is to be Created:-**

A task can be created which upon receipt of an SMS or some other business logic can nitiate an outgoing call. In the outgoing call this task will pass a message in form of the VXML response .

During the creation of a task , the action need to be set to “initiate call”. The details of the phone number to call (the sender of the SMS) and what to say (the content of the SMS) can be specified in the action Parameters field using a map notation.



This is a short reference on how to integrate a VXML/CCXML IVR service provider with Motech IVR module. For more detailed explanation, please refer the IVR Demo document present at the location –

<http://docs.motechproject.org/en/latest/demos/ivr_demo.html>

# Non VXML/CCXML Compliant IVR Integration with MOTECH

## KooKoo IVR Server Integration (For Inbound Calls)

Kookoo is an India based IVR service provider which gives its own proprietary standards for generating XML response hence it does not support VXML/CCXML. So, Kookoo cannot be integrated with the Motech IVR module as shown above through Voxeo. As Kookoo is an India based IVR provider and provides India based Telephone numbers and good Tamil language support so we are using Kookoo as our IVR provider with Motech.

**IMP!!** As Kookoo does not support VXML/CCXML standards so a **new Motech module is developed with the custom codes that will support Kookoo’s specification and can respond to Kookoo’s HTTP request**. This new module has been named as – **KookooIVR**

All HTTP Requests from KooKoo server will be handled by this new module, hence it is indeed not dependent on Motech Base IVR module.

**KooKooIVR module integration with MOTECH**

Call KooKoo No. from any phone

**MOTECH Server**

****

HTTP Request

Motech Core Platform

**KooKoo**

**IVR Server**

Custom Module Integra

Call details like caller no. etc.

KooKoo responds accordingly

KooKooIVR

Module

**KooKoo Tunes** (XML) Commands (like <playtext>,<playaudio> etc.)

HTTP Response

**Steps on how KookooIVR module works:**

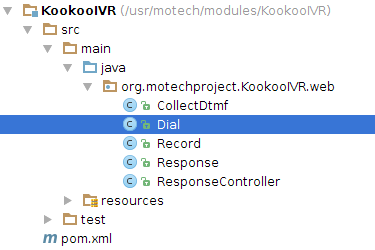
1. KooKoo server gets a call from caller.
2. KooKoo server makes a HTTP request to MOTECH Server’s KooKooIVR Module, passing along the caller detail.
3. Caller detail, caller no. etc. are stored in HTTPRequest Object.
4. Appropriate action is taken based on the request made and appropriate method is called.
5. The method KooKoo IVR module generates the XML (KooKooXML Commands) as per the request
6. These commands are passed to KooKoo Server as response.
7. KooKoo Server performs the action as the command received and accordingly interact with the user.

### How our custom ‘KooKoo IVR’ Module works in integration with MOTECH

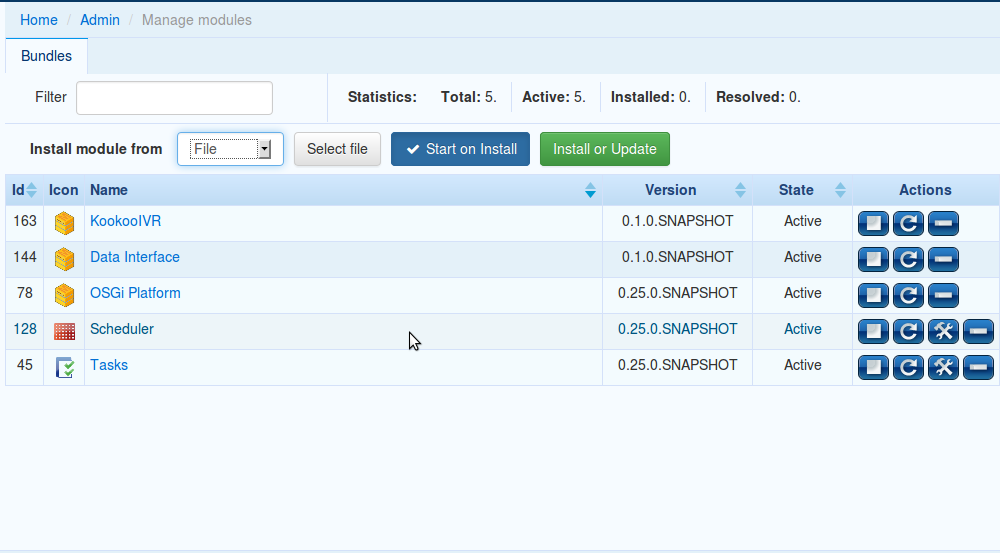
The KookooIVR module has been developed to respond to the KooKoo’s HTTPRequests. This KookooIVR module has been archived as a JAR file which then can be integrated with the core Motech application. For more information on how to develop a new module for Motech, please refer the demo available at the location: - <http://docs.motechproject.org/en/latest/demos/hello_world.html>

The JAR file provided is – **KookooIVR-0.1-SNAPSHOT.jar**

The code structure of the KookooIVR module is shown below: -



The JAR file created for KookooIVR need to be installed on Motech platform using the User Interface provided by Motech. Please find below the snapshot of the Motech UI from where any new JAR can be installed and hence integrated with Motech.



The above snapshot is taken from the “Manage Module” section of the Motech UI.

**There is an option to install a module by selecting the appropriate JAR file and then clicking on the “Install or Update” button. After successful installation of the JAR file, the new module will appear below with the list of all the modules (KookooIVR has been installed the same way).**

The KookooIVR module will be accessed by KooKoo server by hitting the URL: -

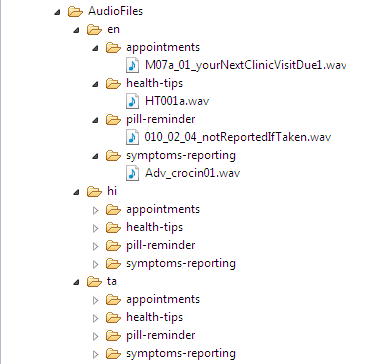
<http://localhost:8080/motech-platform-server/module/KookooIVR/getResponse>

The term “localhost” in the above URL need to be replaced with the IP address of the public server on which the Motech-KookooIVR has been deployed.

### Audio WAV Files

**The audio wav files have been used to play the speech in different languages**.

The audio wav files for required languages say English, Hindi, and Tamil need to be hosted on a public server like the below specified structure: -



**Pre-Requisites for working with the KookooIVR module**

1. An account need to be created with KooKoo for using the KooKoo IVR services.
2. The Motech-KookooIVR module need to be deployed on a public server so that KooKoo Server can access it.
3. The audio wav files for different languages also need to be deployed on a public server as KooKoo server will fetch them too.

The ‘KookooIVR module’ developed can be separately deployed as a WAR file on any public server for testing purpose. After its successful deployment, the appropriate URL need to be provided to you KooKoo account so that KooKoo server can access the application whenever a request comes.

## Kookoo IVR Server Integration (For Outbound Calls)

**MOTECH Server**

Motech Core Platform

1. HTTP Request

**KooKoo**

**IVR Server**



Custom Module Integra

caller id and callback URL etc.

3. KooKoo make Call

KooKooIVR

Module

2. KooKoo responds with a status

4. HTTP POST Request

1. Motech server Make an outbound call request to the REST API by making a HTTP request:  
     
   [**http://www.kookoo.in/outbound/outbound.php?phone\_no=090xxxxxxxxx&api\_key=XXXXXXXXXXXXXXXXXXXXXXXXX&outbound\_version=2& url=http://www.yourwebsite.com&caller\_id=9140xxxxxxx&callback\_url=http://yourwebsite.com/ outbound\_callback.php**](http://www.kookoo.in/outbound/outbound.php?phone_no=090xxxxxxxxx&api_key=XXXXXXXXXXXXXXXXXXXXXXXXX&outbound_version=2&%20url=http://www.yourwebsite.com&caller_id=9140xxxxxxx&callback_url=http://yourwebsite.com/%20outbound_callback.php)**.**

Here we are setting our caller id and also providing a callback URL.

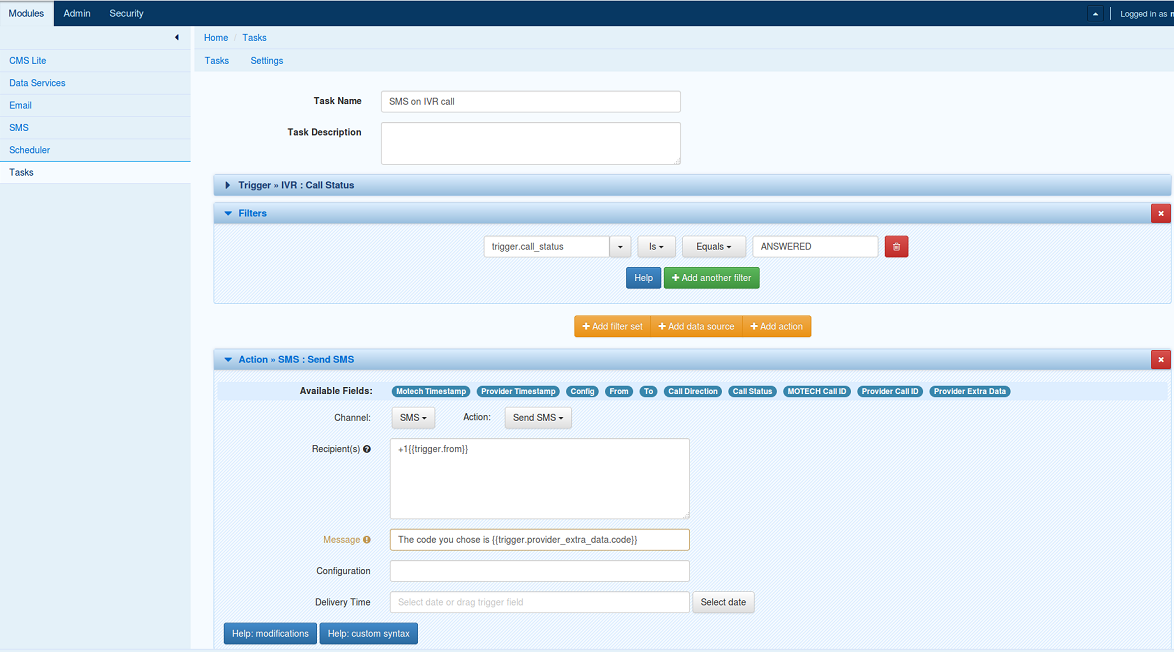
1. Once the request is made, KooKoo server responds with a status: <response><status>queued</status><message>1234552525</message></response>. The status either can be 'queued' or 'error'.
   1. If it is queued we get the unique id
   2. If it is an error, we will have to retry this call and get an error message.
2. In the background, KooKoo server will be making the call and handling it.
3. Once the call is finished KooKoo will make an HTTP POST request to Motech server callback URL:  
     
   [**http://www.yourwebsite.com/outbound\_callback.php?sid=0xfde3223234gsssss&caller\_id=9140xxxxxx&phone\_no=09xxxxxxxxx&duration=81& start\_time=xxxxxxx&status=answered**](http://www.yourwebsite.com/outbound_callback.php?sid=0xfde3223234gsssss&caller_id=9140xxxxxx&phone_no=09xxxxxxxxx&duration=81&%20start_time=xxxxxxx&status=answered)**.**

# Defining Rules and Business logics for KooKoo IVR Module

Since KookooIVR has been developed as a separate module in MOTECH so any rules or business logic that need to be implemented can be written separately in the KookooIVR module. Hence it will not have much of a connection with the existing Motech IVR module and only the data from the Motech database will be referred to trigger any call.

Let’s say, some call need to be triggered to all the mothers who have reached their 3 months of pregnancy. So depending on the details in the Motech database, developer can write their own scheduler in some other module for triggering any Phone Call through KookooIVR module. Since KookooIVR has been developed as a separate module and KooKoo response is not in VXML/CCXML standard so most of the Motech base IVR features are not used here.

Howsoever, the Tasks module of the Motech can be used to create any scheduled task. Any Task can be created from the UI provided by the Motech. As shown below in the diagram, any task can be scheduled depending on any particular event. This task will be linked to the custom rules that has been written in the KookooIVR module. Recipient’s Phone Number, call message everything can be managed from the Task module UI provided by Motech.



The above shown figure for Task creation can again be referred from the documents provided at the location -

<http://docs.motechproject.org/en/latest/demos/ivr_demo.html>

For writing any custom rules and business logic in the newly developed KookooIVR module, same approach need to be followed. Here also an object need to be created that will fetch the records from Motech Data Service and hence according to the response returned from database, respective scheduler can be written that can trigger any outbound calls as shown above through KooKoo server.

**~~ end of doc ~~**