# Software of Multi-Session Conference Implemented in CallXML

Georgi Krastev
University of Rousse
8 Studentska Str, 7017 Rousse, Bulgaria
+359 82 888 672
GKrastev@ecs.ru.acad.bg

Margarita Teodosieva
University of Rousse
8 Studentska Str, 7017 Rousse, Bulgaria
+359 82 888 464
mst_ru_bg@yahoo.com

Stoyanka Smrikarova
University of Rousse
8 Studentska Str, 7017 Rousse, Bulgaria
+359 82 888 748
SSmrikarova@ecs.ru.acad.bg

## ABSTRACT
This paper describes a developed application for carrying out a multi-session conference based on wireless communication, implemented in CallXML. To make the application more effective a separate document (code) has been added which checks the communication channel. After the check it informs about the current state of the channel which determines whether the conference will be realized.

## Keywords
Multi-session conference, CallXML

## 1. INTRODUCTION
At the current stage of development Internet and the telephone networks are converging. Today the telephone network [2] turns more and more into a programming environment and in the near future it will lead to changing our concepts for the abilities of Internet. The research and development in the field of multimedia conference connections are especially topical.

The generalized flowchart of the developed application for multi-session conference [1, 3], i. e. a conference with the use of which more than three people can converse in real time, is given in figure 1. A separate document performs a check of the communication channel. After the check it informs about the current state of the channel, which determines whether the conference will be realized.

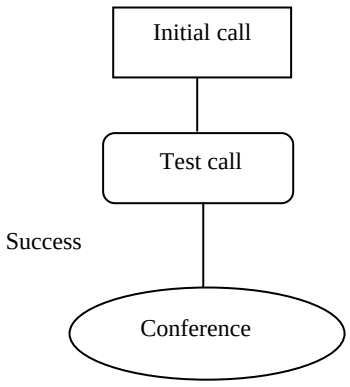


**Figure 1. Generalized flowchart**

In the first document “Initial call” a session is established which is connected to another separate session of the document “Test call” by giving the number which should be dialed. In reality before the necessary conference connection is established a conference between the two sessions of these two documents is established. In fact this number is the number of the actual application, which opens the conference connection called “Conference”. The code “Test call” checks the communications channel and returns to “Initial call” a report about the state of this channel. Depending on the reply “Initial call” generates different states.

## 2. CREATING “INITIAL CALL”
In order to start a new session an outgoing call is initiated and the call state is announced. In order to save the identifier of the current session the variable session.ID is used. The new session will have its own identifier. It is necessary to keep the identifier of the current session and to pass the variable, which is holding it, to the new session. The variable NumToCall is also passed – this is the telephone number to be dialed.

```
<?xml version="1.0" encoding="UTF-8" ?>
<callxml version="2.0">
 <block>
  <assign var="ParentSessionID" value="$session.ID;"/>
  <assign var="NumToCall" value="1020304050"/>
 </block>
</callxml>
```

In order to start indeed a new session the *run is used. It initiates a new session and “tells” the CallXML documents to use for the new session the URL or URI, given in the value attribute. In the same way as the goto element has a file attribute: runvalue="TestCall.xml", the variable that is passed submit="*", as well as the method through which the variable is sent: method="get"; the attribute, called var, contains the identifier session.ID of the new session: var="NewSessionID". The value of this attribute is not the name of the new session, but the name of the variable, which holds the identifier of the new session.

```
<?xml version="1.0" encoding="UTF-8" ?>
<callxml version="2.0">
 <block>
  <assign var="ParentSessionID" value="$session.ID;"/>
  <assign var="NumToCall" value="1020304050"/>
  <run value="TestCall.xml" submit="*" method="get" />
  <wait value="10s"/>
```

```
  </block>
</callxml>
```

Here events are sent to the opposite side, which in fact is sending events from the new to the old session (that is why the variable ParentSessionID was created). The most important here is that the XML file will continue working even after the 'run' was launched (unlike goto, which submits the control to another block). That is why before accomplishing the outgoing connection the new session holds by playing hold music or by waiting until the call is made:

```
<?xml version="1.0" encoding="UTF-8" ?>
<callxml version="2.0">
  <block>
    <assign var="ParentSessionID" value="$session.ID;"/>
    <assign var="NumToCall" value="1020304050"/>

    <run value="TestCall.xml" submit="*" method="get"  />
    <wait value="10s"/>
    <block label="HoldMusic" repeat="3">
      <playaudio format="audio/wav"
        value="HoldMusic.wav"termdigits=""/>
      </block>
  </block>
</callxml>
```

## 3. ADDING EXTERNAL EVENTS AND HANDLING THEM

Here the external events are added, which are served using the onExternalEvent handlers.

```
<?xml version="1.0" encoding="UTF-8" ?>
<callxml version="2.0">
  <block>
    <assign var="ParentSessionID" value="$session.ID;"/>
    <assign var="NumToCall" value="1020304050"/>
    <run value="TestCall.xml" submit="*" method="get"  />
    <block label="HoldMusic" repeat="3">
      <playaudio format="audio/wav"
        value="HoldMusic.wav"
        termdigits=""/>
      <onexternalevent value="Success">
       <call value="$NumToCall;"
        maxtime="30s"/>
              <onmaxtime>
                <goto value=#test/>
       <hangup/>
      </onexternalevent>
      <onexternalevent value="Busy">
        <text>
          All lines are currently busy.
          Please hang up and try your call again later.
        </text>
        <hangup/>
      </onexternalevent>
      <onexternalevent value="TimedOut">
        <text>
          There is no answer.
          Please hang up and try your call again later.
        </text>
        <hangup/>
      </onexternalevent>
      <onexternalevent value="Error">
        <text>
          A connection can not be made at this time.
          Please hang up and try your call again later.
        </text>
        <sendemail from="MyApp@here.com"
          to="YourEmail@there.net" type="debug">
          We caught an error in our application.  Details follow...
        </sendemail>
        <hangup/>
      </onexternalevent>
    </block>
  </block>
</callxml>
```

When a "Success" state is returned the conference application is automatically dialed. In case during dial-up the line changes its state then it is impossible to connect which will activate the onmaxtime element.

## 4. CREATING A “TEST CALL”

Here the second file is created, called "TestCall.xml" – this is the file, which initiates the outgoing call and then sends a message back to the first session:

```
<?xml version="1.0" encoding="UTF-8" ?>
<callxml version="2.0">
  <block>
    <call value="$NumToCall;"
        maxtime="30s"/>
    <onanswer>
      <sendevent value="Success"
            session="$ParentSessionID;"/>
      <wait value="unlimited"/>
    </onanswer>
    <oncallfailure>
```

```
    <sendevent value="Busy"
            session="$ParentSessionID;"/>
   </oncallfailure>
   <onmaxtime>
    <sendevent value="TimedOut"
            session="$ParentSessionID;"/>
   </onmaxtime>
   <onerror>
    <sendevent value="Error"
            session="$ParentSessionID;"/>
   </onerror>
  </block>
</callxml>
```

The code dials the number and passes the attribute maxtime="30s". Then one can see the element, which sends the message between the sessions: sendevent. Here we can see how the ParentSessionID variable is used, which was created in the first script.

Using the "wait" command it is ensured that the session is held open until the opposite side hangs up. This element is active only when the opposite side answers. That is why in the first script there was the WAV file hold music. The conference connection connects the two sessions but a possible hang-up by one of the sessions should be ensured. Using the wait command an unlimited period of time is waited <wait value="Unlimited"/>.

Since using this application a conference call in real time is established the given value of the attribute maxtime="30s" is not sufficient.

# 5. CREATING A "CONFERENCE"

## 5.1 Preparing for the conference

When creating a conference it is necessary to assign a name to it, which is selected arbitrarily. In order to make the application flexible and to support simultaneous conferences numbering is used. The <inputdigits> element allows for collecting digits and writing them in the "conferenceName" variable.

```
<?xml version="1.0" encoding="UTF-8" ?>
<callxml version="2.0">
  <inputdigits
     label      = "getconid"
     value      = "enterConferenceID.wav"
     var        = "conferenceName"
     maxdigits  = "10"
     termdigits  = "#"
     cleardigits = "false"
     maxtime    = "30s"
     maxsilence  = "5s">
      <ontermdigit value="#"/>
   <onmaxsilence>
    <playaudio value="noInput.wav"/>
    <goto value="#getconid"/>
   </onmaxsilence>
   <onmaxtime/>
   <onmaxdigits/>
  </inputdigits>
  <block>
  </block>
</callxml>
```

## 5.2 Creating a Conference

After giving a name to the conference the following should be done:

- Inform the use that he/she is already in a conference call;
- Initiate the actual conference call.

By playing audio sounds the user will be prepared for the forthcoming conference. To create the conference the <createconference> tag is used filling in some of its attributes – the conference name and the variable name which will hold the identifier of this conference. The name of this conference is used to uniquely identify the conference – for example if an attempt is made to create a conference with the same name then the identifier of the already existing one will be returned. That is why we assign the above given name:

```
<?xml version="1.0" encoding="UTF-8" ?>
<callxml version="2.0">
  <inputdigits
     label      = "getconid"
     value      = "enterConferenceID.wav"
     var        = "conferenceName"
     maxdigits  = "10"
     termdigits  = "#"
     cleardigits = "false"
     maxtime    = "30s"
     maxsilence  = "5s">
   <ontermdigit value="#"/>
   <onmaxsilence>
    <playaudio value="noInput.wav"/>
    <goto value="#getconid"/>
   </onmaxsilence>
   <onmaxtime/>
   <onmaxdigits/>
  </inputdigits>
  <playaudio value="joinConference.wav"/>
  <block label="test">
   <createconference name="$conferenceName;"
var="conferenceid" />
  </block>
```

```
</callxml>
```

## 5.3 Joining a Conference

After everything described above is completed, it is necessary to add a few elements to handle given events, so that the users are able to connect after they have already left. To join the <joinconference> element will be used and some of its attributes are filled in: the conference identifier and the termdigits attribute.

```
<?xml version="1.0" encoding="UTF-8" ?>
<callxml version="2.0">
 <inputdigits
    label      = "getconid"
    value      = "enterConferenceID.wav"
    var        = "conferenceName"
    maxdigits  = "10"
    termdigits  = "#"
    cleardigits = "false"
    maxtime    = "30s"
    maxsilence  = "5s">
   <ontermdigit value="#"/>
   <onmaxsilence>
    <playaudio value="noInput.wav"/>
    <goto value="#getconid"/>
   </onmaxsilence>
   <onmaxtime/>
   <onmaxdigits/>
 </inputdigits>
 <playaudio value="joinConference.wav"/>
 <block label="test">
   <createconference name="$conferenceName;"
var="conferenceid" />
   <joinconference id="$conferenceid;" termdigits="#"/>
   <ontermdigit>
     <playaudio value="leaveConference.wav" termdigits="123"/>
     <wait value="3000s" termdigits="123"/>
     <ontermdigit value="2">
       <goto value="#test"/>
     </ontermdigit>
     <ontermdigit value="3">
       <goto value="#getconid"/>
     </ontermdigit>
   </ontermdigit>
 </block>
 <onerror>
 <sendemail from="MyApp@here.com"
   to="YourEmail@there.net" type="debug">
   We caught an error in our application.  Details follow...
 </sendemail>
 </onerror>
</callxml>
```

## 6. CONCLUSIONS

The undisputed advantage of the mobile phones is the optimization in the quality of the voice devices. The developed software allows for several lines in different sessions to connect between each other in such a way that the participants can freely talk to each other. This type of applications can be successfully applied in the field of electronic commerce and the distance technology for mobile learning.

## 7. REFERENCES


[1] CallXML description, http://www.community.voxeo.com

[2] Faynberg, Igor, Gaurge Lawrence, Hui-Lan Lu, Convergent Networks and Servers: Internet marking and PSTN, Wiley Computer Publishing, 2000.

[3] King P. et. al., Handheld Device Markup Language Specification, April 11, 2003.