**#18329** - <https://redmine.vnc.biz/issues/18329>

1. **Description**:

Implement a function to be called to invite an external user

**See** [**https://redmine.vnc.biz/projects/vnc-zimlet-chat/wiki/ExternalUser**](https://redmine.vnc.biz/projects/vnc-zimlet-chat/wiki/ExternalUser)

function inviteExternalUser($externalUserEmail, $conferenceName) {

//make GET request to create external user conference

//send Email to external user email with the external conference URL

//display successful invitation sent or display errors

}

This function will be called, when the user clicks on the AddUser button and enters the external-user-email-address.

UI: <https://redmine.vnc.biz/issues/18315>

invitees = {"internal\_invitees":["alice","bob","smith"],  
"external\_invitees":["alice1@gmail.com","bob1@gmail.com"]}

API: <https://redmine.vnc.biz/projects/vnc-zimlet-chat/wiki/ExternalUser#API>

Procedure

\*Normal user: alice@vnc.biz

\*Normal user: eve@vnc.biz

\*External User: bob@gmail.com

1. Alice and Eve are in a normal VNCtalk videoconference
2. Alice then decides to invite bob@gmail.com to join the videoconference
3. Alice clicks on the "Add users" button and writes bob's email address (bob@gmail.com)
4. VNCtalk then recognizes that bob is an external user, so it handles this differently: The VNCtalk client from Alice will perform the following steps:

1. Make GET REQUEST to *createconference.php* providing all needed parameters ([SEE API](https://redmine.vnc.biz/projects/vnc-zimlet-chat/wiki/ExternalUser#API))
2. If the request is successful send an email to *bob@gmail.com* telling him to go to this URL:<http://external.prosody-dev3.vnctalk.zimbra-vnc.de/index.php?conference=$CONFERENCEID>

1. Bob receives the email containing the URL
2. Bob calls the URL and enters his email address (*bob@gmail.com*) in the field to authenticate himself.
3. Bob will be redirected to<http://external.prosody-dev3.vnctalk.zimbra-vnc.de/conference.php>
4. Then Bob joined the VNCtalk JitsiMeet conference
5. **Analytic**
   1. **Root cause**:
   2. **Solutions**:

* separate invitation to list of internal and external users.
* get authentication token (APIKEY)
* call external user api
* send email invite for external users.
* Handle error when call API /createconference.php

1. **Implementation**
   1. **Code:**

* separate invitation to list of internal and external users:

|  |
| --- |
| self.sendGroupVideoConferenceInvitationWithExternalUser = function(invitees,room,isAcceptDefault,isConfRunning){  //Opening minichat and setting browser title  biz\_zcs\_vnc\_talk\_zimlet\_HandlerObject.setBrowserTitle();  var chat\_room = CURRENT\_ROOM\_XID = room + "@conference." + MINI\_DOMAIN ;  self.chat('groupchat', chat\_room, JappixCommon.getXIDNick(chat\_room), hex\_md5(chat\_room),null,null,isAcceptDefault);  biz\_zcs\_vnc\_talk\_zimlet\_HandlerObject.setBrowserTitle();  //Handle invitees internal  if(invitees != null && invitees.internal\_invitees.length > 0){  self.sendInvitationToJoinRoom(invitees.internal\_invitees,true);  }  //Handle external internal  console.log("Invite these external users.");  console.log(invitees.external\_invitees);  for (i = 0; i < invitees.external\_invitees.length; i++) {  self.inviteExternalUser(invitees.external\_invitees[i],room);  }  //calling function to start conferernce.  if(!isConfRunning){  biz\_zcs\_vnc\_talk\_zimlet\_HandlerObject.joinConference(room);  setTimeout(function(){  biz\_zcs\_vnc\_talk\_zimlet\_HandlerObject.setBrowserTitle()  JappixMini.addIconToMinimizedWindow(chat\_room);  },1000);  }  setTimeout(function(){  biz\_zcs\_vnc\_talk\_zimlet\_HandlerObject.setBrowserTitle();  JappixMini.switchPane();  },10000);  }; |

* get authentication token (APIKEY)

|  |
| --- |
| biz\_zcs\_vnc\_talk\_zimlet\_HandlerObject.getVNCChatAuthToken() |

* call external user api

|  |
| --- |
| self.callExternalUserAPI = function(apiFunctionName,paramString, cb) {  var MINI\_EXTERNAL\_DOMAIN = SCHEME+"://"+ MINI\_EXTERNAL\_DOMAIN\_PREFIX +"."+MINI\_DOMAIN;  var API\_URL = MINI\_EXTERNAL\_DOMAIN + "/api/" + apiFunctionName;  var xmlHttpRequest = new XMLHttpRequest();  xmlHttpRequest.open('POST', "/service/proxy?target="+AjxStringUtil.urlComponentEncode(API\_URL), true);  xmlHttpRequest.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');  xmlHttpRequest.onload = function () {  // do something to response  response = this.responseText;  //console.log(response)  cb(response);  };  xmlHttpRequest.send(paramString);  return response;  }; |

* send email invite for external users:

|  |
| --- |
| self.sendMailInvite = function(list\_user, body\_text, subject\_text) {  // send email to list user  for (var i = 0; i < list\_user.length; i++) {  var soapDoc = AjxSoapDoc.create("SendMsgRequest", "urn:zimbraMail");  var m = soapDoc.set("m");  m.setAttribute("idnt", con.jid);  var node\_e = soapDoc.set("e", null, m);  node\_e.setAttribute("p", con.jid.split('@')[0]);  node\_e.setAttribute("a", con.jid);  node\_e.setAttribute("t", "f");  node\_e.setAttribute("p", list\_user[i].split('@')[0]);  node\_e.setAttribute("a", list\_user[i]);  node\_e.setAttribute("t", "t");  var node\_s = soapDoc.set("su", subject\_text, m);  var node = soapDoc.set("mp", null, m);  node.setAttribute("ct", "text/plain");  var subnode = soapDoc.set("content", body\_text, node);  var params = {  soapDoc: soapDoc,  asyncMode: true,  callback: (new AjxCallback(this, self.\_handleSOAPResponseXML)),  errorCallback: (new AjxCallback(this, self.\_handleSOAPErrorResponseXML))  };  appCtxt.getAppController().sendRequest(params);  }  }; |

Make popup when send email successfully:

|  |
| --- |
| self.\_handleSOAPResponseXML = function(result) {  ...  //Pop up when send email successfully  var dialog = appCtxt.getMsgDialog();// get a simple message dialog  dialog.reset(); // reset the dialog  dialog.setMessage("Send email successfully !"); // set the message "info" style  dialog.popup();  }; |

* Function inviteExternalUser:

|  |
| --- |
| self.inviteExternalUser = function(externalUserEmail, conferenceName) {  var APIVERSION="v1";  var APIKEY=biz\_zcs\_vnc\_talk\_zimlet\_HandlerObject.getVNCChatAuthToken();  var CALLERFULLJID=con.fulljid;  //Set Owner on video conference  if (typeof ROOMDETAIL.chatrooms[CURRENT\_ROOM\_XID] != 'undefined' ) {  var INITIATOR = ROOMDETAIL.chatrooms[CURRENT\_ROOM\_XID][0].fulljid;  } else {  var INITIATOR = CALLERFULLJID;  }  var EXTERNALUSER=externalUserEmail;  var ROOMNAME=conferenceName;  var ROOMSUBJECT=conferenceName;  var JITSIMEETURL="https://"+ MINI\_DOMAIN + "/vnctalk-jitsi-meet/index.html";  var paramString = 'api='+ APIVERSION + '&apikey=' + APIKEY + '&initiator=' + INITIATOR + '&callerfulljid=' + CALLERFULLJID + '&extuser=' + EXTERNALUSER + '&room=' + ROOMNAME + '&subject=' + ROOMNAME + '&jitsimeet=' + JITSIMEETURL;  //Post API /createconference.php  self.callExternalUserAPI("createconference.php",paramString,function(response) {  var responseJSON = $.parseJSON(response);  //Check response error  if (responseJSON.error != null) {  console.log(responseJSON.error);  var dialog = appCtxt.getMsgDialog();// get a simple message dialog  dialog.reset(); // reset the dialog  dialog.setMessage(responseJSON.error); // set the message "info" style  dialog.popup();  return;  }  //Send email  var conferenceID = responseJSON.result;  var url = "http://external." + MINI\_DOMAIN + "/index.php?conference=" + conferenceID;  var body = "User: " + CALLERFULLJID + "invite you to video conference at link " + url;  var subject = "[" + MINI\_DOMAIN + "]" +"Invite to video conferernce";  self.sendMailInvite([externalUserEmail],body,subject);  });  } |