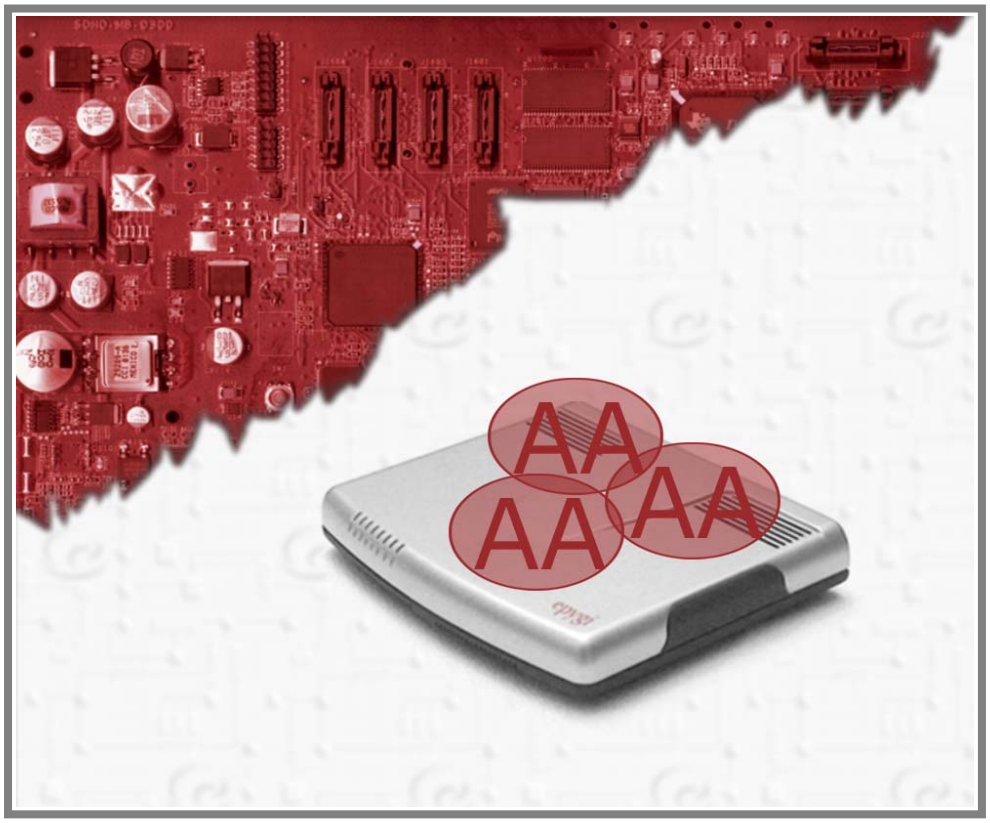
Quadro SW 3.0.x Support for Multiple Auto Attendants



Abstract: This document describes the Quadro new feature of supporting Multiple Auto Attendants (AA), explains the concept of a custom scenario for AA, and presents the subset of VoiceXML elements and attributes that can be used for writing AA custom scenarios. A sample custom scenario is also provided.

Table of Contents:

1 Introduction

This document describes the Quadro new feature of supporting Multiple Auto Attendants (AA), explains the concept of a custom scenario for AA, and presents the subset of VoiceXML elements and attributes that can be used for writing AA custom scenarios. A sample custom scenario is also provided.

2 Quadro SW 3.0.x Support for Multiple Auto Attendants

Starting with SW image 3.0.x, Quadro supports multiple Auto Attendants. The maximum number of AAs is limited to the possible number of extensions, i. e. the total number of AAs and extensions cannot exceed 70 + 1 (for “00”).

All AAs (the generated default «**00**» included) can work either by the **defaul**t scenario or a **custom** scenario.

The **default scenario** is the scenario that the Quadro AA has used so far (greeting callers, asking either to enter the party's extension or to press # for extension's directory, etc).

The **default scenario is fixed**, that is, the administrator has no option to add a new menu to the scenario, or to add one or more options to an existing one, etc. The only option the administrator has in customising the scenario is modifying the greeting message.

The **custom option for AA scenario** allows the administrator to **create a new scenario** to meet the requirements of a specific usage of the IP PBX. The administrator is allowed to change (specify) the number of menus, the number of their options, voice prompts, but NOT the outcomes (final options) of the whole scenario, i.e., whatever the scenario is, it may end by one of the following possible outcomes:

1. Verify login information,
2. Connect to an extension,
3. Make an external call,
4. Activate voice mail service.

See also Appendix 3 VoiceXML Objects for Final Options of an AA Custom Scenario.

3 Writing Scenario Files for Quadro Auto Attendants

Scenario files for Quadro Custom AAs must be written using a subset of VoiceXML 2.0 elements (for the complete list of elements and associated attributes supported by Quadro SW 3.0.x, see Voice XML Elements Supported by Quadro SW 3.0.x or Voice XML Elements Supported by Quadro SW 3.0.x (with Descriptions and Samples)).

Quadro uses the subset of VoiceXML 2.0 to create audio dialogs that feature digitized audio, recognition of DTMF key input, recording of spoken input, and telephony.

A VoiceXML document is primarily composed of top-level elements called dialogs. Each dialog determines the next dialog to transition to. Transitions are specified using IDs, which define the next document and dialog to use. If an ID does not refer to a document, the current document is assumed. If it does not refer to a dialog, the first dialog in the document is assumed. Document execution begins at the first dialog by default. As each dialog executes, it determines the next dialog. Execution is terminated when a dialog does not specify a successor, or if it has an element that explicitly exits the conversation.

There are two types of dialogs: forms and menus:

- Forms define an interaction that collects values for a set of form item variables.
- A menu presents the user with a choice of options and then transitions to another dialog based on that choice.

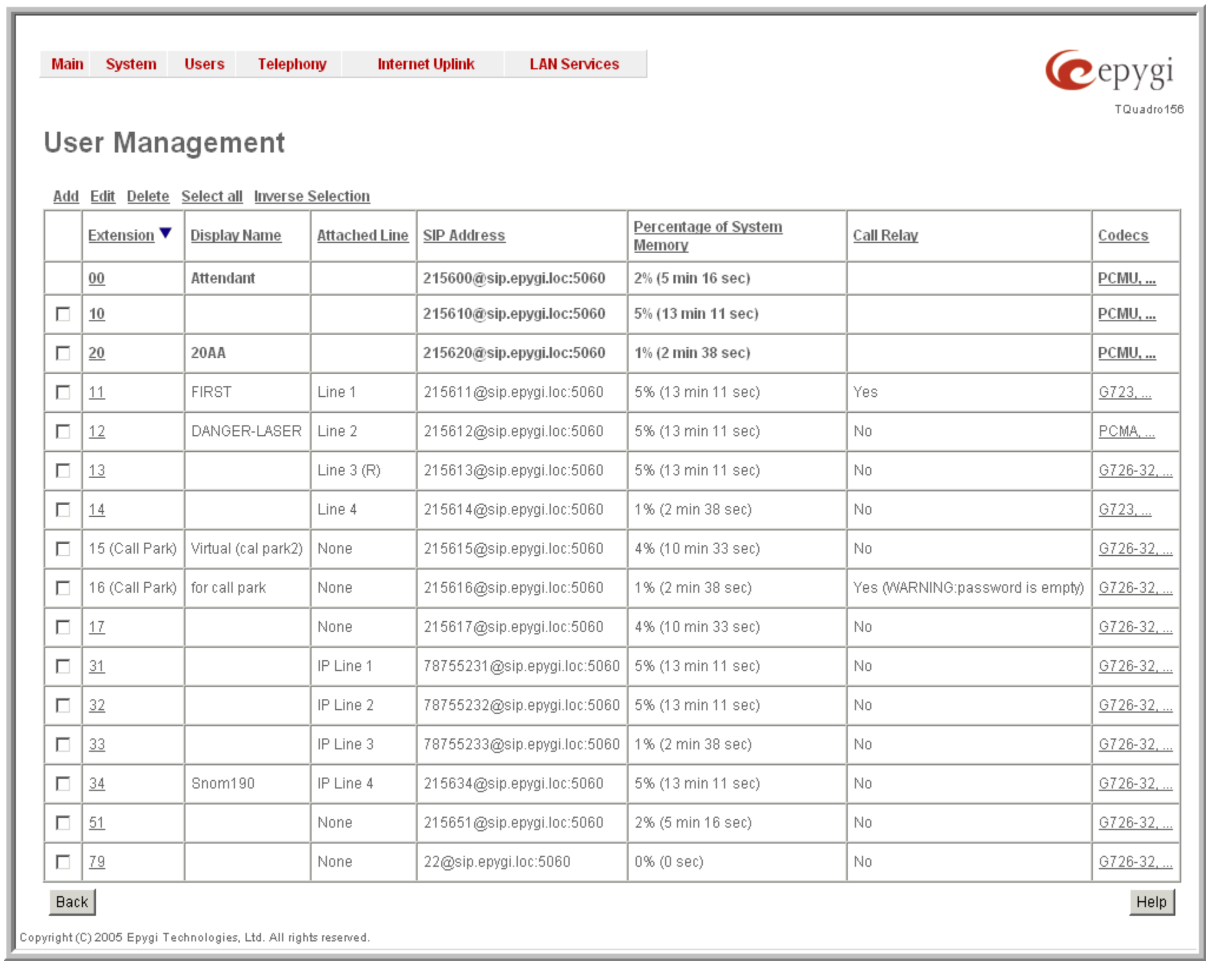
4 Creating an AA with a Custom Scenario

Here are the steps the administrator needs to perform in order to configure AA to use a custom scenario:

1. Prepare the VoiceXML scenario and voice messages (voice prompts).
2. Add a new entry to the **User Management** table with **Type** set to **Attendant**.
3. Specify the settings for AA (Attendant Settings, SIP Registration Settings, etc.) and upload the custom scenario file and voice messages.
4. Define a rule in the Local Routing table to route calls from some source(s) to the created custom AA (optional).

4.1 Creating a New Extension as Auto Attendant

1 On the **User Management** page, click on **Add** to add a new extension (Fig. 1).



User Management

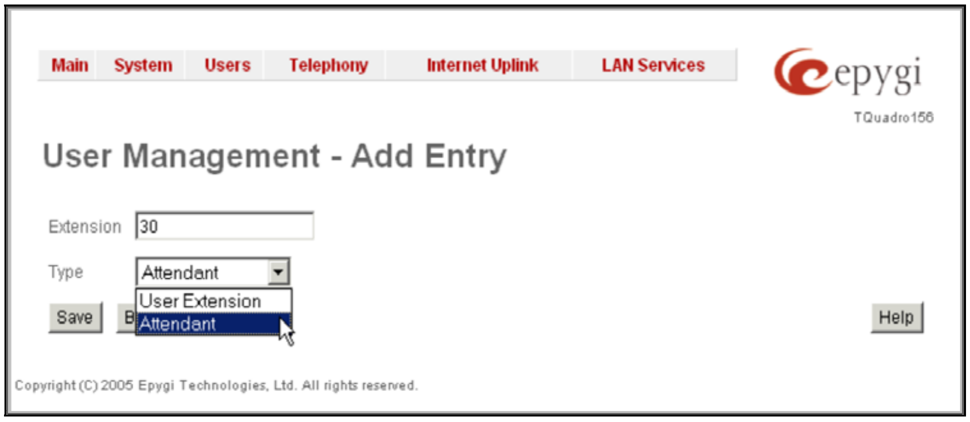
Add Edit Delete Select all Inverse Selection

	Extension ▼	Display Name	Attached Line	SIP Address	Percentage of System Memory	Call Relay	Codecs
	00	Attendant		215600@sip.epygi.loc:5060	2% (5 min 16 sec)		PCMU, ...
☐	10			215610@sip.epygi.loc:5060	5% (13 min 11 sec)		PCMU, ...
☐	20	20AA		215620@sip.epygi.loc:5060	1% (2 min 38 sec)		PCMU, ...
☐	11	FIRST	Line 1	215611@sip.epygi.loc:5060	5% (13 min 11 sec)	Yes	G723, ...
☐	12	DANGER-LASER	Line 2	215612@sip.epygi.loc:5060	5% (13 min 11 sec)	No	PCMA, ...
☐	13		Line 3 (R)	215613@sip.epygi.loc:5060	5% (13 min 11 sec)	No	G726-32, ...
☐	14		Line 4	215614@sip.epygi.loc:5060	1% (2 min 38 sec)	No	G723, ...
☐	15 (Call Park)	Virtual (cal park2)	None	215615@sip.epygi.loc:5060	4% (10 min 33 sec)	No	G726-32, ...
☐	16 (Call Park)	for call park	None	215616@sip.epygi.loc:5060	1% (2 min 38 sec)	Yes (WARNING:password is empty)	G726-32, ...
☐	17		None	215617@sip.epygi.loc:5060	4% (10 min 33 sec)	No	G726-32, ...
☐	31		IP Line 1	78755231@sip.epygi.loc:5060	5% (13 min 11 sec)	No	G726-32, ...
☐	32		IP Line 2	78755232@sip.epygi.loc:5060	5% (13 min 11 sec)	No	G726-32, ...
☐	33		IP Line 3	78755233@sip.epygi.loc:5060	1% (2 min 38 sec)	No	G726-32, ...
☐	34	Snom190	IP Line 4	215634@sip.epygi.loc:5060	5% (13 min 11 sec)	No	G726-32, ...
☐	51		None	215651@sip.epygi.loc:5060	2% (5 min 16 sec)	No	G726-32, ...
☐	79		None	22@sip.epygi.loc:5060	0% (0 sec)	No	G726-32, ...

Copyright (C) 2005 Epygi Technologies, Ltd. All rights reserved.

Fig. 1: User Management

2 On the **User Management – Add Entry** page, enter an extension, you would like the AA to be reached at, and set the **Type** to **Attendant** (Fig. 2). A new AA will be created with the default scenario (Fig. 3)**.**

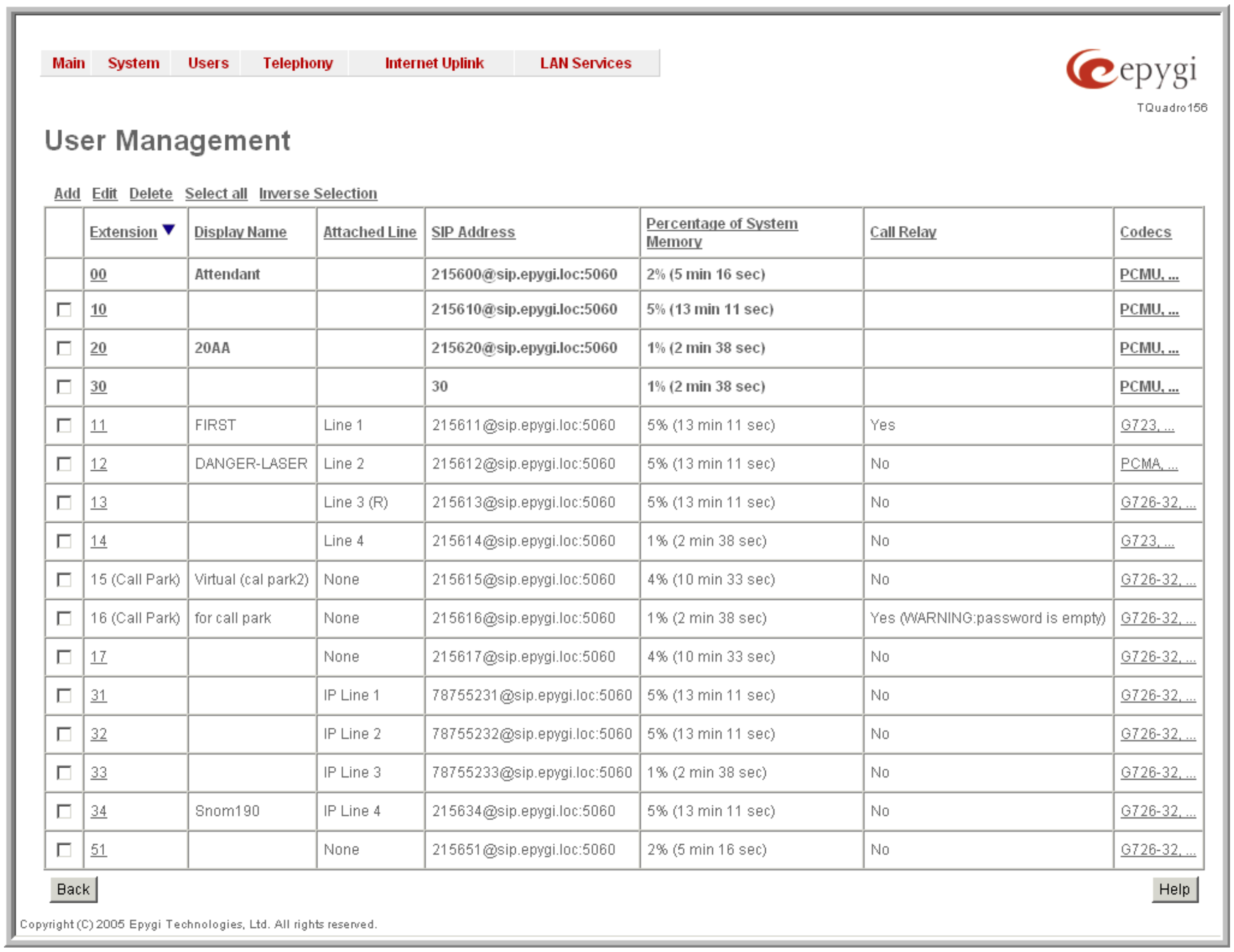


User Management - Add Entry

Extension	30
Type	Attendant

Copyright (C) 2005 Epygi Technologies, Ltd. All rights reserved.

Fig. 2: User Management - Add Entry



User Management

	Extension ▼	Display Name	Attached Line	SIP Address	Percentage of System Memory	Call Relay	Codecs
	00	Attendant		215600@sip.epygi.loc:5060	2% (5 min 16 sec)		PCMU, ...
☐	10			215610@sip.epygi.loc:5060	5% (13 min 11 sec)		PCMU, ...
☐	20	20AA		215620@sip.epygi.loc:5060	1% (2 min 38 sec)		PCMU, ...
☐	30			30	1% (2 min 38 sec)		PCMU, ...
☐	11	FIRST	Line 1	215611@sip.epygi.loc:5060	5% (13 min 11 sec)	Yes	G723, ...
☐	12	DANGER-LASER	Line 2	215612@sip.epygi.loc:5060	5% (13 min 11 sec)	No	PCMA, ...
☐	13		Line 3 (R)	215613@sip.epygi.loc:5060	5% (13 min 11 sec)	No	G726-32, ...
☐	14		Line 4	215614@sip.epygi.loc:5060	1% (2 min 38 sec)	No	G723, ...
☐	15 (Call Park)	Virtual (cal park2)	None	215615@sip.epygi.loc:5060	4% (10 min 33 sec)	No	G726-32, ...
☐	16 (Call Park)	for call park	None	215616@sip.epygi.loc:5060	1% (2 min 38 sec)	Yes (WARNING:password is empty)	G726-32, ...
☐	17		None	215617@sip.epygi.loc:5060	4% (10 min 33 sec)	No	G726-32, ...
☐	31		IP Line 1	78755231@sip.epygi.loc:5060	5% (13 min 11 sec)	No	G726-32, ...
☐	32		IP Line 2	78755232@sip.epygi.loc:5060	5% (13 min 11 sec)	No	G726-32, ...
☐	33		IP Line 3	78755233@sip.epygi.loc:5060	1% (2 min 38 sec)	No	G726-32, ...
☐	34	Snom190	IP Line 4	215634@sip.epygi.loc:5060	5% (13 min 11 sec)	No	G726-32, ...
☐	51		None	215651@sip.epygi.loc:5060	2% (5 min 16 sec)	No	G726-32, ...

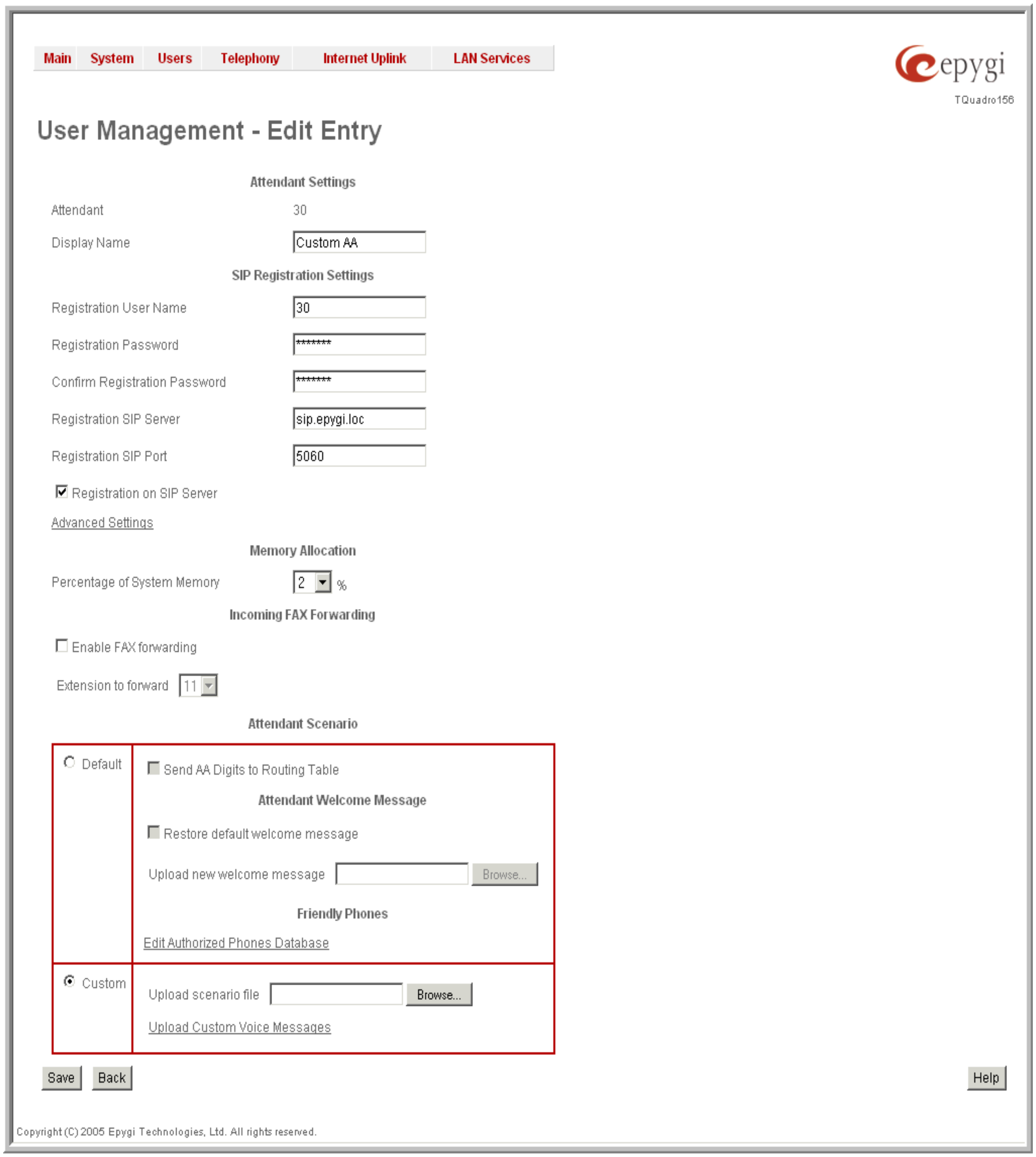
Copyright (C) 2005 Epygi Technologies, Ltd. All rights reserved.

Fig. 3: Resulting User Management Table

4.2 Specifying the Settings for AA and Uploading the Custom Scenario

To specify the settings for AA, select that extension and click on **Edit**.

1. Fill in the settings on this page and select the **Custom** option for **Attendant Scenario** (Fig. 4).



User Management - Edit Entry

Attendant Settings

Attendant	30
Display Name	Custom AA

SIP Registration Settings

Registration User Name	30
Registration Password	********
Confirm Registration Password	********
Registration SIP Server	sip.epygi.loc
Registration SIP Port	5060

☑ Registration on SIP Server

Advanced Settings

Memory Allocation

Percentage of System Memory 2 %

Incoming FAX Forwarding

☐ Enable FAX forwarding

Extension to forward 11

Attendant Scenario

○ Default
- ☐ Send AA Digits to Routing Table
- Attendant Welcome Message
- ☐ Restore default welcome message
- Upload new welcome message [ ] Browse...
- Friendly Phones
- Edit Authorized Phones Database

● Custom
- Upload scenario file [ ] Browse...
- Upload Custom Voice Messages

Save Back Help

Copyright (C) 2005 Epygi Technologies, Ltd. All rights reserved.

Fig. 4: User Management - Edit Entry

2. Upload the scenario file.
3. Click on the **Upload Custom Voice Message** link to open the **Upload File** page (Figure 5) where your custom voice messages can be uploaded.

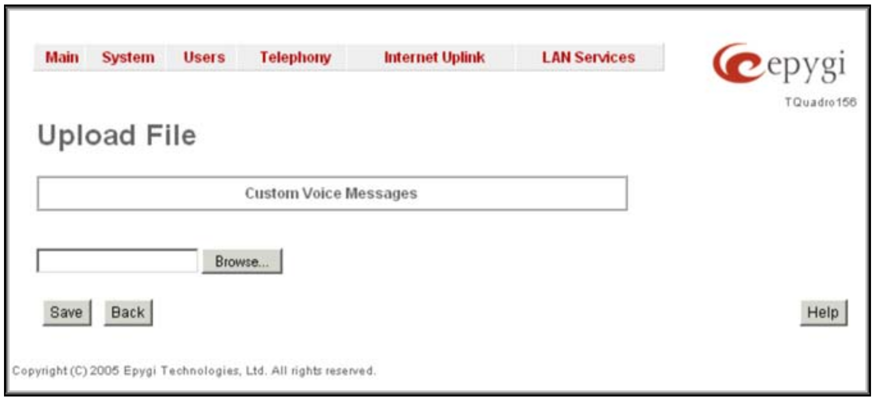


Fig. 5: Upload File

5 Appendix: VoiceXML Elements Supported by Quadro

Elements	Attributes	Parents	Children
assign	expr, name	block, catch, filled, noinput, nomatch	none
audio	expr, src	prompt	none
block	name	form	assign, clear, exit, goto, prompt, reprompt, throw, var
break	time	prompt	none
catch	count, event	field, form, menu, record, transfer, vxml	assign, clear, exit, goto, prompt, reprompt, throw, var
choice	dtmf, event, eventexpr, expr, next	menu	none
clear	namelist	block, catch, filled, noinput, nomatch	none
else	none	block, catch, filled, noinput, nomatch	assign, clear, exit, goto, prompt, reprompt, throw, var
elseif	cond	block, catch, filled, noinput, nomatch	assign, clear, exit, goto, prompt, reprompt, throw, var
exit	expr, namelist	block, catch, filled, noinput, nomatch	none
field	expr, modal, name, type	form	catch, filled, link, noinput, nomatch, option, prompt
filled	none	field, record, transfer	assign, clear, enumerate, exit, goto, prompt, reprompt, throw, var
form	id	vxml	block, catch, field, link, noinput, nomatch, record, transfer, var
goto	expr, expritem, next, nextitem	block, catch, filled, noinput, nomatch	none
if	cond	block, catch, filled, noinput, nomatch	assign, clear, exit, goto, prompt, reprompt, throw, var
link	dtmf, event, eventexpr, expr, next	field, form, vxml	none
menu	dtmf, id	vxml	catch, choice, noinput, nomatch, prompt, property, script, value
noinput	count	field, form, menu, record	assign, clear, exit, goto
nomatch	count	field, form, menu, record	assign, clear, exit, goto
object	classid, name	form	catch, filled, noinput, nomatch, prompt
option	dtmf, value	field	
param	expr, name, type, value	object	
prompt	bargein, count, timeout	block, catch, field, filled, menu, record	audio
record	beep, dest, destexpr, dtmfterm, expr, name	form	catch, filled, noinput, nomatch, prompt
reprompt	none	block, catch, filled	none
throw	event, eventexpr	block, catch, error, filled, help, if	none
var	expr, name	block, catch, filled, form, vxml	none
Vxml	version	none	catch, form, link, menu, script, var

6 Appendix: VoiceXML Elements Supported by Quadro

Elements	Descriptions	Attributes	Descriptions
<assign>	Assigns a value to an existing variable explicitly declared by the <var> element.	**<name>**	The name of the variable being assigned to.
		<expr>	The new value of the variable.

```
<var name="Var1" expr="'value of the first var'"/>
<var name="Var2" expr="'value of the second var '"/>
<form id="Form1">
<block>
                <assign name="Var1" expr="'another value'"/>
                <assign name="Var1" expr="Var2"/>
        </block>
</form>
```

<audio>	Plays an audio wave file within a prompt. (8bit, 8Khz, u-law (a-law) files)	**<expr>**	The expr attribute is used when we want to use a generated value rather than an explicit constant expression for our audio file's destination.
		<src>	The name (path) of the audio prompt.

```
<var name="Var1" expr="'attwelcome.wav'"/>
<form id=" Form1">
        <block>
                <prompt> <audio src="mysoundfile.wav"/> </prompt>
                <prompt> <audio expr="Var1"/> </prompt>
        </block>
</form>
```

<block>	A form-item container of (non-interactive) executable code	**<name>**	The value or 'ID' of the block used for navigational purposes.

```
<form id="Form1">
        <block>
        <prompt> <audio src="mysoundfile.wav"/> </prompt>
        <goto nextitem= "Block2"/>
        </block>
        <block name="Block2">
                <prompt> <audio src="mysoundfile.wav"/> </prompt>
        </block>
</form>
```

<break>	Designate a pause in the audio output.	**<time>**	A value of '5s' would indicate a 5 second pause within the audio output.

```
<form id="Form1">
        <block>
                <prompt> <audio src="mysoundfile.wav"/> </prompt>
                <break time="2s"/>
        </block>
</form>
```

<catch>	Catches an event. The content nested within the catch element will be executed when its particular *event* is thrown.	**<count>**	Allows different <catch> elements to handle the same event based on the number of occurrences of the event. When an event is thrown for the first time, the catch element with a *count* of 1 will be executed.
		<event>	The event or events to catch (separated by space). If the attribute is unspecified, all events are to be caught.

```
        <catch count="1" event="noinput">
                <prompt> <audio src="mysoundfile.wav"/> </prompt>
        </catch>
```

<choice>	Defines a menu item.	**<dtmf>**	The dtmf key linked to a specific menu choice.
		<event>	The event to be thrown upon a choice match.
		<eventexpr>	An expression evaluating to the event being thrown to the application.
		<expr>	A variable name whose value will be used to determine the dialog ID or document path
		<next>	The dialog ID or document path that the application transitions to upon a user's selection.

```
<menu id="Menu1">
        <prompt> <audio src="mysoundfile.wav"/> </prompt>
                <choice dtmf="1" next="#Form1"/>
                <choice dtmf="2" next="#Form2"/>
                <choice dtmf="3" expr="Var1"/>
                <choice dtmf="4" event="nomatch"/>
                <choice dtmf="5" eventexpr="Var2 "/>
</menu>
```

<clear>	Sets a variable(s) to an undefined value.	**<namelist>**	The space-separated list of variables to be reset.

```
<var name="Var1" expr="'some value'"/>
<form id="Form1">
        <block>
                <clear namelist="Var1"/>
        </block>
</form>
```

<exit>	Terminates the current dialog and returns control to the interpreter.	***none***	

```
<form id="Form1">
        <block>
                <exit/>
        </block>
</form>
```

<field>	Allows the interpreter to collect information from the user.	**<expr>**	The initial value of the element; if this value is 'undefined', (default), then the element will be executed. If not, then the element will not be visited until explicitly set to 'undefined' .
		<modal>	If set to false (the default) all active grammars are turned on while collecting this field; otherwise, only the field's grammars are enabled.
		<name>	The variable name that declares the field-item variable for the dialog.

		<type>	The type of field, i.e., the name of a built in grammar type.

```
<form id="Form1">
        <field name="Field1" type="digits">
                <prompt> <audio src="mysoundfile1.wav"/> </prompt>
        </field>
        <field name="Field2" type="digits?length=2">
                <prompt> <audio src="mysoundfile2.wav"/> </prompt>
        </field>
        <field name="Field3" type="digits" modal="true">
                <prompt> <audio src="mysoundfile3.wav"/> </prompt>
        </field>
</form>
```

<filled>	Specifies actions to perform when input items are filled.	***none***	

```
<form id="Form1">
        <field name="Field1">
                <prompt> <audio src="mysoundfile1.wav"/> </prompt>
                <filled>
                        <prompt> <audio src="mysoundfile2.wav"/> </prompt>
                </filled>
        </field>
</form>
```

<form>	A container for all field-items.	**<id>**	The name of the form used to refer to the form within the document or from another document.

```
<form id="F1">
        <var name="Var1" expr="'variable with a form scope'"/>
                <field name="Field1">
                        <prompt> <audio src="mysoundfile1.wav"/> </prompt>
                </field>
                <block>
                        <prompt> <audio src="mysoundfile2.wav"/> </prompt>
                </block>
        </form>
```

<goto>	Transitions application execution to a specific *form* within the current document, or to an entirely separate document.	**<expr>**	An expression that yields the ID of the next dialog to visit.
		<expritem>	An expression that yields the name of the next form item to visit.
		<next>	The ID of the next dialog to visit.
		<nextitem>	The name of the next form item to visit in the current form.

```
<var name="Var1" expr="'Block4'"/>
<form id="Form1">
        <block name="Block1">
                <prompt> <audio src="mysoundfile.wav"/> </prompt>
                <goto nextitem="Block2"/>
        </block>
        <block name="Block2">
                <prompt> <audio src="mysoundfile.wav"/> </prompt>
                <goto expritem="Var1"/>
        </block>
        <block name="Block3">
                <prompt> <audio src="mysoundfile.wav"/> </prompt>
                <goto next="#Form2"/>
        </block>
        <block name="Block4">
                <prompt> <audio src="mysoundfile.wav"/> </prompt>
                <goto nextitem="Block3"/>
        </block>
</form>
```

<if>	Provides a method to utilize conditional logic expressions which allow the developer to change the control flow within the application based on user utterances, variable values, or events.	**<cond>**	The cond attribute specifies any valid expression, which equates to the value to be evaluated.

```
<form id="Form1">
        <field name="Field1">
                <prompt> <audio src="mysoundfile.wav"/> </prompt>
                <filled>
                        <if cond="Field1=='abc'">
                <prompt> <audio src="mysoundfile1.wav"/> </prompt>
                        <elseif cond="Field1=='def'"/>
                <prompt> <audio src="mysoundfile2.wav"/> </prompt>
                        <else/>
                <prompt> <audio src="mysoundfile3.wav"/> </prompt>
                        </if>
                </filled>
        </field>
</form>
```

<link>	Allows to easily implement a document or application scoped *grammar* and transition/event handler for a caller's input.	**<dtmf >**	The dtmf key which can be used in conjunction with any other voice grammar specified within the *link* itself to validate a successful grammar match.
		<event>	An event to be thrown to the application.
		<eventexpr>	An expression that evaluates to the event being thrown to the application.
		<expr>	An value that defines the target URI. Either *expr* or *next* may be specified, but not both.
		<next>	The destination URI or form item to transition the caller to when the *link grammar* is matched.

```
<link dtmf="1" next="#Menu1"/>
<link dtmf="2" next="#Menu2"/>
<link dtmf="3" expr="Var1"/>
<link dtmf="4" event="nomatch"/>
<link dtmf="5" eventexpr="Var2"/>
```

<menu>	A dialog for choosing among alternative destinations.	**<dtmf>**	When set to *'true'*, assigns the first 9 menu choices implicit dtmf grammar values, unless the choices already have them explicitly assigned. Defaults to false.

		<id>	The navigational identifier of the form element that allows the menu to be the target of a <goto>.

```
        <menu id="Menu1">
                <prompt> <audio src="mysoundfile.wav"/> </prompt>
        <choice dtmf="1" next="#Form1"/>
        <choice dtmf="2" next="#Form2"/>
        </menu>
```

<noinput>	A shorthand for ***<catch event="noinput">***, handling the *"noinput"* event, thrown when the application expects DTMF input, but has received none from the caller.	**<count>**	Allows different <noinput> elements to handle the event based on the number of occurrences of the event. When an event is thrown for the first time, the catch element with a *count* of 1 will be executed.

```
<form id="Form1">
        <field name="Field1">
                <noinput count="1">
                        <prompt> <audio src="mysoundfile1.wav"/> </prompt>
                        <reprompt/ >
                </noinput>
                <noinput count="2">
                        <prompt> <audio src="mysoundfile2.wav"/> </prompt>
                </noinput>
                <noinput count="5">
                        <prompt> <audio src="mysoundfile3.wav"/> </prompt>
                        <exit/ >
                </noinput>
        </field>
</form>
```

<nomatch>	A shorthand for ***<catch event="nomatch">***, handling the *"nomatch"* event, thrown when the caller inputs a value that is not recognized by any of the active grammars.	**<count>**	Allows different <nomatch> elements to handle the event based on the number of occurrences of the event. When an event is thrown for the first time, the catch element with a *count* of 1 will be executed.

```
<form id="Form1">
        <field name="Filed1">
                <nomatch count="1">
                        <prompt> <audio src="mysoundfile1.wav"/> </prompt>
                        <reprompt/ >
                </nomatch>
                <nomatch count="2">
                        <prompt> <audio src="mysoundfile2.wav"/> </prompt>
                </nomatch>
                <nomatch count="5">
                        <prompt> <audio src="mysoundfile3.wav"/> </prompt>
                        <exit/ >
                </nomatch>
        </field>
</form>
```

<object>	Exposes implementation platform-specific functionality.	**<classid>**	The name specifying platform dependent object implementation.
		<name>	The variable name that declares the field-item variable for the dialog.

```
<var name="Var1" expr="'somevalue'"/>
<form id="Form1">
        <object name="Obj1" classid="Class1">
                <param name="Param1" expr="'22'"/>
                <param name="Param2" value="22"/>
                <param name="Param3" expr="Var1"/>
        </object>
</form>
```

<option>	Lists choices to the caller.	**<dtmf>**	The dtmf key that a caller may input to activate a particular listed choice.
		<value>	The string to use for the field item return when a user selects a particular option. If not specified, then the *dtmf* value is returned.

```
<form id="Form1">
        <field name="Field1">
                <prompt> <audio src="mysoundfile.wav"/> </prompt>
                <option dtmf="1" value="value1"/>
                <option dtmf="2" value="value2"/>
                <option dtmf="3" value="value3"/>
        </field>
</form>
```

<param>	Specifies the value passed to *objects*.	**<expr>**	An expression defining the value for the parameter.
		<name>	The name of the parameter
		<value>	The string to use for the field item return when a user selects a particular option. If not specified, then the *dtmf* value is returned.

```
<var name="Var1" expr="'somevalue'"/>
<form id="Form1">
        <object name="Obj1" classid="Class1">
                <param name="Param1" expr="'22'"/>
                <param name="Param2" value="22"/>
                <param name="Param3" expr="Var1"/>
        </object>
</form>
```

<prompt>	The prompt element allows the developer to output audio to the caller.	**<bargein>**	Specifies whether ('true') or not ('false') the caller will be able to interrupt the audio output with a dtmf keypress.
		<count>	A number that allows emitting different prompts if the user is doing something repeatedly, For instance, a *prompt count=2* could play more detailed instructions to the caller should his first input proves to be invalid. Defaults to 1,
		<timeout>	Specifies the amount of the time the interpreter should wait for a user response before throwing a *noinput* event. Defaults to 5s (5 seconds).

```
<form id="Form1">
        <field name="Filed1">
                <prompt bargein="false" timeout="5s" count ="1">
                        <audio src="mysoundfile1.wav"/>
                </prompt>
        </field>
</form>
```

<record>	An input item which records audio from the caller and stores the resultant audio file in its namespace variable.	**<beep>**	If true, a beep is emitted just prior to recording. Defaults to false.
		<dest>	The ‘catcher’ path to upload the recorded audio file to.
		<destexpr>	An expression resolving to the destination of a record audio/catcher script.
		<dtmfterm>	The dtmfterm attribute specifies whether the caller will be allowed to terminate the recording in progress with the press of a dtmf key. When set to 'true', (default), any dtmf keypress can end the recording session. If set to 'false', then the recording continues until a maxtime.
		<expr>	The initial value of the form item variable; default is undefined. If initialized to a value, then the form item will not be visited unless the form item variable is cleared.
		<name>	The input item variable that will hold the recording.

```
<form id="Form1">
        <record name="Rec1">
                <prompt> <audio src="mysoundfile.wav"/> </prompt>
        </record>
</form>
```

<reprompt>	Increases the prompt counter by one and replays the most recent *prompt* .	***none***	

```
<form id="Form1">
        <field name="Field1">
                <prompt> <audio src="mysoundfile.wav"/> </prompt>
                <nomatch>
                <reprompt/>
                </nomatch>
        </field>
</form>
```

<throw>	Triggers an event which can be caught and handled by using the *catch* element.	**<event>**	The name of the event to *throw* to the application.
		<eventexpr>	An expression evaluating to the name of the event being thrown.

```
<var name="Var1" expr="'noinput'"/>
<form id="Form1">
        <block>
                <throw event="nomatch"/>
        </block>
        <block>
                <throw event="Var1"/>
        </block>
</form>
```

<var>	Declares a VXML variable.	**<expr>**	The initial value of the variable (optional).
		<name>	The variable name.

```
<var name="Var1" expr="'somevalue'"/>
<var name="Var2" expr="'somevalue'"/>
<form id="Form1">
        block>
                <assign name="Var1" expr="'anothervalue'"/>
                <assign name="Var2" expr="Var1"/>
        </block>
</form>
```

<vxml>	The initial declaration that defines a document as a VoiceXML application.	**<version>**	The VoiceXML version number.

```
<vxml version = "2.0">
<  var name="Var1" expr="'somevalue'"/>
        <form id="Form1">
                <block>
                <prompt> <audio src="mysoundfile.wav"/> </prompt>
                <goto next="#Menu1"/>
                </block>
        </form>
  <menu id="Menu1">
                <prompt> <audio src="mysoundfile.wav"/> </prompt>
  </menu>
</vxml>
```

7 Appendix: VoiceXML Objects for Final Options of AA's Custom Scenario

1. checklogin	Verify login information for specified as param “extension” and “password”.

```
<object name="objfieldname" classid="checklogin">
    <param name="extension" expr="’some extension’"/>
    <param name="password" expr="’some password’"/>
</object>
```

2. dial	Dials to the destination (“pattern”) specified as param.

```
<object name="objfieldname" classid="dial">
    <param name="pattern" expr="’some pattern’"/>
</object>
```

3. connect	Dials to the extension specified as param.

```
<object name="objfieldname" classid="connect">
    <param name="extension" expr="’some extension’"/>
</object>
```

4. service	Activates service, specified as param.

```
<object name=" objfieldname" classid="service">
    <param name="name" expr="'vm'"/>
</object>
```

8 Appendix: A Sample Scenario File for Custom AA

The xml file below is an AA's custom scenario file. An AA with this scenario will greet callers; ask either to enter the party's extension or to press # for help. If the caller presses #, it asks whether the caller would like to address his question to Technical Support (1), to Testing and Documentation (2) or he would like to get some product information.

The pre-recorded voice messages are as follows:

epygiwelcome.wav	"Welcome to Epygi Technologies' Customer Service!"
menu1.wav	"If you know your party's extension, please, enter it now" or "press # for help"
menu2.wav	"To address your question to Technical Support, press 1" "For Testing and Documentation, press 2" "To get product information, press 3"
techsupport.wav	"To call Aram press 1" " To call Serzh press 2"
documentation.wav	"To call Liana press 1" "To call Diana press 2"
information.wav -	Music☺

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0">
        <noinput count="3">
                <prompt> <audio src="timeover.wav" /> </prompt>
                <exit/>
        </noinput>
        <link dtmf="**" next="#mainform"/>
        <form id="mainform">
                <block>
                        <prompt bargein="true"> <audio src="epygiwelcome.wav" /> </prompt>
                        <goto nextitem="field1"/>
                </block>
                <field name="field1" type="digits?length=2" modal="false">
                        <link dtmf="#" next="#menu2"/>
                        <prompt timeout="5s"> <audio src="menu1.wav" /> </prompt>
                        <filled>
                                <goto nextitem="field2"/>
                        </filled>
                </field>
                <object name="field2" classid="connect">
                        <param name="extension" expr="field1"/>
                        <filled>
                                <prompt> <audio src="timeover.wav" /> </prompt>
                                <exit/>
                        </filled>
                        <nomatch>
                                <prompt> <audio src="attinvext.wav" /> </prompt>
                                <goto nextitem="field1"/>
                        </nomatch>
                </object>
        </form>

        <menu id="menu2">
                <prompt timeout="5s"> <audio src="menu2.wav" /> </prompt>
                <choice dtmf="1" next="#techsupport"/>
                <choice dtmf="2" next="#documentation"/>
                <choice dtmf="3" next="#information"/>
        </menu>

        <form id="techsupport">
                <field name="field1">
                        <prompt timeout="5s"> <audio src="techsupport.wav" /> </prompt>
                        <option dtmf="1" value="22"/>
                        <option dtmf="2" value="23"/>
                        <filled>
                                <goto nextitem="field2"/>
                        </filled>
                </field>
                <object name="field2" classid="connect">
                        <param name="extension" expr="field1"/>
                        <nomatch>
                                <prompt> <audio src="attinvext.wav" /> </prompt>
                                <goto nextitem="field1"/>
                        </nomatch>
                </object>
        </form>

        <form id="documentation">
                <field name="field1">
                        <prompt timeout="5s"> <audio src="documentation.wav" /> </prompt>
                        <option dtmf="1" value="24"/>
                        <option dtmf="2" value="25"/>
                        <filled>
                                <goto nextitem="field2"/>
                        </filled>
                </field>
                <object name="field2" classid="connect">
                        <param name="extension" expr="field1"/>
                        <nomatch>
                                <prompt> <audio src="attinvext.wav" /> </prompt>
                                <goto nextitem="field1"/>
                        </nomatch>
                </object>
        </form>

        <form id="information">
                <block>
                        <prompt timeout="5s"> <audio src="information.wav" /> </prompt>
                        <goto next="#menu2"/>
                </block>
        </form>
</vxml>
```