XML-Based Languages for Intelligent Service Creation

Evelina Pencheva[1], Ivaylo Atanasov[2]

Abstract - **CPL is a language that can be used to explain to SIP servers what to do with a call. VoiceXML is a language that is used to describe speech dialogues between the computer and the end user. In this paper we investigate how XML-based languages like CPL and VoiceXML can be used in creating value-added services in IP networks.**

Keywords - **service creation, intelligent network, CPL, VoiceXML**

I. INTRODUCTION

Intelligent Network (IN) was designed in the early 1990s for delivering value added services in circuit switched telephony networks. At that time the World Wide Web was still in its infancy and in many countries a single national operator still owned the network. Although there are many features of IN that can be recognized in the Internet, there is one important differrence. In IN these features are centrally controlled; in the Internet they are completely distributed through the network.

The IN and Internet protocol (IP) technologies are converging in a variety of ways including the use of IP for transmission of voice - voice over IP (VoIP) - for more cost effective transport of voice communications. In addition, efforts are underway to allow intelligence in public switched networks to interface and interact with intelligence in IP based networks, and doing so, provide greater overall intelligence.

An important effort to define capabilities for hybrid IN + IP networks involves the integration of IN with the Session Initiation Protocol (SIP). This integration will allow for more flexible service control options than could be afforded through either technology by itself.

The Call Processing Language (CPL) is a XML-based language to describe and control Internet telephony services. It is designed to be implementable on either network servers or user agents. It is meant to be simple, extensible, and independent of operating system or signaling protocol. It is suitable for running on a server where users may not be allowed to execute arbitrary programs, as it has no variables, loops, or ability to run external programs.

VoiceXML is a markup language that enables integration of voice services with data services using the familiar client-server paradigm. It separates user interaction code (in VoiceXML) from service logic. The language promotes service portability across implementation platforms and is easy to use for simple interactions, and yet provides language features to support complex dialogs.

[1]Evelina Pencheva is with the Faculty of Telecommunications, Technical University – Sofia, blvd "Kliment Ohridsky" 8, 1000 Sofia, Bulgaria, enp@tu-sofia.bg

[2]Ivaylo Atanasov is with the Faculty of Telecommunications, Technical University – Sofia, blvd "Kliment Ohridsky" 8, 1000 Sofia, Bulgaria, iia@tu-sofia.bg

In this paper we explore the capabilities of CPL and VoiceXML that help end users to define their call treatment. We consider an example of malicious call identification service and suggest service scripts in CPL and VoiceXML.

II. SERVICE CREATION IN INTELLIGENT NETWORKS

The IN provides great flexibility to service creation in general and also to the tailoring of services to suit the exact requirements of a particular user. Service creation in IN involves three basic components – SIBs, service features, and services. SIBs are the smallest building blocks that describe reusable network capability used to create service features. Some functions are commonly used for many services, for example, number translation services or mass calling services. These functions are called service features and are built of one or more SIBs. A service is built by combining one or more SIBs and features or services or both. To illustrate how SIBs can be combined to build service logic script, the malicious call identification service is considered.

The malicious call identification (MCI) service is built by combining the following core features: call logging (LOG) and originating call screening (OCS). LOG allows a record to be prepared each time a call is received by a special telephone number. OCS allows the served user to bar calls (call screening) from certain areas based on the district code of the area from which the call originates.

Malicious call identification service can incorporate one optional feature: customer profile management (CPM). CPM allows the subscriber to manage the service profile in real time, that is, changing answering places, control the announcements to be played, perform call distribution, change time-dependent routing, and so on.

Combining the optional feature CPM and the core feature OCS creates the remote control of malicious call identification service.

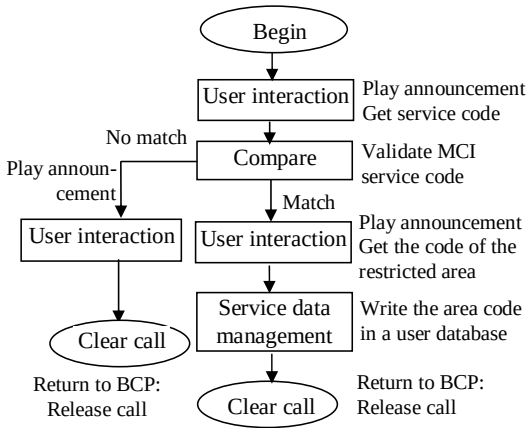

Figure 1 Service script for MCI service activation

When modeling service logic two cases have to be considered: service activation and service execution. Figure 1 shows a simplified version of a service script for the activation of MCI service. For the sake of simplicity, Figure 1 shows only SIBs flow with comments, but not SIB's parameters. When the subscriber wants to activate the MCI service, he has to dial a special number, provided by the network operator. This number accesses the management functions of the subscriber's profile.

In essence, this service activation logic performs the following steps:

- A message is played asking for the service code and user input is received in the form of DTMF tones.
- The service code is verified against the MCI service code.
- If the received code is MCI service code, a message is played to the user, asking for the district code of the restricted area and the user input is received.
- The restricted area code is stored in the user database and the call is cleared.
- If the received code is not MCI service code, an error message is played to the user and the call is cleared.

Basic call process (BCP) is a special SIB that describes the phases of call set up. At each of these phases it is possible to interrupt the call setup and to start execution of a service script. After finishing the processing of the service, the last SIB hands back control to the BCP.

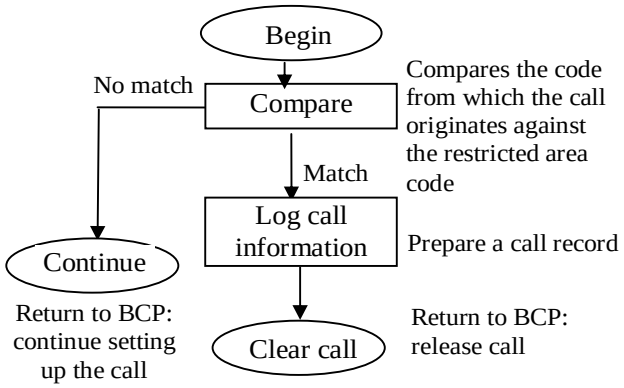

Figure 2 Service script for MCI service execution

Figure 2 shows a simplified version of a service logic for MCI service execution. The service proceeds in the following way.

- The district code from which the call originates is compared against the restricted area code.
- If the call originates from the restricted area, a record is prepared and the call is cleared.
- If the call originates from an area that is different than the restricted area the call setup continues.

III. XML-BASED LANGUAGES AND IN

The CPL can be used to describe and control Internet telephony services. It is not tied to any particular signaling architecture or protocol; it is anticipated that it will be used with both the SIP and H.323.

SIP is an application-layer Internet protocol for setting up multimedia conferences over the Internet. A SIP server can perform any of the IN features, such as those for numbering, routing, charging, access, and restriction. CPL provides tools for creating call-processing scripts that runs on SIP servers.

CPL is powerful enough to describe a large number of IN services and features, but it is limited in power so that it can run safely in Internet telephony servers. The intention is to make it impossible for users to do anything more complex (and dangerous) than describe Internet telephony services. Table 1 shows how some of the IN SIBs can be translated to CPL.

TABLE 1
MAPPING IN SIBS TO CPL

IN SIB	*CPL equivalent*
Algorithm	No direct equivalent: CPL is not a programming language
Charge	No direct equivalent
Compare	Switch
Distribute	Switch
Limit	Switch follow by reject
Log call information	Nonsignaling action (log)
Queue	No direct equivalent; could be implemented in CPL by proxying calls to a special queuing server
Screen	Switch follow by reject
Service data management	Location modifier with lookup (CPL does not allow to write in a database)
Status notification	Signaling action (proxy)
Translate	Location modifier
User interaction	No direct equivalent; could be implemented in CPL by proxying calls to a special voice response server with VoiceXML interpreter.
Verify	No direct equivalent.

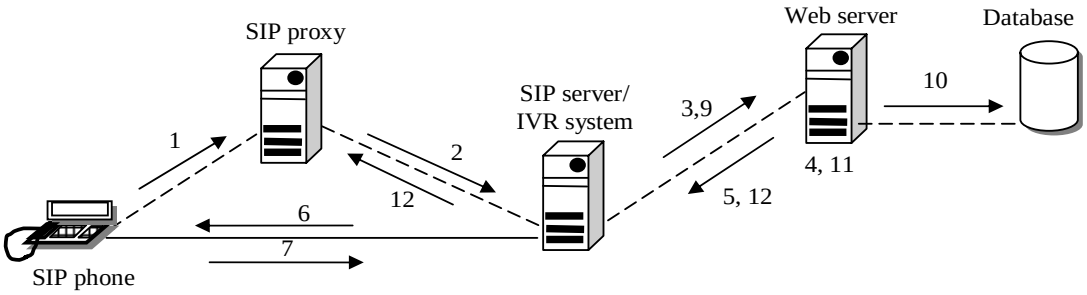

Figure 3 A scenario for MCI service activation

A voice service is viewed as a sequence of interaction dialogs between a user and an implementation platform. The dialogs are provided by document servers, which may be external to the implementation platform. Document servers maintain overall service logic, perform database and legacy system operations, and produce dialogs. A VoiceXML document specifies each interaction dialog to be conducted by a VoiceXML interpreter. User input affects dialog interpretation and is collected into requests submitted to a document server. The document server may reply with another VoiceXML document to continue the user's session with other dialogs. The user is prompted for input and then records it. The user input can be played back, and if the user approves it, is sent on to the server for storage using the HTTP[1] POST method. If the platform supports simultaneous recognition and recording, form and document scoped grammars can be active while the recording is in progress.

The JSpeech Markup Language (JSML) is a text format used by applications to annotate text input to speech synthesizers. JSML elements provide a speech synthesizer with detailed information on how to speak text and thus enable improvements in the quality, naturalness and understandability of synthesized speech output. JSML defines elements that describe the structure of a document, provide pronunciations of words and phrases, indicate phrasing, emphasis, pitch and speaking rate, and control other important speech characteristics. JSML can be used with conjunction of the VoiceXML.

IV. MCI SERVICE IMPLEMENTED BY CPL AND VOICEXML

The IN standards describing the interaction between service control function and service data function are defined in terms of INAP[2] information flows. In Internet telephony services, these interactions are replaced with HTTP requests to HTTP servers that hold the call screening data for a particular subscriber.

Let us consider MCI service provided over the Internet. Having the service the user can screen the call from a particular area. The steps for MCI activation, as shown in the Figure 3, are as follows:

1. The user dials a special number provided by service provider. The user agent (the software in the terminal) sends the *Invite* request to the SIP proxy.
2. The SIP proxy server looks at the address, determines that it belongs to SIP server with embedded Interactive Voice Relay (IVR) system and sends the *Invite* request on to SIP server/IVR system.
3. The SIP server/IVR system analyses the request and concludes that the user wants to activate a service. It determines the Web server that stores the service preparation script and sends a HTTP *GET* request to the Web server asking for service preparation script. The service preparation script can be a script written in PHP, Java, Perl or some other script language.
4. The service preparation script is executed and as result it creates a VoiceXML script that has to maintain the dialogue with the user in context of the service activation.
5. The Web server returns the HTTP reply with the VoiceXML script to the SIP server/IVR system.
6. The SIP server/IVR system starts interpreting the VoiceXML script and speaking to the user. The user is asked for the service he wants to activate.
7. The user inputs the MCI service code by DTMF tones.
8. The dialogue between the IVR system and the user continues for getting the restricted area code from which the user doesn't want to receive calls.
9. The SIP server/IVR system posts the information retrieved from the user to the Web server.
10. The Web server sends a SQL request to the Database for updating user data. The database acknowledges the update.
11. The Web server creates a CPL script for MCI service.
12. The CPL script is posted to the SIP proxy of the user.

Figure 4 shows an example of VoiceXML script that retrieves the user information needed for service activation. A subdialogue is used to get the service code and a form dialogue captures the restricted area code (359259). A subdialog is a mechanism for decomposing complex sequences of dialogs to better structuring, or to create reusable components. In the example, service activation may involve gathering several pieces of information, such as service code, the restricted area code, and others. A customer care service might be structured with several independent applications that could share this basic building block, thus it would be reasonable to construct it as a subdialog.

[1] Hyper Text Transfer Protocol
[2] Intelligent Network Application Protocol

```
<?xml version="1.0"?>
<vxml version="1.0">
  <form id="callcenter">
  <var name="service_code"/>
  <subdialog name="serviceinfo"
                src="srvc_info.vxml#basic">
  <filled>
    <assign name="service_code" expr="srvcinfo.code"/>
  </filled>
  </subdialog>
  <field name="area_code" type="digits">
     <prompt> What is the restricted area code?</prompt>
     <filled>
       <submit next="/cgi-bin/updatemci"/>
     </filled>
  </field>
 </form>
</vxml>

<?xml version="1.0"?>
<vxml version="1.0">
<form id="basic">
  <filled>
    <field name="code" type="digits">
     <prompt>
       What is the code of the service you want to activate?
     </prompt>
    </field>
    <return namelist="code"/>
  </filled>
</form>
</vxml>
```

Figure 4 A VoiceXML script for MCI service activation

Figure 5 shows an example of CPL script for the MCI service. If a call originates from area with 359259 code it is rejected and the information is logged. The SIP server should include information in the log, such as the time of the logged event information that triggered the call to be logged, and so forth.

```
<?xml version="1.0"?>
<cpl>
 <incoming>
  <address-switch field="origin" subfield="tel">
   <address subdomain-of="359259">
    <reject status="reject" reason="Not allowed to
          receive calls from 359259 subdomain"/>
     <log name=”rejectedcalls” comment=”origin”/>
   </address>
  </address-switch>
 </incoming>
</cpl>
```

Figure 5 A CPL script for MCI service

V. CONCLUSION

As the next generation network seemed to be IP-based SIP is seen as the future of call signaling and telephony. Many value-added services we are accustomed to use in the telephony networks can be implemented in IP networks with SIP.

Extremely complex telecom applications, as found in call centers, have become even more complex when integrating with e-mail and web applications for managing service profiles. For example, both call routing and e-mail routing to agents – based on various criteria such as queue length, call origin, time of day, customer ID – can be reduces to simple XML scripts when using SIP and CPL. CPL has restricted expressive power and can be used to describe decision structures for routing calls. As most of the services require user interaction CPL can be combined with VoiceXML to adapt the services to customer call-processing preferences.

SIP in conjunction of CPL and VoiveXML can implement any of IN features as those for numbering, routing, charging, access, and restriction.

REFERENCES

[1] Alan B. Johnston, SIP: Understanding the session Initiation Protocol, Artech House, 2004

[2] VoiceXML Forum, Voice eXtensible markup Language VoiceXML, 2000

[3] J. Lennox, X. Wu, CPL: A Language for User Control of Internet Telephony Services, RFC 3880, 2004

[4] Johan Zuidweg, Next Generation Intelligent Networks, Artech House, 2002