IMPLEMENTING INTELLIGENT NETWORK SERVICES BY SLPL

Ivaylo Ivanov Atanasov

Department of Telecommunications, Technical University of Sofia, 7 Kliment Ohridski st., 1000, phone: +359 2 965 2050, e-mail: iia@tu-sofia.bg

The paper presents features of a new XML-based language for service logic processing. The language is designed to support Open Service Access (OSA) interfaces. On contrary with the other mark-up languages for service creation it allows service logic to interfere on both call-related and call-unrelated events and provides means for outchannel signaling. The paper exploits language capabilities for implementing intelligent network services

Keywords: Open Service Access, markup languages, intelligent network services

1. INTRODUCTION

Services in next generation networks (NGN) are expected to be feature and content rich, highly customized and ubiquitous. The technology used to provide services will be independent of underlying network technology. This separation will be achieved through application programming interfaces (APIs). A promising technology for service delivering in NGN is Parlay/OSA (Open service access). Although intended for 3G mobile networks it is applicable to NGN domain also.

One of the ways for implementing OSA is by the use of markup languages. Several markup languages for service creation in NGN are proposed [1, 2]. Some of these languages like Call processing language (CPL) and VoiceXML are standardized. These markup languages can be used mainly in communication session control and none of them support the full spectrum of network capabilities exposed by OSA APIs. Oriented towards call control, the existing markup languages can not derive added value from mobility, charging, terminal capability, messaging and other network functions accessed through OSA interfaces.

In this paper a short introduction to a new markup language SLPL (Service logic processing language) is presented. SLPL is intended to create services combining network functionalities provided by OSA APIs. To illustrate the language capabilities it is discussed how traditional intelligent network services can be implemented in OSA environment and an example of SLPL service description is given.

2. SERVICE LOGIC PROCESSING LANGUAGE

SLPL is a markup language intended for NGN service creation. As the language is based on eXtensible Markup Language (XML), it is easy to use, highly customized, platform and programming language independent. These language features make SLPL appropriate in area of rapid application development. Further as the language

supports OSA interfaces that hide network specificity and protocol complexity from service layer, it is applicable to 3rd party service development.

Existing service creation languages are mainly oriented to call control, i.e. the initiation, manipulation and termination of communication sessions. On contrary, SLPL is designed to support the whole variety of OSA interfaces. The language allows services to derive added value from network functionalities such as data session control, mobility, user interaction, terminal capabilities, charging and so on.

Suggested service creation languages such as CPL, SCML and VoiceXML possess restricted expressive power for manipulation of communication sessions and provide high level of abstraction. SLPL allows handling with lower level details considering some telecommunication aspects as presented in OSA APIs. SLPL provides means for flow control, method invocation and returned result processing that draws the language close to programming languages.

Time constructions of SLPL make possible time measuring and supervision. To model 'real' time the language defines time-specific types and methods. These types and methods allow decisions to be made based on time, date, day of week, month of year and so on.

Telecommunication services become more customized and this requires support of great amount of data. Usually service data are organized in structures and are stored in databases. To access user and/or service specific data service logic needs to query database. SLPL provides methods for database access that allow data retrieval and modification.

3. INTELLIGENT NETWORK SERVICES IMPLEMENTED IN OSA ENVIRONMENT

In NGN the services and applications have the greatest revenue potential. It is difficult to predict which will be the most attractive application in the future, but the subscribers are expecting to use at least the services they are accustomed to.

Recently the subscribers benefit added value form intelligent network (IN) services. Initially, the IN concept was designed for fixed circuit switched networks, but later it was extended with features that adapt it to mobile networks. Following the computer-telephony integration some solutions are proposed for interworking between IN and IP-based networks.

Although developed for 3G mobile networks Parlay/OSA is applicable to NGN environment. OSA appears to create the pathway for future of IN. All IN services and features defined as a minimum set can be implemented also with OSA APIs. As the intention of developing SLPL is to support the palette of network functionalities accessed through OSA, it is possible to describe IN services and features using SLPL. SLPL provides means for OSA data and method definition and method invocation. The application accesses network function by invocation of methods that belong to network-side interfaces and receives results returned by methods of its interfaces.

All IN services can be implemented by OSA APIs using SLPL so these services can be available to users in IP-based networks.

Call related IN services implemented in OSA environment uses Call Control API. The IN call-unrelated services allow user interactions outside the context of a call, for example in mobile networks user authentication and location updating. These services implemented by OSA APIs use Mobility API. It is possible call-related and call-unrelated user interactions to use out-channel signaling connections (for example short message or TCP/IP connection). These IN services implemented by OSA APIs use Multimedia Messaging API or Data Session Control API. To allow service logic to interfere based on call-related or call-unrelated events it has to subscribe to receive notifications form the corresponding OSA APIs for those events.

Most of IN services may need originating or terminating user prompting and when implemented in OSA environment these services use User Interaction APIs. The IN services with special charging treatment are implemented in OSA using Charging API. IN service features for access control such as authentication, authorization and off-net access or restriction services like call screening and closed user group require service logic to consult with database. SLPL constructions for database access allow service logic to retrieve service-specific subscriber data from an external database.

4. AN EXAMPLE OF INTELLIGENT NETWORK SERVICE DESCRIBED IN SLPL

In order to give an example how IN services can be implemented in OSA environment using SLPL let us consider freephone service regarded as the "queen" of IN services. The freephone service allows the calling party to call free of charge. The called party pays for the call. The service becomes very powerful when combined with time- and origin-dependent routing and customer control.

Let us consider a company having a restaurant chain and delivering food at home. The company has several restaurants throughout the city with different working hours. The company is subscribed for freephone service and has one universal number for all restaurants throughout the city. When a customer dials the unique number he automatically reaches a nearby restaurant. The service is responsible for contacting the nearby restaurant. When all the restaurants are closed, the service routes the calls to an answering machine, which provides the callers with information about hours of operation.

The service exploits the following OSA interfaces:

- Call control: to accept freephone calls;
- Mobility: to locate the user if he made a mobile call;
- Charging: to apply reverse charging;
- User interaction: to play an announcement to the caller.

To access service data the service logic has to query an external data base.

Figure 1 shows the scenario for freephone delivery service with the use of OSA APIs.

1. The application creates an object implementing the IpAppCallControlManager interface.
2. The application subscribes for notification on freephone call events.
3. When a hungry customer dials the freephone number a message with the new call event is passes to IpAppCallControlManager interface.
4. The notification message is forwarded to the IpAppLogic.
5. The service logic looks up in an external database and retrieves the freephone delivery service data.
6. The service logic determines if it is a mobile call or a fixed call. If it is a mobile call then the service logic creates a new IpAppUserLocation object.
7. The application requests the location of the customer.
8. The result of the location request for the customer is passed to its callback object.
9. The customer location is forwarded to the service logic.
10. The service logic determines which nearby restaurants are. The service logic determines if the nearest restaurant is open. If it is closed, the service logic continues searching restaurants.
11. In case an open restaurant is found the call is routed there.
12. The result of the call being answered is passed to its callback object.
13. The previous message is forwarded to IpAppLogic.
14. In case all restaurants in the vicinity are closed the application creates a new UICall object.
15. A new UICall object is created.
16. An announcement is played informing the customer about the opening hours.
17. When the announcement is completed the application is informed.
18. The message is forwarded to the application.
19. The application releases the UICall.
20. When the call is ended the application is informed.
21. The notification on call ending is forwarded to the application.
22. The application creates a local object implementing the IpAppChargingSession interface.
23. The application orders the creation of a session.
24. A new charging session is created.
25. The application requests to charge the company for the call.
26. The payment is acknowledged.
27. The acknowledgment is forwarded to the application.
28. The application releases the session.

The skeleton of freephone delivery service logic described in SLPL is shown in Fig.2

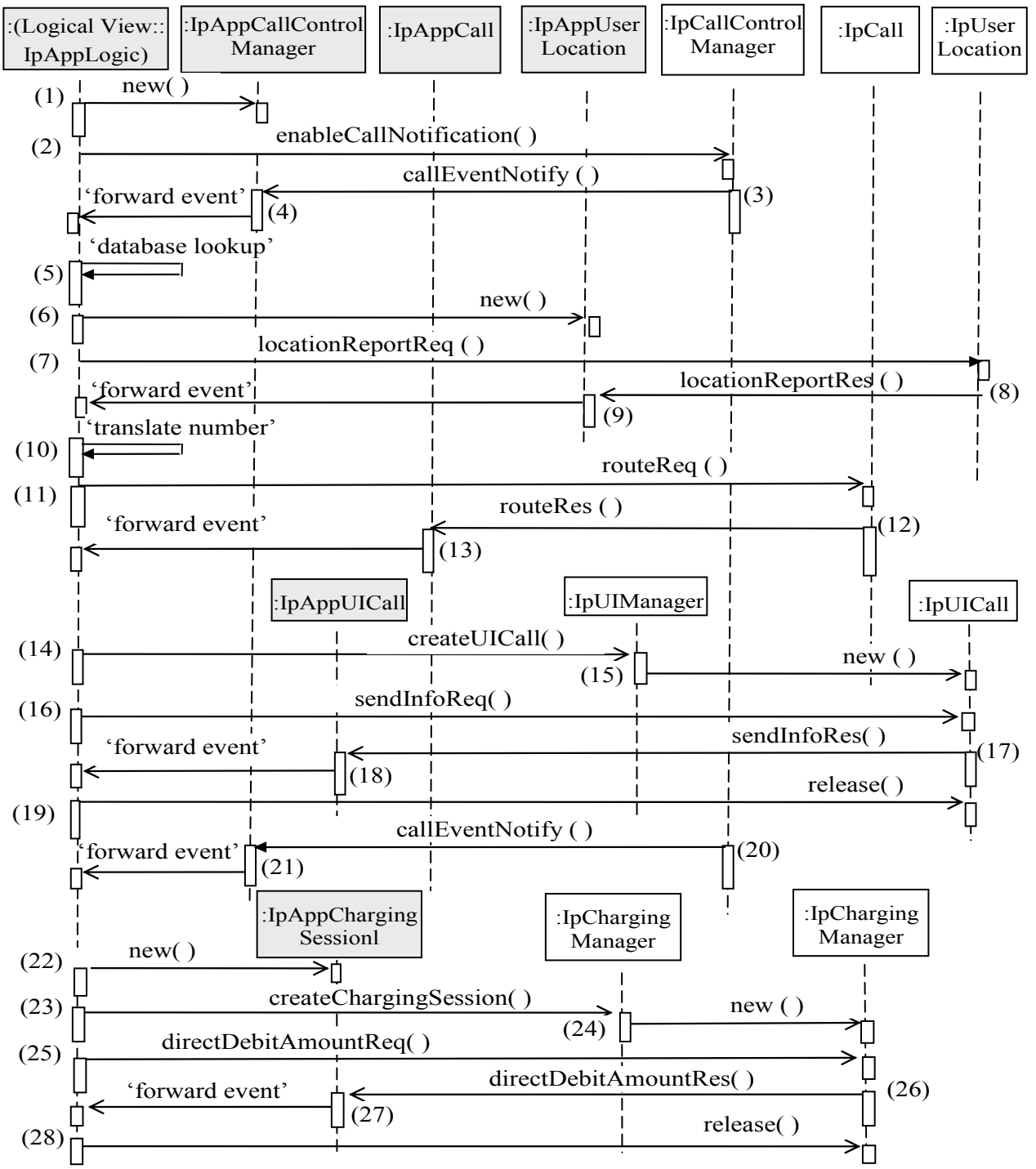

Fig. 1 Sequence diagram for freephone delivery service

5. CONCLUSION

SLPL suggests a mark-up approach to NGN service creation. In comparison with the existing standardized scripting languages that are call processing oriented, SLPL supports all OSA interfaces allowing service logic to interfere on call related and call-unrelated events. SLPL is protocol agnostic and can be used for 3rd party service

development. Traditional value-added services provided in intelligent networks can be implemented by SLPL and thus accessible also in IP-based networks. As the language is intended to be used with OSA APIs, services described in SLPL can benefit from network functions hiding network protocol complexity. These language features get telecommunication service development near to IT community and shorten time to market.

```
<logic>
  <define>
    <types/>
    <variables>
      <!—DEFINITION OF VARIABLES NEEDED FOR METHODS OF
          NETWORK-SIDE INTERFACES AND THEIR CALLBACKS-- >
    </variables>
    <methods>
      <!-- DEFINITION OF APPLICATION-SIDE INTERFACES  -->
    </methods>
  </define>
  <execute>
    <!-- FRAMEWORK AUTHENTICATION -->
    <!--  1.SET PARAMETERS. 2.INVOKE enableCallNotification -->
    <wait/> <!-- RECEIVE NOTIFICATION ON CALL EVENT -->
    <!--  1.DATABASE LOOKUP. 2.DETERMINE CALL TYPE -->
    <!--  1.SET PARAMETERS. 2.INVOKE locationReportReq -->
    <wait/><!-- RECEIVE LOCATION REPORT -->
    <!--  1.DETERMINE NEARBY RESTAURANTS. 2.TRANSLATE NUMBER -->
    <!--  1.SET PARAMETERS. 2.INVOKE routeReq -->
    <wait/><!-- RECEIVE ROUTING RESULT -->
    <!--  1.SET PARAMETERS. 2.INVOKE createUICall -->
    <!--  1.SET PARAMETERS. 2.INVOKE sendInfoReq -->
    <wait/><!-- RECEIVE SENDING INFO REPORT  -->
    <!--  1.SET PARAMETERS. 2.INVOKE release -->
    <wait/> <!-- RECEIVE NOTIFICATION ON CALL EVENT -->
    <!--  1.SET PARAMETERS. 2.INVOKE createChargingSession -->
    <!--  1.SET PARAMETERS. 2.INVOKE directDebitAmountReq -->
    <wait/> <!-- RECEIVE REPORT FOR PAYMENT -->
    <!--  1. SET PARAMETERS. 2.INVOKE release -->
  </execute>
</logic>
```

Fig.2 Skeleton of freephone delivery service logic described in SLPL

REFERENCES

[1] Bakker J-L, D.Tweedie and M. Umnehopa, *Evolving Service Creation; New developments in network Intelligence,* Teletronikk, 2004

[2]Bakker J, R. Jain, *Next Generation Service Creation Using XML Scripting Languages*, www.argreenhouse.com/papers/jlbakker/ bakker-icc2002.pdf

[3]I. Atanasov , *New XML-based Language for OSA User Interaction Interface Description*, TELECOM'2005, Varna, Bulgaria, Conference proceedings.

[4]Pencheva E., I. Atanasov, *New XML-based Language for OSA Mobility Interface Description*, TELECOM'2005, Varna, Bulgaria, Conference proceedings.

[5]Atanasov I., E. Pencheva, *Service creation using a new mark-up language*, TELSIKS'2005, Hish, Serbia and Monte Negro, Proceedings, pp 575-578

[6] Atanasov I., E. Pencheva, *A new service logic processing language*, ELECTRONICS'2005, Sozopol, Bulgaria, Proceedings.