Characterizing Semantic Warnings of Service Description in Call Processing Language on Internet Telephony

Pattara Leeraplute[1], Tomokazu Taki[1], Masahide Nakamura[2], Tohru Kikuno[1]
[1]Department of Informatics and Mathematical Science
Graduate School of Engineering Science Osaka University, Japan
Tel: +81-6-6850-6567 , Fax: +81-6-6850-6569
E-mail: pattara@ics.es.osaka-u.ac.jp
[2]Graduate School of Information Science
Nara Institute of Science and Technology
Tel: +81-743-72-5312 , Fax: +81-743-72-5319
E-mail: masa-n@is.aist-nara.ac.jp

Abstract: The Call Processing Language (CPL, in short), recommended in RFC 2824 of IETF, is a service description language for the Internet Telephony. The CPL allows users to define their own services, which dramatically improves the choice and flexibility of the users.

The syntax of the CPL is strictly defined by DTD (Document Type Definition). However, compliance with the DTD is not a sufficient condition for correctness of a CPL script. There are enough rooms for non-expert users to make semantical mistakes in the service logic, which could lead to serious system down.

In this paper, we present six classes of semantic warnings for the CPL service description: MF, IS, CR, AS, US, OS. For each class, we give the definition and its effects with an example script. These warnings are not necessarily errors. However, these warnings will help users to find ambiguity, redundancy and inconsistency in their own service description.

1. Introduction

As the Internet is widely spread in society, high-quality services with the Internet are required and developed, such as Video on Demand, e-learning, on-line banking and Web services. Among the various Internet services, this paper especially focuses on the *Internet telephony*[3], which is also called *Voice over IP*, (VoIP, in short). The Internet telephony has been widely studied and standardized at the protocol level (i.e., H323[5] by ITU-T, SIP[4] by IETF).

Now, the concern is shifting to the service level; how to provide supplementary services (e.g., call forwarding, voice mail, etc.) on the Internet telephony. One of the major issues is the *programmable service*, which allows users to define and create their own supplementary services. The *Call Processing Language* [2] (CPL, in short), based on XML, is recommended as a service description language in RFC2824 of the Internet Engineering Task Force (IETF) [1]. Users can deploy their own service just by putting the CPL scripts on the local VoIP server (called *signalling server*). This improves the range of user's choice and flexibility, significantly.

There is, however, a drawback of the programmable service. The service description of non-experts cannot always achieve the high quality. Also, users might make faults in the CPL scripts that lead to serious system down.

To cope with this problem, this paper tries to characterize *semantic warnings* of service description written in the CPL. As seen in many programming languages, the warnings are not necessarily errors. However, they could cause ambiguity, redundancy and inconsistency, which are often the major source of errors. We believe that the proposed warnings will help users to improve the quality of the CPL scripts.

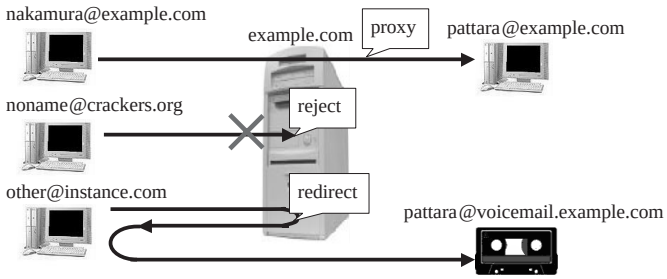

Figure 1. Behavior of the requirement

2. Describing services with CPL

Let us define a new service based on the following requirements, using CPL:

- I (`pattara@example.com`) want to receive incoming calls only from domain `example.com`.
- I want to reject all calls from malicious crackers (belonging to `crackers.org`).
- I want to redirect any other calls to my voice mail (`pattara@voicemail.example.com`).

Figure 1 summarizes the requirement. Figure 2 shows an implementation of the service. As in the XML, the CPL *tags* describe nested structures, starting from `<tag>` and ending with `</tag>`. `<tag />` is an abbreviation of `<tag> </tag>`. The first two lines are for the declaration of XML and DTD (Document Type Definition). The tag `<cpl>` means the start of a body of the CPL script. The portion surrounded by `<subaction> </subaction>` defines a *subaction*, which is a sub-routine called from the main-routine. `<incoming>` tag specifies actions activated when an incoming call is received.

Next, `<address-switch>` allows the CPL to have a conditional branch with respect to the addresses. In this example, the condition is extracted from the host address of the caller (`field= "origin" subfield=host`). If the host's domain matches `example.com` (`<address`

subdomain-of= "example.com">), then the location is set to sip:pattara@example.com, and the call is proxied there (<proxy />) . If the domain matches crackers.org, the call is rejected (<reject />). Otherwise, the subaction voicemail is called. In the subaction voicemail, the location is set to pattara@voicemail.example.com, and the call is redirected there. That is, the caller places the call again to the new address.

For a formal definition of CPL, please refer to the full specification [2].

3. Characterizing Semantic Warnings

The CPL is a relatively simple language, as it has no variables, loops, or ability to run external programs. This allows simple but strict syntax definition by the DTD, and minimizes such complex semantic errors as the ones in the general programming languages.

However, compliance with the DTD is not a sufficient condition for correctness of a CPL script. There are enough rooms for *non-expert users* to make various mistakes, which make the CPL scripts complex, ambiguous and inconsistent.

Here we propose six clasees to be considered as the semantic warnings. These might not be necessarily errors, but should be avoided. In the following subsections, we present a definition, effects and an example script for each warning class.

3.1 Multiple forwarding addresses (MF)

Definition: After multiple addresses are set by <location> tags, <proxy> or <redirect> comes.

Effects: The CPL allows calls to be proxied (or redirected) to multiple address locations by cascading <location> tags. However, if the call is redirected to multiple locations, then the caller would confuse to which address the next call should be placed. Or, if the call is proxied, a race condition might occur depending on the configuration of the proxied terminals. As a typical example, if a user simultaneously sets the forwarding address to his handy phone and voice mail that immediately answers the call. Then the call never reaches his handy phone.

Example CPL: Figure 3 shows an example. The user is setting the forwarding address to his handy phone pattara@mobile.example.com and voice mail pattara@voicemai.example.com, simultaneously. If the user configures the voice mail to immediately answer the call, then no call reaches the mobile phone.

3.2 Identical switches with the same parameters (IS)

Warning class: Identical switches with the same parameters (IS)

Definition: After a switch tag with a parameter, the same switch with the same parameter comes.

Effects: The CPL has no variables or no loop. So, a condition evaluated in the former switch tag never changes in the latter switch tag. Hence, the conditional branch specified in the latter switch is in vain, since the condition must have been evaluated already. This would increase the ambiguity of the CPL script.

Example CPL: Figure 4 shows an example. When a call arrives the user, this script will check the originator's host name. If it matches home.org, the call will be proxied to pattara@home.org. On the other hand, the originator's host name will be checked again if it matches home.org or not. If yes, this script tries to proxy the call to pattara@mobile.net. But in fact this proxy is never executed. The second switch is redundant and meaningless.

3.3 Call rejection in all paths (CR)

Definition: All execution paths terminate at <reject>.

Effects: No matter which path is selected, the call is rejected. No call processing is performed, and all executed actions and evaluated conditions are nullified. This is not a problem only when the user wants to reject all calls explicitly. However, complex conditional branches and deeply nested tags will make this problem difficult to be found, on the contrary to user's intention.

Example CPL: Figure 5 shows an example. By this script, any incoming call is rejected, no matter who is the originater: "anonymous", "pattara@example.com" or others. All actions and evaluated conditions are meaningless after all.

3.4 Address set after address switch (AS)

Definition: When <address> and <otherwise> tags are specified as outputs of <address-switch>, the same address evaluated in the <address> is set in the <otherwise> block.

Effects: The <otherwise> block is executed when the current address does not match the one specified in <address>. If the address is set as a new current address in <otherwise> block, then a violation of the conditional branch might occur. A typical example is that, after screening a specific address by <address-switch>, the call is proxied to the address, although any call to the address must have been filtered.

Example CPL: Figure 6 shows an example. When the user make a (outgoing) call, this script will check the destination of the call. The call should be rejected if the destination address is pattara@example.com, according to the condition specified in <address>. However, in the <otherwise> block, the call is proxied to pattara@example.com, which must have been rejected.

3.5 Unused Subactions (US)

Definition: Subaction <subaction id= "foo" > exists, but <subaction ref= "foo" > does not.

Effects: The subaction is defined but not used. The defined subaction is completely redundant, and should be removed to decrease server's overhead for parsing the CPL script.

Example CPL: Figure 7 shows an example. In this script, a subaction ”mobile” that was declared in the subsection part is not used in the body of the script. So, the unused subaction ”mobile” is redundant and should be removed.

3.6 Overlapped Conditions in Switches (OS)

Definition: The condition is overlapped among the multiple output tags of a switch.

Effects: According to the CPL specification, if there exist multiple output tags for a switch, then the condition is evaluated in the order the tags are presented, and the first tag to match is taken. If the conditions specified in the outputs are overlapped (or identical), then the former tag is always taken. In extreme cases, the latter tag is never executed, which is a redundant description.

Example CPL: Figure 8 shows an example. When a call reaches the subscriber, this script will check the destination of the call. If the destination's address contains `pattara`, the call will be proxied to his home telephone address. However, if the destination's address is `pattaraleelaprute`, this script tries to proxy the call to his mobile phone. But in fact, if the proxy to `pattara` has already occurred, proxy to `pattaraleelaprute` will never occur.

4. Conclusion

In this paper we discussed the six semantic warnings of service descriptions in the CPL: MF, IS, CR, AS, US, OS. Although the evaluation is needed, we believe that the proposed classes contribute to the quality improvement of the programmable service in CPL.

Our primary future work is to apply the proposed classes to the Feature Interaction problem in the Internet telephony, which is known as functional conflicts among multiple services. Also, there could be several other types of warnings to be found. The quantitative evaluation of the proposed classes is important research.

Acknowledgments

This work is partly supported by Grant-in-Aid for Encouragement of Young Scientists (No.13780234), from Japan Society for the Promotion of Science.

References

[1] J.Lennox and H.Schulzrinne,“Call processing language framework and requirements,” Request for Comments 2824, Internet Engineering Task Force,May 2000. http://www.ietf.org/rfc/rfc2824.txt?number=2824

[2] J.Lennox and H.Schulzrinne,“CPL:A Language for User Control of Internet Telephony Service,” Internet Engineering Task Force,Jan 2002. http://www.ietf.org/internet-drafts/draft-ietf-iptel-cpl-06.txt

[3] H.Schulzrinne and J.Rosenberg, “Internet Telephony: Architecture and protocols - an IETF perspective,” Computer Networks and ISDN Systems, vol.31, pp.237-255, Feb 1999.

[4] M.Handley, H.Schulzrinne, E.Schooler, and J.Rosenberg, “SIP:session initiation protocol,” Request for Comments 2543, Internet Engineering Task Force, Feb 2002. http://www.ietf.org/internet-drafts/draft-ietf-sip-rfc2543bis-09.txt

[5] ITU-T Recommendation H.323, “Packet-Based Multimedia Communications Systems,” February 1998.

```
<?xml version="1.0" ?>
<!DOCTYPE cpl PUBLIC "-//IETF//DTD RFCxxxx
                                CPL 1.0//EN" "cpl.dtd">

<cpl>
  <subaction id="voicemail">
    <location url="sip:pattara@voicemail.example.com">
      <redirect />
    </location>
  </subaction>

  <incoming>
    <address-switch field="origin" subfield="host">
      <address subdomain-of="example.com">
        <location url="sip:pattara@example.com">
          <proxy />
        </location>
      </address>
      <address subdomain-of="crackers.org">
        <reject />
      </address>
      <otherwise>
        <sub ref="voicemail" />
      </otherwise>
    </address-switch>
  </incoming>
</cpl>
```

Figure 2. Example of a CPL script

```
<?xml version="1.0" ?>
<!DOCTYPE cpl PUBLIC "-//IETF//DTD RFCxxxx
                                CPL 1.0//EN" "cpl.dtd">

<cpl>
  <incoming>
    <location url="sip:pattara@mobile.example.com">
    <location url="sip:pattara@voicemai.example.com">
      <proxy />
    </location>
    </location>
  </incoming>
</cpl>
```

Figure 3. Example of MF

```
<?xml version="1.0" ?>
<!DOCTYPE cpl PUBLIC "-//IETF//DTD RFCxxxx
                              CPL 1.0//EN" "cpl.dtd">

<cpl>
  <incoming>
     <address-switch field="origin" subfield="host">
        <address subdomain-of="home.org">
           <location url="sip:pattara@home.org">
             <proxy />
           </location>
        </address>
        <otherwise>
          <address-switch field="origin" subfield="host">
              <address subdomain-of="home.org">
                 <location url="sip:pattara@mobile.net">
                   <proxy />
                 </location>
              </address>
              <otherwise>
                 <location url="sip:pattara@office.com">
                   <proxy />
                 </location>
              </otherwise>
          </address-switch>
        </otherwise>
     </address-switch>
  </incoming>
</cpl>
```

Figure 4. Example of IS

```
<?xml version="1.0" ?>
<!DOCTYPE cpl PUBLIC "-//IETF//DTD RFCxxxx
                              CPL 1.0//EN" "cpl.dtd">

<cpl>
  <incoming>
    <address-switch field="origin">
      <address is="anonymous">
        <reject status="reject"
            reason="I don't accept anonymous calls" />
      </address>
      <address is="sip:pattara@example.com">
        <reject status="reject"
            reason="I don't accept call from Pattara" />
      </address>
      <otherwise>
        <reject status="reject"
            reason="I don't accept call from anyone" />
      </otherwise>
    </address-switch>
  </incoming>
</cpl>
```

Figure 5. Example of CR

```
<?xml version="1.0" ?>
<!DOCTYPE cpl PUBLIC "-//IETF//DTD RFCxxxx
                              CPL 1.0//EN" "cpl.dtd">

<cpl>
  <outgoing>
    <address-switch field="destination">
      <address is="sip:pattara@example.com">
        <reject status="reject"
            reason="I don't call Pattara" />
      </address>
      <otherwise>
        <location url="sip:pattara@example.com">
            <proxy/>
        </location>
      </otherwise>
    </address-switch>
  </incoming>
</cpl>
```

Figure 6. Example of AS

```
<?xml version="1.0" ?>
<!DOCTYPE cpl PUBLIC "-//IETF//DTD RFCxxxx
                              CPL 1.0//EN" "cpl.dtd">

<cpl>
  <subaction id="mobile">
    <location url="sip:jones@mobile.example.com" >
      <proxy />
    </location>
  </subaction>

  <incoming>
    <location url="sip:jones@example.com">
      <proxy />
    </location>
  </incoming>
</cpl>
```

Figure 7. Example of US

```
<?xml version="1.0" ?>
<!DOCTYPE cpl PUBLIC "-//IETF//DTD RFCxxxx
                              CPL 1.0//EN" "cpl.dtd">

<cpl>
  <incoming>
    <address-switch field="destination" >
      <address contains="pattara">
        <location url="sip:pattara@home.example.com">
          <proxy />
        </location>
      </address>
      <address is="pattaraleelaprute">
        <location url=
              "sip:pattaraleelaprute@home.example.com">
          <proxy />
        </location>
      </address>
    </address-switch>
  </incoming>
</cpl>
```

Figure 8. Example of OS