Network Working Group
Internet-Draft
Expires: August 18, 2005

X. Wu
H. Schulzrinne
Columbia University
February 14, 2005

LESS: Language for End System Services in Internet Telephony
draft-wu-iptel-less-00

Status of this Memo

Copyright Notice

Abstract

In Internet telephony, end systems can take a large role in providing services, especially in networks without pre-configured infra-structure, such as peer-to-peer networks. Since we believe that end system services differ in their requirements from network services, we define a new service creation scripting language called the Language for End System Services (LESS). LESS inherits many

characteristics from the Call Processing Language (CPL). It contains commands and events for direct user interaction and the control of media applications. This document defines the basic elements of LESS and several commonly used LESS extensions.

Table of Contents

1 Introduction

One of the key promises of Internet telephony lies in the ability of developing and deploying innovative services rapidly and efficiently. Internet telephony services are not limited to those performed in servers operated by service providers; end systems can play a much larger role in providing services than in traditional telephone networks.

We define end systems as the systems that are located at the end of a session signaling routing path and can initiate or accept session setup messages. By this definition, end systems are not limited to the user-operated systems, instead, they can also be conference servers. Accordingly, we define network servers as the entities that perform session message routing. We believe it is necessary to define a service creation language specifically for end systems because end system services are different from network services.

There are services that are clearly implementable only in end systems or in network, but most can be moved to either location, or split between the two. Providing services in user-operated end systems has the advantage that on-the-spot interaction with users is much easier. Network services can only interact via protocol messages and possibly media content, rather than GUIs. It might be possible to incorporate user interaction into scripts running in servers, but it is likely to be far more cumbersome and subject to network delays.

In Internet telephony, usually end systems are the only entities where signaling and media flows converge. This is different from the legacy PSTN. Thus, any service that requires interaction with user media is likely to be easier to implement in end systems. The more detailed analysis on the differences between end system services and network services can be found in our articles [11] [12].

Because of the differences between end system services and network services, we define the Language for End System Services specifically for end system service creation. Section 2 introduces the design decisions for LESS. Section 3 briefly describe the categories of elements in LESS, we then provide more detailed description of the elements in each category in the following sections. Extensibility is very important for a service creation language. Section 9 defines how to write LESS extensions. Several commonly used extensions are defined in the following sections. Section 17 provides service examples written in LESS. Section 19 shows the XML schema of the language.

1.1 Conventions of This Document

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [1] and indicate requirement levels for compliant LESS implementations.

> Some paragraphs are indented, like this; they give motivations of design choices, advice to implementors, or thoughts on future development of or extensions to LESS. They are not essential to the specification of the language, and are non-normative.

2 LESS design decisions

The goal of the language is to allow end users to program their own communication services with little training in a graphical service creation environment. To achieve the goal, the language must represent a high-level abstraction of communication behaviors. The elements in the language must have semantic meanings. The language has to be simple and safe. The Call Processing Language (CPL) [2] has all the above features. However, CPL is originally designed for network servers, such as SIP proxy servers. It lacks of the functions for end system services, e.g., accepting calls, making outgoing calls, and interacting with users. CPL does not have variables, which is very important for call decision making and user interactions. We have LESS inherit the basic structure and many elements from CPL and enhance it with more elements for end system services.

In this section, we first provide the high-level structure of LESS. We then analyze the simplicity and safety of the design rules.

2.1 High-level Structure

As shown in Figure 1, we consider that every service starts with a trigger invoked, such as an incoming call, an event notification, a user interaction event, or a timer event. The service will then check its current context, such as user status, device status, and the status related to the trigger. We name the check points as switches, which can branch call decisions. Once a service gets its matching condition, it will perform pre-defined actions, e.g., accepting an incoming call, holding an existing call, or sending an instant message. We use modifiers to provide parameters to actions, e.g., media modifier to define media information of a call action. Additional actions may get performed based on the result of an action.

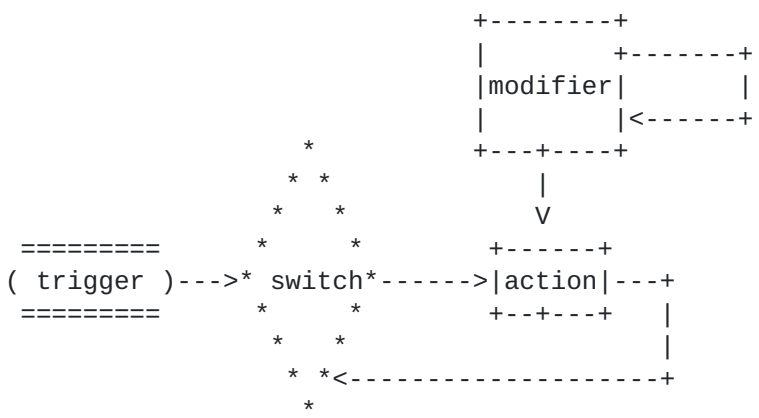

Figure 1: The high-level structure of LESS

The above high-level abstraction makes LESS having a tree-like structure, as shown in Figure 2. The tree-like structure makes LESS easy to analyze and safe for service programming.

2.2 Simplicity

"The fewer concepts to understand in a language the better" [13], though, in some cases two smaller concepts might be simpler and more flexible than one more powerful but complicated concept. For example, separating commonly used action arguments, such as people's contact locations, as modifiers is better than putting the arguments into every action.

"Simplicity enters in four disguises: uniformity (rules are few and simple), generality (a small number of general functions provide as special cases of more specialized functions), familiarity (familiar symbols and usages are adopted whenever possible), and brevity (economy of expression is sought)" [14]. For LESS, we consider analyzability is more important than brevity because LESS is an XML-based language designed for communication services and has people without programming experience as its target users. In communication service creation, users may encounter feature interaction [15] problems, how to analyze and solve feature interactions is essential for users to create usable services. We discuss the familiarity, generality, uniformity, and analyzability of LESS below.

2.2.1 Familiarity

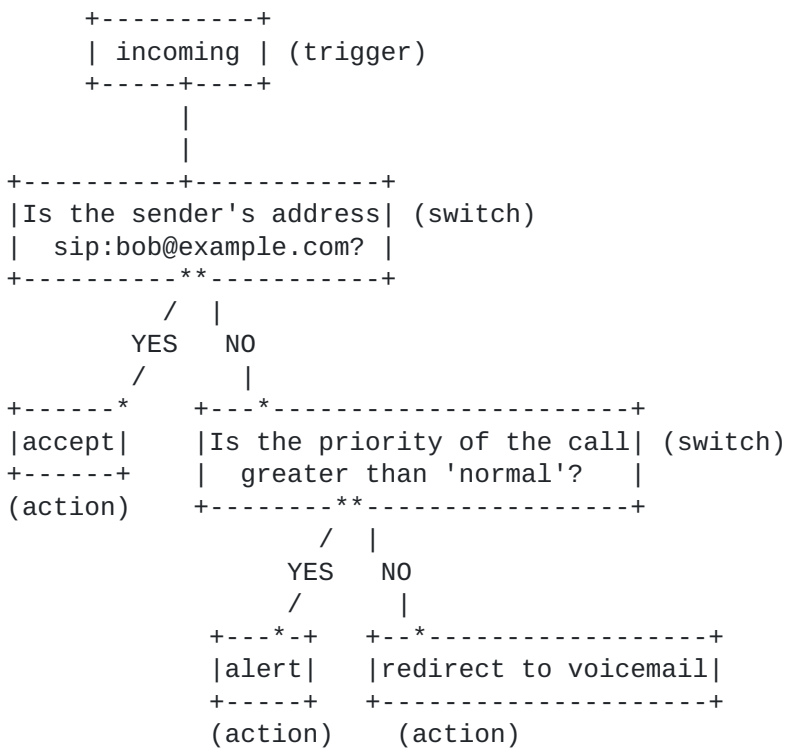

Figure 2: Sample decision tree

The familiarity requires a language not to violate common use of notation. We choose to use the Extensible Markup Language (XML) [3] to define LESS, uses the datatypes defined in XML schema. By this way, the elements defined in LESS all have semantic meaning and easy for users to understand and remember.

2.2.2 Generality

The generality requires a language to have a small number of general elements. The special elements of the language can be introduced by specializing the general elements. As illustrated in Section 2.1, LESS has only four kinds of elements, namely triggers, switches, actions, and modifiers. Every new element defined in LESS extensions must fall in one kind of the elements. In another words, in LESS's XML schema, LESS will have four basic abstract elements, namely 'trigger', every element must be a substitutionGroup of one of the four abstract elements.

2.2.3 Uniformity

The uniformity requires a language with few and simple rules. There are only four rules in LESS. The rules can be summarized below:

Trigger rule: A LESS script gets invoked if and only if one of its triggers get matched. A trigger must be the root of a decision tree. Two triggers with the same name and attributes cannot appear in the same script.

Switch rule: Only switches can branch call decisions. One switch have no more than two branches. A switch must be a child of a trigger or another switch in a decision tree.

Action rule: Only actions can change call status and call context and only actions can be LESS decision tree leaves. There can be subsequent actions after one action. No triggers can follow an action.

Modifier rule: A modifier can only be used as the parent element of actions to enforce the actions.

2.2.4 Analyzability

Another aspect of language simplicity is how easy to analyze a program written by the language. For LESS, we are more interested in how LESS handling feature interaction problems in communication services. As discussed in our technical report "Feature Interactions in Internet Telephony End Systems" [15], we can handle feature interaction problems among LESS scripts by the action conflict tables and the tree merging algorithm defined in the technical report. The process is straight-forward, and can also help to solve detected interactions.

2.3 Safety

Since LESS will be used by people without programming experience, we may expect errors that seem naive by experienced programmers. We should also expect that people may use third-party created malicious service scripts. The language design should help people to prevent errors and protect people from malicious scripts. The error prevention mechanisms for LESS fall into two categories, one is to put restrictions on the language itself, the other is to put restrictions on LESS engines. We are more interested in the restrictions on the language itself since that is directly related to the language design. The restrictions on LESS engines may cause service portability problem so we should try to prevent errors in the language design whenever possible.

2.3.1 Type safety

Type checking is a very effective way in catching programming errors, from the trivial errors, such as misspelled identifiers, to the fairly deep errors, such as violations of data structure invariants.

LESS is an XML-based language and uses XML schema [4] to define its elements. "The strong typing mechanism in XML Schema, along with the large set of intrinsic types and the ability to create user-defined types, provides for a high level of type safety in instance documents. This feature can be used to express more strict data type constraints, such as those of attribute values, when using XML Schema for validation." [4]

In LESS, there are no user defined variables so only static type checking is needed. There are many advantages to have a statically type-checked language. Static type checking can provide earlier, and usually more accurate information on programmer errors. It can eliminate the need for run-time checks, which may slow program execution. Statically type-checked language may be less expressive than dynamically type-checked language, however, since LESS is not designed to handle all communication services and targets to end users, safety is more important than expressiveness.

2.3.2 Control flow safety

LESS has a tree-like structure for call decision making. LESS does not allow a tree node to back-reference to its ancestors or itself. This means there is no loop or recursion in LESS scripts, and will exclude the possibility of non-terminating or non-decidable LESS scripts. However, LESS still provides some mechanisms for repeated events handling. The timer trigger can get invoked periodically. The lookup modifier implies that follow-up actions are applied to every location in the result set.

Run-time feature interactions may confuse users. The LESS trigger rule ensures that a specific trigger can appear no more than once in a LESS script thus helps to avoid run-time feature interactions in the LESS script. The LESS service creation environment should help users to merge multiple service scripts into one as illustrated in the technical report on feature interaction handling in LESS [15].

2.3.3 Memory access

A language can be described as unsafe in that the language allows some means to access memory directly. In LESS, there are no pointers, no direct memory accesses, and even no user-defined variables. LESS has maximum string length defined in its XML schema for every element with string type to prevent buffer overflow attack. If a user put a very long string in a service script, the service script will be

considered invalid when checking against the LESS XML schema.

2.3.4 LESS engine safety

LESS engine developers can do several things to enhance the safety of LESS scripts. The developers must make sure that every trigger has default behaviors defined. If a trigger gets invoked in a LESS engine, but there is no actions defined for the matching context, the LESS engine must perform the default actions for that trigger. This ensures every LESS script is decidable.

LESS is different from CPL [2] when dealing with user trustiness. CPL is designed for running on signaling servers, such as SIP proxy servers. The safety consideration for CPL ensures that semi-trusted users cannot create malicious or incompetent service scripts to interrupt other users' services, including crashing the server, revealing security-sensitive information, and causing denial of service. While LESS is used on users' end devices, usually, it does not need to handle the interference of other users. LESS engine need to ensure safe resource usages, such as CPU usage. It needs to define how deep a decision tree can be, what is the minimum interval for timer trigger, and whether it can control multimedia devices.

2.3.5 Using third-party created or auto-generated service scripts

Sometimes, end users may not be aware of a new service or they may not know how to create a service, the service scripts auto-generated by feature learning, or created by third-parties, may bring great conveniences to users. However, end users have to check the safety of the third-party created or auto-generated scripts, for example, to make sure that the scripts will not crash their systems or forward their calls to unwanted parties.

The safety consideration discussed in previous sections can be applied to third-party or auto-generated service scripts. In addition, the simplicity and the tree-like structure of LESS make it viable to convert any valid LESS scripts into graphical representation of decision trees. This is a big advantage of LESS for end users to do safety checking. Users do not have to check LESS programs line by line, instead, they can watch the graphical representation and easily find out what actions a service script will perform.

3 LESS elements

As we discussed earlier, LESS has four kinds of elements, namely triggers, switches, actions, and modifiers.

A trigger is an entry point to every service. It is the same as the toplevelactions defined in CPL. The basic LESS definition has two triggers, "incoming" (Section 5.1.1) and "timer" (Section 5.1.2). Note that we do not have "outgoing" trigger, which is defined in CPL. Because in end systems, an outgoing call is usually triggered by some user interactions, e.g., pressing a call button. This is different from that in an outbound server, in which "outgoing" triggered by receiving an outgoing call signaling message. We defined user interaction extension, which has one trigger, "UI:command" (Section 12.1.1). The event handling extension has two triggers, "Event:notification" (Section 14.1.2) and , "Event:subscription" (Section 14.1.1). The instant messaging extension has one trigger, "Message:message" (Section 13.1.1).

Switches branch call decisions. This is the same as CPL switches. The basic LESS definition has six switches, namely "time-switch" (Section 5.2.1), "address-switch" (Section 5.2.2), "language-switch" (Section 5.2.5), "priority-switch" (Section 5.2.3), "string-switch" (Section 5.2.4), and "status-switch" (Section 5.2.6). The location-based service extension has two switches, "where-switch" (Section 15.1.1) and "where-relation-switch" (Section 15.1.2). The event handling extension has one switch, "event-switch" (Section 14.2.1).

Actions represent users' decision for trigger handling. This is similar to CPL actions, but we allow multiple actions be executed in parallel and in sequential. Each action has a "next" output. Actions in an action's "next" output will be executed after the action. If multiple actions have the same parent tag, they are executed in parallel.

The basic LESS definition has six signaling actions, namely "accept" (Section 5.4.1), "reject" (Section 5.4.2), "redirect" (Section 5.4.3), "authenticate" (Section 5.4.4), "call" (Section 5.4.5), and "terminate" (Section 5.4.6). It also has three non-signaling actions, "mail" (Section 5.4.7) "log" (Section 5.4.8), and "wait" (Section 5.4.9).

We defined a mid-call handling extension. It has two actions, "Midcall:transfer" (Section 11.1.1), and "Midcall:merge" (Section 11.1.2). The user interaction extension has two actions, "UI:alert" (Section 12.2.1). "UI:getinput" (Section 12.2.2). The instant messaging extension has one action, "IM:sendmsg" (Section 13.2.1). The event handling extension has five actions, "Event:subscribe" (Section 14.3.4), "Event:notify" (Section 14.3.5), "Event:approve" (Section 14.3.1), "Event:deny" (Section 14.3.2), and "Event:defer" (Section 14.3.3). The queue handling extension has two actions, "Queue:enqueue" (Section 16.1.1), and "Queue:dequeue" (Section 16.1.2).

Modifiers provide action parameters. We have three modifiers defined in basic LESS definition, namely "location" (Section 5.3.1), "lookup" (Section 5.3.2), and "remove-location" (Section 5.3.3). In media handling extension, we defined "media" (Section 10.1.1) modifier.

4 Variables

LESS does not allow user-defined variables, however, it allows several pre-defined variables to retrieve system information, user information, agent information, trigger information, and action information. The variables can be used in descriptive messages, such as the reason parameter of a reject message. We use the format 'variable-name' to represent a variable in a LESS script. For example, in a "reject" action, we can provide its reason parameter as below.

```
<reject code="486" reason="Sorry, I am {user.activity}">
```

4.1 System information

system.bandwidth represents the bandwidth of the end system running the script.

4.2 User information

user.presence represents script owner's online/offline information.

user.activity represents script owner's current activity. Activity values are defined as in the Rich Presence Extensions to the Presence Information Data Format (RPID) [5].

user.mood represents script owner's current mood. Mood values are defined as in RPID.

user.location represents script owner's physical location.

user.language represents script owner's preferred language.

4.3 Agent information

agent.number-of-calls represents the number of existing calls in the user agent running the script.

agent.media-capabilities represents the media capabilities in the user agent running the script.

4.4 Trigger information

trigger.origin represents the originator of a trigger's signaling message.

trigger.destination represents the destination of a trigger's signaling message.

trigger.priority represents the priority of a trigger's signaling message.

trigger.timestamp represents when the signaling message is received.

trigger.subject represents the subject of a trigger's signaling message.

4.5 Action information

action.last-action-result represents the return value of last actions. Every action may return a value. In this document, if the return value is not specified, the return value will be empty.

5 Basic LESS elements

5.1 Triggers

5.1.1 "incoming" trigger

The "incoming" trigger is the same as that defined in CPL [2], Section 2.1.

5.1.2 "timer" trigger

The "timer" trigger is to handle timer-based events, e.g., automatically making an outgoing call at a specific time. The "timer" trigger has the same attributes as those defined in "time-switch" (Section 5.2.1). Their syntax is shown in Figure 3.

```
   Trigger:  "timer"
Parameters:  "dtstart"     Start of interval (RFC 2445 DATE-TIME)
             "dtend"       End of interval (RFC 2445 DATE-TIME)
             "duration"    Length of interval (RFC 2445 DURATION)
             "freq"        Frequency of recurrence ("secondly",
                           "minutely", "hourly", "daily",
                           "weekly", "monthly", or "yearly")
             "interval"    How often the recurrence repeats
             "until"       Bound of recurrence (RFC 2445 DATE-TIME)
             "count"       Number of occurrences of recurrence
             "bysecond"    List of seconds within a minute
             "byminute"    List of minutes within an hour
             "byhour"      List of hours of the day
             "byday"       List of days of the week
             "bymonthday"  List of days of the month
             "byyearday"   List of days of the year
             "byweekno"    List of weeks of the year
             "bymonth"     List of months of the year
             "wkst"        First day of the work week
             "bysetpos"    List of values within
                           set of events specified
```

Figure 3: Syntax of the "timer" trigger

5.2 Switches

5.2.1 "time-switch"

The "time-switch" is the same as that defined in CPL [2], Section 4.4.

5.2.2 "address-switch"

The "address-switch" is the same as that defined in CPL [2], Section 4.1.

5.2.3 "priority-switch"

The "priority-switch" is the same as that defined in CPL [2], Section 4.5.

5.2.4 "string-switch"

The "string-switch" is the same as that defined in CPL [2], Section

4.2.

5.2.5 "language-switch"

The "language-switch" is the same as that defined in CPL [2], Section 4.3.

5.2.6 "status-switch"

Status-switches allow a LESS script to make decisions based on the status of the script owner, or other people whose status can be acquired. The syntax of "status-switch" is summarized in Figure 4.

Node:	"status-switch"	
Outputs:	"status"	Specific status to match
Parameters:	"uri"	The principal's URI
	"status-name"	The name of the status to check. Available values are "active-calls", "activity", "mood", and "presence".
Output:	"status"	
Parameters:	"greater"	Used for "active-calls"
	"less"	Used for "active-calls"
	"equal"	Used for "active-calls"
	"is"	Exact match, not for "active-calls"
	"contains"	Not for "active-calls"

Figure 4: Syntax of the "status-switch" node

"status-switch" takes two parameter. "uri" indicates the URI of the principal for status checking. If omited, the principal for checking will be the script owner. "status-name" indicates what status to check. It has four possible values:

"active-calls" checks for ongoing active calls.

"presence" checks for online/offline status of the principal.

"mood" checks for the mood of the principal.

"activity" checks for the activity of the principal.

The "status" output takes five parameters, indicating different conditions to match. The "greater", "less", and "equal" can only be

used for "active-calls", indicating whether the number of active calls is greater than, less than, or equal to a specific value. The "is" parameter takes a list of strings for its value. User status MUST equal to all the strings in the list for being matched. The "contains" parameter also takes a list of strings for match. if user status equals to any string in the list, the switch get matched.

5.3 Modifiers

5.3.1 "location" Modifier

The "location" modifier is the same as that defined in CPL [2], Section 5.1.

5.3.2 Location Lookup

The "lookup" modifier is the same as that defined in CPL [2], Section 5.2.

5.3.3 Location Removal

The "remove-location" modifier is the same as that defined in CPL [2], Section 5.3.

5.4 Actions

As we discussed before, LESS allows multiple actions get executed sequentially or in parallel. Every LESS action will have a default output "next". In the following sections, we will omit this default output for the syntax of actions.

5.4.1 "accept"

"accept" actions cause the user agent to accept an incoming call. Their syntax is shown in Figure 5.

```
                    Node:  "accept"
                 Outputs:  None
               Next node:  None
              Parameters:  None
```

Figure 5: Syntax of the "accept" node

There is no parameters for the "accept" action. We can use "media"

modifier to change "accept" action's media parameters, as described in Section 10.1.1.

5.4.2 "reject"

The "reject" modifier is the same as that defined in CPL [2], Section 6.3.

5.4.3 "redirect"

The "redirect" modifier is the same as that defined in CPL [2], Section 6.2.

5.4.4 "authenticate"

"authenticate" actions cause user agents to authenticate local users or remote users. Their syntax is shown in Figure 6.

```
      Node:  "authenticate"
   Outputs:  "success"       Next node if authentication succeeded
             "failure"       Next node if authentication failed
Parameters:  "method"        Authentication method to use. Possible values
                             are "basic" and "digest".
             "realm"         The realm for authentication challenge.
             "user"          Required user for authentication.
             "target"        Which party to authenticate. Possible values
                             are "remote" and "local".

    Output:  "success"
Parameters:  none

    Output:  "failure"
Parameters:  none
```

Figure 6: Syntax of the "authenticate" node

"authenticate" action takes four arguments: "method", "realm", "user" and "target". The default value for "method" is "digest", which representing digest authentication.

The action has two outputs, "success" and "failure", handles the situation under which the authentication succeeds or fails, accordingly.

5.4.5 "call"

"call" actions cause the user agent to make an outgoing call. Their syntax is shown in Figure 7.

```
      Node:  "call"
   Outputs:  "accepted"     Next node if call attempt get accepted.
             "busy"         Next node if call attempt returned "busy".
             "failure"      Next node if call attempt failed.
             "noanswer"     Next node if call attempt was not
                            answered before timeout.
             "redirection"  Next node if call attempt was redirected.
Parameters:  "timeout"      Time to try before giving up call attempt.

    Output:  "accepted"
Parameters:  none

    Output:  "busy"
Parameters:  none

    Output:  "noanswer"
Parameters:  none

    Output:  "failure"
Parameters:  none

    Output:  "redirection"
Parameters:  none
```

Figure 7: Syntax of the "call" node

The "call" action has one parameter, "timeout", which specifies the time, as a positive integer number of seconds, to wait for the call attempt. We can also use "media" modifier to change "call" action's media parameters, as described in Section 10.1.1.

The "call" action has five outputs. The "accepted" is followed if the call was accepted; the "busy" is followed if the call was busy; the "noanswer" is followed if the call was not answered before the "timeout" parameter expired; the "failure" is followed if the call failed for any other reason; the "redirection" is followed if the call was redirected.

5.4.6 "terminate"

"terminate" actions cause the user agent to terminate one or more ongoing sessions or call attempts. Their syntax is shown in Figure 8.

Node:	"terminate"	
Outputs:	None	
Next node:	None	
Parameters:	"uri"	Select calls to terminate based on the "uri" list.
	"subject"	Select calls to terminate based on the "subject" list.
	"calls"	Possible values are "all", "last", and "this". Default value is "this".

Figure 8: Syntax of the "terminate" node

There are three parameters for the "terminate" action, which define how to select calls to terminate. If "uri" parameter presented, all calls to one of the "uri" values will get terminated. If "subject" parameter presented, all calls with one of the "subject" values will get terminated. If the "calls" parameter is "all", all calls matching "uri" and "subject" parameters will get terminated, if it is "last", the last established session will get terminated, if it is "this", the calls associated with the trigger event of the script will get terminated.

5.4.7 "mail"

The "mail" action is the same as that defined in CPL [2], Section 7.1.

5.4.8 "log"

The "log" action is similar to that defined in CPL [2], Section 7.2. But we added a new parameter for recording a call. The recording parameter makes it easy to associate a log with a recorded call. Their new syntax is shown in Figure 9.

```
            Node:  "log"
         Outputs:  None
       Next node:  Any node
      Parameters:  "name"       Name of the log file to use
                   "comment"    Comment to be placed in log file
                   "recording"  The URI to place recording files
```

Figure 9: Syntax of the "log" node

5.4.9 "wait"

"wait" actions cause the user agent to perform subsequent actions in the "next" output after a duration of time. Their syntax is shown in Figure 10.

```
                  Node:  "wait"
               Outputs:  None
             Next node:  Any node
            Parameters:  "duration"  Time to wait
```

Figure 10: Syntax of the "wait" node

6 Subactions and Ancillary Information

Subactions and ancillary information are the same as those defined in CPL Section 8 and Section 9.

7 Security Considerations

LESS shares the same security considerations as CPL, as defined in CPL document [2], Section 13.

8 IANA Considerations

This document registers a new MIME type, application/less+xml, and a new URN per RFC 2141 [6], RFC 2648 [7], and RFC 3688 [8].

The XML namespace urn:ietf:params:xml:ns:less will only refer to the version of LESS in this document and will not change. Any LESS enhancements MUST be made by extensions and MUST have different

namespaces.

8.1 URN Sub-Namespace Registration for urn:ietf:params:xml:ns:less

URI: urn:ietf:params:xml:ns:less

Registrant Contact: Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
    "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
     content="text/html;charset=iso-8859-1"/>
  <title>Language for End System Services Namespace</title>
</head>
<body>
  <h1>Namespace for the Language for End System Services</h1>
  <h2>urn:ietf:params:xml:ns:less</h2>
  <p><a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
END
```

8.2 Schema registration

This specification registers XML Schema for LESS, as per the guidelines in [8].

URI: urn:ietf:params:xml:schema:less

Registrant contact:
Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

XML: The XML can be found in Section 19.1.

8.3 MIME Registration

As an XML type, LESS's MIME registration conforms with "XML Media Types," RFC 3023 [9].

MIME media type name: application

MIME subtype name: less+xml

Mandatory parameters: none

Optional parameters: charset
As for application/xml in RFC 3023.

Encoding considerations: As for application/xml in RFC 3023.

Security considerations: See Section 7, and Section 10 of RFC 3023.

Interoperability considerations: Different user agents may use incompatible address types. However, all potential interoperability issues should be resolvable at the time a script is uploaded; there should be no interoperability issues which cannot be detected until runtime.

Published specification: This document.

Applications which use this media type: SIP user agents.

Additional information:

Magic number: None

File extension: .less or .xml

Macintosh file type code: "TEXT"

Person and e-mail address for further information:
Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

Intended usage: COMMON

Author/Change Controller: The IETF.

9 LESS Extensions

User agents MAY support additional LESS features beyond those listed in this document. LESS extensions are indicated by XML namespaces [10]. Every extension MUST have an appropriate XML namespace assigned to it. The XML namespace of the extension MUST be different from the XML namespace defined in Section 15.2. The extension MUST NOT change the syntax or semantics of the basic LESS schema defined in this

document. All XML tags and attributes that are part of the extension MUST be appropriately qualified so as to place them within that namespace.

A LESS script SHOULD NOT specify any namespaces it does not use. For compatibility with non-namespace-aware parsers, a LESS script MAY omit the base LESS namespace for a script which does not use any extensions.

A LESS server MUST reject any script which contains a reference to a namespace which it does not understand. It MUST reject any script which contains an extension tag or attribute which is not qualified to be in an appropriate namespace.

In the XML schema of LESS, we introduce four abstract elements, namely `trigger', `switch', `action', and `modifier', which accordingly have the abstract type `TriggerType', `SwitchType', `ActionType', and 'ModifierType'. Any trigger in a LESS extension MUST be defined as the substitutionGroup of the abstract `trigger' element, and has the type extended from the `TriggerType'. Any switch in a LESS extension MUST be defined as the substitutionGroup of the abstract `switch' element, and has the type extended from the `SwitchType'. Any action in a LESS extension MUST be defined as the substitutionGroup of the abstract `action' element, and has the type extended from the `ActionType'. Any modifier in a LESS extension MUST be defined as the substitutionGroup of the abstract `modifier' element, and has the type extended from the `ModifierType'.

We defined several commonly used extensions that are very important for LESS-based services as below.

10 Media handling extension

Media handling extensions define the elements that can manipulate session media information.

10.1 Modifiers

10.1.1 "Media:media" Modifier

"Media:media" modifiers specify what media to use for an signaling action, such as "accept" and "call", as well as the parameters of the media. Their syntax is shown in Figure 11.

Node:	"Media:media"	
Outputs:	None	(Next node follows directly)
Next node:	Any node	
Parameters:	"media"	A list of medias. Possible values are "audio", and "video".
	"input"	The source of input. It can be "microphone", for audio, "camera" for video, or a URL to a media file.
	"mode"	Possible values are "sendonly", "recvonly", "sendrecv", and "inactive".
	"codec"	The codec used for the media.

Figure 11: Syntax of the "Media:media" node

10.2 Actions

10.2.1 "Media:mediaupdate" Action

"Media:mediaupdate" actions cause the user agent to change the media setup for an ongoing session. Their syntax is shown in Figure 12.

Node:	"mediaupdate"
Outputs:	None
Next node:	None
Parameters:	None

Figure 12: Syntax of the "mediaupdate" node

There is no parameters for the "mediaupdate" action. We can use "media" modifier to change "mediaupdate" action's media parameters, as described in Section 10.1.1. "Media:mediaupdate" actions can be used to handle some mid-call handling services, such as "mute" and "hold".

10.3 IANA Considerations

This extension registers a new URN per RFC 2141 [6], RFC 2648 [7], and RFC 3688 [8].

The XML namespace urn:ietf:params:xml:ns:less:media will only refer

to the version of LESS media extension in this document and will not change. Any other LESS enhancements MUST be made by extensions and MUST have different namespaces.

10.3.1 URN Sub-Namespace Registration for urn:ietf:params:xml:ns:less:media

URI: urn:ietf:params:xml:ns:less:media

Registrant Contact: Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
    "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
     content="text/html;charset=iso-8859-1"/>
  <title>Media Extension of the Language for End System Services
Namespace</title>
</head>
<body>
  <h1>Namespace for the Media Extension of the Language for End
System Services</h1>
  <h2>urn:ietf:params:xml:ns:less:media</h2>
  <p><a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
END
```

10.3.2 Schema registration

This specification registers XML Schema for LESS, as per the guidelines in [8].

URI: urn:ietf:params:xml:schema:less:media

Registrant contact:
Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

XML: The XML can be found in Section 19.2.

10.3.3 MIME Registration

LESS Media extension has the same MIME type as LESS as defined in Section 8.3.

11 Mid-call Handling extension

Mid-call handling actions can manipulate ongoing sessions. As we discussed before, "Media:mediaupdate" can handle "mute" and "hold" services. This extension defines two new actions, namely "Midcall:transfer" and "Midcall:merge".

11.1 Actions

11.1.1 "Midcall:transfer" Action

"Midcall:transfer" actions cause the user agent to transfer existing calls to another user. Their syntax is shown in Figure 13.

```
      Node:  "Midcall:transfer"
   Outputs:  "accepted"          Next node if "transfer" was accepted.
             "busy"              Next node if "transfer" returned "busy".
             "failure"           Next node if "transfer" failed.
             "noanswer"          Next node if "transfer" was not
                                 answered before timeout.
             "redirection"       Next node if "transfer" was redirected.
Parameters:  "timeout"           Time to try before giving up "transfer".

    Output:  "accepted"
Parameters:  none

    Output:  "busy"
Parameters:  none

    Output:  "noanswer"
Parameters:  none

    Output:  "failure"
Parameters:  none

    Output:  "redirection"
Parameters:  none
```

Figure 13: Syntax of the "Midcall:transfer" node

The "Midcall:transfer" action takes the same parameter and outputs as

"call" action, as defined in Section 5.4.5.

11.1.2 "Midcall:merge" Action

"Midcall:merge" actions will merge multiple calls into an end system hosted conference. By default, audio streams will be mixed and sent to all the call participants. Video streams are forwarded to all the call participants. If there are held or muted calls, the hold and mute status are kept. Their syntax is shown in Figure 14.

```
      Node:  "Midcall:merge"
   Outputs:  None
 Next node:  None
Parameters:  "uri"           Select calls to merge based on the "uri" list.
             "subject"       Select calls to merge based on
                             the "subject" list.
```

Figure 14: Syntax of the "Midcall:merge" node

There are two parameters for the "Midcall:merge" action, which define how to select calls to merge. If "uri" parameter presented, all calls to one of the "uri" values will be merged. If "subject" parameter presented, all calls with the one of the "subject" values will be merged. If further calls want to get merged into an existing end system hosted conference, users can use any value in "uri" list or "subject" list to refer to an existing conference.

11.2 IANA Considerations

This extension registers a new URN per RFC 2141 [6], RFC 2648 [7], and RFC 3688 [8].

The XML namespace urn:ietf:params:xml:ns:less:midcall will only refer to the version of LESS mid-call handling extension in this document and will not change. Any other LESS enhancements MUST be made by extensions and MUST have different namespaces.

11.2.1 URN Sub-Namespace Registration for urn:ietf:params:xml:ns:less:midcall

URI: urn:ietf:params:xml:ns:less:midcall

Registrant Contact: Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
    "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
     content="text/html;charset=iso-8859-1"/>
  <title>Mid-call Handling Extension of the Language for End
System Services Namespace</title>
</head>
<body>
  <h1>Namespace for the Mid-call Handling Extension of the
Language for End System Services</h1>
  <h2>urn:ietf:params:xml:ns:less:midcall</h2>
  <p><a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
END
```

11.2.2 Schema registration

This specification registers XML Schema for LESS, as per the guidelines in [8].

URI: urn:ietf:params:xml:schema:less:midcall

Registrant contact:
Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

XML: The XML can be found in Section 19.2.

11.2.3 MIME Registration

LESS Media extension has the same MIME type as LESS as defined in Section 8.3.

12 User Interaction extension

User Interaction extension handles user inputs and perform actions for alerting users and getting user inputs. It has one trigger and two actions defined.

12.1 Triggers

12.1.1 "UI:command" Trigger

"UI:command" handles user invoked commands, such as pressing call button, hold button, or transfer button. Their syntax is shown in Figure 15.

```
            Trigger:  "UI:command"
         Parameters:  "command"     Command name to match.
```

Figure 15: Syntax of the "UI:command" trigger

"UI:command" trigger takes one parameter, "command", to check the command to match. Commands can be "call", "hold", "mute", "transfer", or any other commands users may perform. Usually, a command is associated with a user interface button.

12.2 Actions

12.2.1 "UI:alert" Action

"UI:alert" actions play alerting messages. Their syntax is shown in Figure 16.

```
      Node:  "UI:alert"
   Outputs:  "timeout"   Next node if alerting time exceeds that
                         specified in "duration" parameter.
Parameters:  "duration"  Time to play alerting message before
                         stopping alert.
             "priority"  Priority of alerting. Possible values
                         are "emergency", "urgent", "normal",
                         and "non-urgent".
             "icon"      The icon to display on the alerting interface.
             "message"   The message to display on the alerting interface.
             "style"     Possible alerting styles. Can be a combination of
                         "vibrate", "sound", "flash",
                         and "text". By default, it is "sound".
             "input"     Play a file as specified in the "input" URI.

    Output:  "timeout"
Parameters:  none
```

Figure 16: Syntax of the "UI:alert" node

12.2.2 "UI:getinput" Action

"UI:getinput" actions acquire users' input. Their syntax is shown in Figure 17.

```
          Node:  "UI:getinput"
       Outputs:  "noanswer"                                  Next node if
no input after the time
                                                             specified in
"timeout" parameter.
    Parameters:  "timeout"                                   Time to wait
for user input.
                 "source"                                    Input source,
possible values are
                                                             "screen" and
"sound".
                 "target"                                    Get input
from "local" or "remote" user.
  Return value:  Return a text string acquired from user input.

        Output:  "noanswer"
    Parameters:  none
```

Figure 17: Syntax of the "UI:getinput" node

12.3 IANA Considerations

This extension registers a new URN per RFC 2141 [6], RFC 2648 [7], and RFC 3688 [8].

The XML namespace urn:ietf:params:xml:ns:less:ui will only refer to the version of LESS mid-call handling extension in this document and will not change. Any other LESS enhancements MUST be made by extensions and MUST have different namespaces.

12.3.1 URN Sub-Namespace Registration for urn:ietf:params:xml:ns:less:ui

URI: urn:ietf:params:xml:ns:less:ui

Registrant Contact: Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

XML:

```
              BEGIN
              <?xml version="1.0"?>
              <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
                  "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
              <html xmlns="http://www.w3.org/1999/xhtml">
              <head>
                <meta http-equiv="content-type"
                   content="text/html;charset=iso-8859-1"/>
                <title>User Interaction Extension of the Language for End
System Services Namespace</title>
              </head>
              <body>
                <h1>Namespace for the User Interaction Extension of the
Language for End System Services</h1>
                <h2>urn:ietf:params:xml:ns:less:ui</h2>
                <p><a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
              </body>
              </html>
              END
```

12.3.2 Schema registration

This specification registers XML Schema for LESS, as per the guidelines in [8].

URI: urn:ietf:params:xml:schema:less:ui

Registrant contact:
Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

XML: The XML can be found in Section 19.2.

12.3.3 MIME Registration

LESS Media extension has the same MIME type as LESS as defined in Section 8.3.

13 Instant Messaging extension

13.1 Triggers

13.1.1 "IM:message" Trigger

"IM:message" handles incoming instant messages. Their syntax is shown in Figure 18.

```
                    Trigger:  "IM:message"
                 Parameters:  None
```

Figure 18: Syntax of the "IM:message" trigger

13.2 Actions

13.2.1 "IM:sendmsg" Action

"IM:sendmsg" actions cause the user agent to send an outgoing instant message. Their syntax is shown in Figure 19.

```
      Node:  "IM:sendmsg"
   Outputs:  "accepted"     Next node if the message get accepted.
             "failure"      Next node if failed to send the message.
             "noanswer"     Next node if no response for the message
                            before timeout.
             "redirection"  Next node if the message was redirected.
Parameters:  "timeout"      Time to try before giving up sending message.

    Output:  "accepted"
Parameters:  none

    Output:  "noanswer"
Parameters:  none

    Output:  "failure"
Parameters:  none

    Output:  "redirection"
Parameters:  none
```

Figure 19: Syntax of the "IM:sendmsg" node

The parameters and outputs of "IM:sendmsg" actions are the same those for "call" actions, as defined in Section 5.4.5.

13.3 IANA Considerations

This extension registers a new URN per RFC 2141 [6], RFC 2648 [7],

and RFC 3688 [8].

The XML namespace urn:ietf:params:xml:ns:less:im will only refer to the version of LESS mid-call handling extension in this document and will not change. Any other LESS enhancements MUST be made by extensions and MUST have different namespaces.

13.3.1 URN Sub-Namespace Registration for urn:ietf:params:xml:ns:less:im

URI: urn:ietf:params:xml:ns:less:im

Registrant Contact: Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
    "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
     content="text/html;charset=iso-8859-1"/>
  <title>Instant Messaging Extension of the Language for End
System Services Namespace</title>
</head>
<body>
  <h1>Namespace for the Instant Messaging Extension of the
Language for End System Services</h1>
  <h2>urn:ietf:params:xml:ns:less:im</h2>
  <p><a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
END
```

13.3.2 Schema registration

This specification registers XML Schema for LESS, as per the guidelines in [8].

URI: urn:ietf:params:xml:schema:less:im

Registrant contact:
Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

XML: The XML can be found in Section 19.2.

13.3.3 MIME Registration

LESS Media extension has the same MIME type as LESS as defined in Section 8.3.

14 Event Handling extension

Event handling extension is used to handle event subscriptions and notifications. It has two triggers, one switch, and five actions defined.

14.1 Triggers

14.1.1 "Event:subscription" Trigger

"Event:subscription" handles incoming event subscription. Their syntax is shown in Figure 20.

```
Trigger:  "Event:subscription"
Parameters:  None
```

Figure 20: Syntax of the "Event:subscription" trigger

14.1.2 "Event:notification" Trigger

"Event:notification" handles incoming event notification. Their syntax is shown in Figure 21.

```
Trigger:  "Event:notification"
Parameters:  None
```

Figure 21: Syntax of the "Event:notification" trigger

14.2 Switches

14.2.1 "Event:event-switch" Switch

Event-switches allow a LESS script to make decisions based on the event values in an incoming event subscription or notification. The syntax of "event-switch" is summarized in Figure 22.

```
      Node:  "Event:event-switch"
   Outputs:  "event"              Specific event to match
Parameters:  None

    Output:  "event"
Parameters:  "package"            Match against a list of event packages.
             "is"                 Match against an event list.
```

Figure 22: Syntax of the "Event:event-switch" node

The "event" output takes two parameters, "package" checks whether an event has a specific package name, "is" checks whether an event is one of the event specified in the event list.

14.3 Actions

14.3.1 "Event:approve" Action

"Event:approve" actions cause the presence agent to approve an incoming event subscription. Their syntax is shown in Figure 23.

```
      Node:  "Event:approve"
   Outputs:  None
 Next node:  None
Parameters:  "expires"        Expiration duration for the subscription.
```

Figure 23: Syntax of the "Event:approve" node

14.3.2 "Event:deny" Action

"Event:deny" actions cause the presence agent to deny an incoming event subscription. Their syntax is shown in Figure 24.

```
                    Node:  "Event:deny"
                 Outputs:  None
               Next node:  None
              Parameters:  None
```

Figure 24: Syntax of the "Event:deny" node

There are no parameters and outputs for the "Event:deny" action.

14.3.3 "Event:defer" Action

"Event:defer" actions cause the presence agent to defer the decision on an incoming event subscription. This will cause the event subscription in pending state. Their syntax is shown in Figure 25.

```
                      Node:  "Event:defer"
                   Outputs:  None
                 Next node:  None
                Parameters:  None
```

Figure 25: Syntax of the "Event:defer" node

There are no parameters and outputs for the "Event:defer" action.

14.3.4 "Event:subscribe" Action

"Event:subscribe" actions cause the presence agent to send an event subscription to a presentity. Their syntax is shown in Figure 26.

```
         Node:  "Event:subscribe"
      Outputs:  "approved"         Next node if the subscription was approved.
                "denied"           Next node if the subscription was denied.
                "pending"          Next node if the subscription was pending.
                "noanswer"         Next node if the subscription was not
                                   answered before timeout.
                "failure"          Next node if the subscription failed for
                                   any other reasons.
                "redirection"      Next node if the subscription was
redirected.
   Parameters:  "timeout"          Time to try before giving up the
subscription.
                "expires"          The duration of the subscription.
                "package"          The event package to subscribe to.
                "content"          The content (e.g., event filtering) in
subscription.

       Output:  "approved"
   Parameters:  none

       Output:  "denied"
   Parameters:  none

       Output:  "pending"
   Parameters:  none

       Output:  "noanswer"
   Parameters:  none

       Output:  "failure"
   Parameters:  none

       Output:  "redirection"
   Parameters:  none
```

Figure 26: Syntax of the "Event:subscribe" node

14.3.5 "Event:notify" Action

"Event:notify" actions cause the presence agent to send an event subscription to a presentity. Their syntax is shown in Figure 27.

```
        Node:  "Event:notify"
     Outputs:  "accepted"      Next node if the notification was accepted.
               "noanswer"      Next node if the notification was not
                               confirmed before timeout.
               "failure"       Next node if the notification failed for
                               any other reasons.
               "redirection"   Next node if the notification was redirected.
  Parameters:  "timeout"       Time to try before giving up the notification.
               "package"       The event package in notification.
               "event"         The event in notification.
               "##any"         Any other parameters for an event.

      Output:  "accepted"
  Parameters:  none

      Output:  "noanswer"
  Parameters:  none

      Output:  "failure"
  Parameters:  none

      Output:  "redirection"
  Parameters:  none
```

Figure 27: Syntax of the "Event:notify" node

14.4 IANA Considerations

This extension registers a new URN per RFC 2141 [6], RFC 2648 [7], and RFC 3688 [8].

The XML namespace urn:ietf:params:xml:ns:less:event will only refer to the version of LESS mid-call handling extension in this document and will not change. Any other LESS enhancements MUST be made by extensions and MUST have different namespaces.

14.4.1 URN Sub-Namespace Registration for urn:ietf:params:xml:ns:less:event

URI: urn:ietf:params:xml:ns:less:event

Registrant Contact: Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
    "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
     content="text/html;charset=iso-8859-1"/>
  <title>Event Handling Extension of the Language for End System
Services Namespace</title>
</head>
<body>
  <h1>Namespace for the Event Handling Extension of the Language
for End System Services</h1>
  <h2>urn:ietf:params:xml:ns:less:event</h2>
  <p><a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
END
```

14.4.2 Schema registration

This specification registers XML Schema for LESS, as per the guidelines in [8].

URI: urn:ietf:params:xml:schema:less:event

Registrant contact:
Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

XML: The XML can be found in Section 19.2.

14.4.3 MIME Registration

LESS Media extension has the same MIME type as LESS as defined in Section 8.3.

15 Location-based service extension

This extension introduces two new switch nodes named where-switch and where-relation-switch. The new switch nodes check people's physical locations and based on the location information to make call control decisions.

15.1 Switches

15.1.1 "Location:where-switch" Switch

"Where-switches" allow a LESS script to make call decisions based on the physical location information. They are summarized in Figure 28.

Node:	"Location:where-switch"	
Outputs:	"where"	Specific the location to match
Parameters:	"type"	"geospatial", "civil"
	"uri"	the location occupier's uri. If omited, check the script owner's location.
Output:	"where"	
Parameters:	"longitude"	Longitude coordinates, valid only for geospatial type
	"latitude"	Latitude coordinates, valid only for geospatial type
	"altitude"	Altitude coordinates, valid only for geospatial type
	"country"	Country of the location, valid only for civil type
	"A1"	National subdivisions (state, region, province, prefecture), valid only for civil type
	"A2"	County, parish, gun (JP), district (IN), valid only for civil type
	"A3"	City, township, shi (JP), valid only for civil type
	"A4"	City division, borough, city district, ward, chou (JP), valid only for civil type
	"A5"	neighborhood, block, valid only for civil type
	"A6"	street, valid only for civil type
	"PRD"	Leading street direction, valid only for civil type
	"POD"	Trailing street suffix, valid only for civil type
	"STS"	Street suffix, valid only for civil type
	"HNO"	House number, numeric part only, valid only for civil type
	"HNS"	House number suffix, valid only for civil type
	"LMK"	Landmark or vanity address, valid only for civil type
	"LOC"	Additional location information, such as room valid only for civil type
	"FLR"	Floor, valid only for civil type
	"NAM"	Name (residence, business or office occupant) valid only for civil type
	"PC"	Postal code, valid only for civil type
	"distance"	Distance value to a specified location.

Valid only if geospatial or
civil location presented.

	"condition"	Valid only if distance presented. Value is (in|out|at|). Default value is "at".
	"unit"	Valid only if distance presented. Value is (m|km|mi|in|ft|yd|naut mi). Default value is "m".
	"direction"	Direction relative to a specified location, Valid only if geospatial or civil location presented. Value is "(south|north|west|east|above|below)".

Figure 28: Syntax of the "Location:where-switch" node

Location:where-switches have two node parameters: The mandatory parameter "type" specifies the location type to match: either geospatial location, or civil location. The optional parameter "uri" specifies the location occupier's uri. If not provided, the location occupier will be the script owner.

The "where" output tag may have one or more parameters, indicating the exact location to match.

The "longitude", "latitude", and "altitude" are used to specify a geographical location. The country, "A1", "A2", "A3", "A4", "A5", "A6", "PRD", "POD", "STS", "HNO", "HNS", "LMK", "LOC", "FLR", "NAM", and "PC" are used to specify a civil location.

The "distance" parameter cannot be used without geographical location or civil location specified. Usually, the distance parameter is used with geographical location since it is difficult to measure the distance to a civil location. The "condition" and "unit" parameters are used to provide more convenient ways for distance specification. The available unit values are (m|km|mi|in|ft|yd|naut mi), mapped to meter, kilometer, mile, inch, foot, yard, and nautical mile, accordingly.

The "direction" parameter cannot be used without geographical location or civil location specified. The value "south" or "north" are usually used when latitude specified. The value "east" or "west" are usually used when longitude specified. The value "above" and "below" are usually used when altitude specified, or FLR specified.

15.1.2 "Location:where-relation-switch"

Location:where-relation-switches allow a LESS script to make call decisions based on the location relation of two persons. They are

summarized in Figure 29.

Node:	"Location:where-relation-switch"	
Outputs:	"where-relation"	Specific location relation to match
Parameters:	"uri1"	the first location occupier's uri
	"uri2"	the second location occupier's uri
Output:	"where-relation"	
Parameters:	"distance"	Distance value between two persons.
	"condition"	Valid only if distance presented. Value is (in|out|at|). Default value is "at".
	"unit"	Valid only if distance presented. Value is (m|km|mi|in|ft|yd |naut mi). Default value is "m".
	"direction"	Direction relation between two persons, value is "(south| north| west|east|above|below)".
	"same"	A space separated list containing location attributes, e.g., LOC, that two persons have the same value.
	"difference"	A space separated list containing location attributes, that two persons have different values.

Figure 29: Syntax of the "Location:where-relation-switch" node

Location:where-relation-switches have two node parameters: The mandatory parameter "uri1" specifies the first location occupier's uri. The optional parameter "uri2" specifies the second location occupier's uri. If not provided, the second location occupier will be the script owner.

The "where-relation" output tag may have one or more parameters, indicating the exact relation to match.

The "distance" parameter indicates the distance between two persons. The "condition" and "unit" parameters are used for distance measuring. They have the same meanings as those defined in "where-switch".

The "direction" parameter indicates relative positions between two persons. For example, "south" means uri1 is south of uri2.

The "same" and "difference" indicating the relationship between two persons' location attributes. The attributes can be geospatial location attributes, such as "longitude", "latitude", and "altitude". They can also be civil location attributes, such as country, "A1", "A2", "A3", "A4", "A5", "A6", "PRD", "POD", "STS", "HNO", "HNS", "LMK", "LOC", "FLR", "NAM", and "PC".

15.2 IANA Considerations

This extension registers a new URN per RFC 2141 [6], RFC 2648 [7], and RFC 3688 [8].

The XML namespace urn:ietf:params:xml:ns:less:location will only refer to the version of LESS mid-call handling extension in this document and will not change. Any other LESS enhancements MUST be made by extensions and MUST have different namespaces.

15.2.1 URN Sub-Namespace Registration for urn:ietf:params:xml:ns:less:location

URI: urn:ietf:params:xml:ns:less:location

Registrant Contact: Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
    "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
     content="text/html;charset=iso-8859-1"/>
  <title>Location-based Service Extension of the Language for End
System Services Namespace</title>
</head>
<body>
  <h1>Namespace for the Location-based Service Extension of the
Language for End System Services</h1>
  <h2>urn:ietf:params:xml:ns:less:location</h2>
  <p><a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
END
```

15.2.2 Schema registration

This specification registers XML Schema for LESS, as per the guidelines in [8].

URI: urn:ietf:params:xml:schema:less:location

Registrant contact:
Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

XML: The XML can be found in Section 19.2.

15.2.3 MIME Registration

LESS Media extension has the same MIME type as LESS as defined in Section 8.3.

16 Queue handling extension

Many call center services require queueing incoming calls for available operators. This extension defines two actions for queue operation.

16.1 Actions

16.1.1 "Queue:enqueue" Action

"Queue:enqueue" actions cause the user agent to put an incoming call in a waiting queue. Their syntax is shown in Figure 30.

Node:	"Queue:enqueue"	
Outputs:	"full"	If the number of calls exceeds the "maxlen".
Parameters:	"queue"	The name of the queue. Possible values can be any string. Two pre-defined names are "hold" and "callback", which hosts holding calls and calls waiting for callback, accordingly. If not provided, use user agent's default queue.
	"maxlen"	The maximum calls can be held in the queue.
Output:	"full"	
Parameters:	none	

Figure 30: Syntax of the "Queue:enqueue" node

16.1.2 "Queue:dequeue" Action

"Queue:dequeue" actions cause the user agent to retrieve a call from a waiting queue. Their syntax is shown in Figure 31.

```
        Node:  "Queue:dequeue"
     Outputs:  "success"        A call get retrieved.
               "failure"        Queue empty or other reasons. "dequeue"
failed.
  Parameters:  "queue"          The name of the queue. Possible values can be
                                any string. Two pre-defined names are "hold"
                                and "callback", which hosts holding calls and
                                calls waiting for callback, accordingly.
                                If not provided, use user agent's default
queue.

      Output:  "success"
  Parameters:  none

      Output:  "failure"
  Parameters:  none
```

Figure 31: Syntax of the "Queue:dequeue" node

17 Examples

The following sections investigate services suitable for end systems. We looked into ITU Q.1211 [16] services, AT&T 5ESS switch services [17], services defined in CSTA Phase III [18], and new services incorporated with other Internet services, such as email and presence information, for appropriate end system services.

17.1 Q.1211 services

The technical report by Lennox et al [19] illustrated how to use SIP to handle Q.1211 services. In table 1 of that report, it defines what Q.1211 services are suitable for end systems. We only focuses on the services appropriate for end systems and tries to use LESS to program those services.

17.1.1 Abbreviated dialing (ABD)

"Abbreviated dialing allows the definition of short (e.g., two digit) digit sequences to represent the actual dialing digit sequence for a public or private numbering scheme."

Internet telephony end systems may simply use speed-dial buttons to handle the work. Instead of dialing a digit sequence, a single

button-click may trigger a call. End systems may also use LESS service scripts to do the translation from a short sequence to an actual sequence. Below is a LESS script example for the ABD service.

```
<less>
  <UI:command command="call">
    <address-switch field="destination">
      <address is="tel:11; phone-context=local">
        <location url="tel:+1-212-939-7054">
          <call/>
        </location>
      </address>
    </address-switch>
  </UI:command>
</less>
```

17.1.2 Attendant (ATT)

"This allows VPN users to access an attendant (operator) position within the VPN for providing VPN service information (e.g., VPN numbers) by dialing a special access code. An Internet telephony end system needs only to be configured with an address of an appropriate local operator to translate the special access code to the actual local address of an attendant, or some address which will resolve to that address." For LESS, this is similar to the ABD service (Section 17.1.1) handling.

17.1.3 Authentication (AUTC)

"This allows verification that a user is allowed to access certain options in the telephone network. This should be in basic end system implementation."

```
<less>
  <UI:command command="call">
    <authenticate user="{message.origin}" target="local">
      <success>
        <!-- If authentication succeeded, accept the call -->
        <call/>
      </success>
      <failure>
        <!-- If authentication failed, terminate the call attempt-->
        <terminate/>
      </failure>
```

```
      </authenticate>
    </UI:command>
  </less>
```

17.1.4 Authorization code (AUTZ)

"This allows a user (typically in a VPN) to override the restrictions placed on the system from which calls are made. This should be in basic end system implementation." Though we can user LESS scripts to automate the authorization process, it is unsafe to save the codes in LESS scripts.

17.1.5 Automatic call back (ACB)

"This feature allows the called party to automatically call back the calling party of the last call directed to the called party."

This can be handled by either end systems or network servers. If handled by a network server, the network server should send a SIP REFER request to the calling party to initiate the call back (This is more like traditional Automatic Callback service).

The service involves two steps, one is to log received invitations, the other is to retrieve the invitations and make calls. Below is a LESS service script handling this service.

```
<less>
  <incoming>
    <status-switch status-name="activity">
      <status is="busy">
        <reject>
          <!-- Any action should have a 'next' for subsequent actions -->
          <!-- If one action has a sibling action, the sibling action
               will be executed in parallel with this action -->
          <next>
            <!-- A "callback" queue is defined -->
            <Queue:enqueue queue="callback"/>
          </next>
        </reject>
      </status>
    </status-switch>
  </incoming>
  <Event:notification>
    <address-switch field="origin">
      <address url="{agent.uri}">
```

```
          <Event:event-switch>
            <Event:event package="presence" status="open" activity="">
              <!-- If no 'number' parameter provided, the Queue:dequeue action
                   will sequentially retrieve all elements in the queue -->
              <Queue:dequeue queue="callback">
                <success>
                  <call/>
                </success>
              </Queue:dequeue>
            </Event:event>
          </Event:event-switch>
        </address>
      </address-switch>
    </Event:notification>
  </less>
```

17.1.6 Call distribution (CD)

"This service feature allows the served user to specify the percentage of calls to be distributed among two or more destinations. Other criteria may also apply to the distribution of calls to each destination."

This is usually done by network servers. If we extend CPL with a switch checking multiple destinations' status, we can handle the service in a proxy server by using CPL.

17.1.7 Call forwarding (CF)

"This service feature allows a user to have his incoming calls addressed to another number, no matter what the called party line status may be." A simple LESS script can handle this below:

```
<less>
  <incoming>
    <location url="sip:bob-home@example.com">
      <redirect/>
    </location>
  </incoming>
</less>
```

17.1.8 Call forwarding on busy/don't answer (CFC)

"This service feature allows the called user to forward particular calls if the called user is busy or does not answer within a specified number of rings."

This service requires an end system to check its status and make call decisions. The following script forwards calls on busy.

```
<less>
  <incoming>
    <status-switch status-name="activity">
      <status is="on-the-phone">
        <location url="sip:bob-home@example.com">
          <redirect/>
        </location>
      </status>
    </status-switch>
  </incoming>
</less>
```

The following script is for call forwarding on no answer.

```
<less>
  <incoming>
    <alert timeout="2000">
      <timeout>
        <location url="sip:bob-home@example.com">
          <redirect/>
        </location>
      </timeout>
    </alert>
  </incoming>
</less>
```

17.1.9 Call gapping (GAP)

"This feature allows the service provider to restrict the number of calls to a served user to prevent congestion of the network." This service should be implemented in network servers. A similar feature, Call Limiter (LIM) (Section 17.1.11), will do the similar thing but for end systems.

17.1.10 Call hold with announcement (CHA)

"The call hold with announcement service feature allows a subscriber to place a call on hold with options to play music or customized announcements to the held party." This service requires user interaction. A LESS script that can enhance the service by customizing the announcements based on addresses is shown below:

```
<less>
  <!-- A trigger to handle user input -->
  <!-- For portability, the values for the parameter 'command'
       are pre-defined, each command can only be applied under a
       certain condition. 'hold' can only be used with an
       ongoing call -->
  <UI:command command="hold">
    <address-switch field="destination">
      <address is="sip:bob@example.com">
        <Media:media media="audio" input="music.au" mode="sendonly">
          <Media:mediaupdate/>
        </Media:media>
      </address>
    </address-switch>
  </UI:command>
</less>
```

17.1.11 Call limiter (LIM)

"This service feature allows a served user to specify the maximum number of simultaneous calls to a served user's destination. If the destination is busy, the call may be routed to an alternative destination."

Internet telephony end systems can have virtually unlimited lines. However, the resources (CPU, memory, I/O devices) are limited for an end device so the number of simultaneous calls is limited. This can be handled by a LESS script below:

```
<less>
  <incoming>
    <status-switch status-name="active-calls">
      <status equal="2">
        <reject status="busy"/>
      </status>
    </status-switch>
  </incoming>
</less>
```

17.1.12 Call logging (LOG)

"This service feature allows for a record to be prepared each time that a call is received to a specified telephone number." The <log> action in LESS can handle this feature.

```
<less>
  <incoming>
    <log/>
  </incoming>
</less>
```

17.1.13 Call queueing (QUE)

"This service feature allows calls which would otherwise be declared busy to be placed in a queue and connected as soon as the free condition is detected. Upon entering the queue, the caller hears an initial announcement informing the caller the call will be answered when a line is available."

The call queueing service can be handled by an application server in the network. End systems can also handle it. The LESS script below handles the service.

```
<less>
  <incoming>
    <status-switch status-name="active-calls">
      <!-- For more than one active call, we use 'active-calls="1+" -->
      <status equal="1">
        <Media:media media="audio" input="music.au" mode="sendonly">
          <accept>
            <next>
              <Queue:enqueue queue="hold"/>
            </next>
          </accept>
        </Media:media>
      </status>
    </status-switch>
  </incoming>
  <Event:notification>
    <status-switch status-name="active-calls">
      <status equal="0">
        <!-- Retrieve only one call from the hold queue -->
        <Queue:dequeue queue="hold" number="1">
```

```
            <success>
              <Media:media media="audio" input="microphone" mode="sendrecv">
                <Media:mediaupdate/>
              </Media:media>
            </success>
          </Queue:dequeue>
        </status>
      </status-switch>
    </Event:notification>
  </less>
```

17.1.14 Call transfer (TRA)

"The call transfer service feature allows a subscriber to place a call on hold and transfer the call to another location."

In LESS, the <transfer> action is used to handle the service. Call transfer is a user triggered service. A LESS script can be triggered by a user input event to perform a transfer.

```
<less>
  <!-- the 'transfer' command only applies to the current ongoing call -->
  <UI:command command="transfer">
    <!-- Transfer all calls between 09:00 ~ 17:00 to specialist@foo.com,
         otherwise, to technician@foo.com -->
    <time-switch>
      <time dtstart="20040716T090000Z"
        duration="PT8H" freq="weekly" byday="MO,TU,WE,TH,FR">
        <location uri="specialist@foo.com">
          <transfer/>
        </location>
      </time>
      <otherwise>
        <location uri="technician@foo.com">
          <transfer/>
        </location>
      </otherwise>
    </time-switch>
  </UI:command>
</less>
```

17.1.15 Call waiting (CW)

"This service feature allows a subscriber to receive a notification that another party is trying to reach his number while he is busy talking to another calling party."

Since an Internet telephony end system can have multiple lines and a better user interface, such as a bigger LCD display, it can alert users for any new calls. No service scripts required for this service.

17.1.16 Closed user group (CUG)

"This service feature allows the user to be a member of a set of users who are normally authorized to make and receive calls only within the group."

This service should be put on network servers, though end systems can handle it by using call screening services. We will introduce call screening services in Section 17.1.30 and 17.1.37.

17.1.17 Consultation calling (COC)

"The consultation calling service feature allows a subscriber to place a call on hold, in order to initiate a new call for consultation."

For Internet telephony end systems, this service is similar to Call hold with announcements (CHA), which we have discussed in Section 17.1.10.

17.1.18 Customer profile management (CPM)

"This service feature allows the subscriber to real-time manage his service profile, i.e. terminating destinations, announcements to be played, call distribution, and so on."

This is about service script management. There should be a friendly service creation and management user interface for end systems.

17.1.19 Customer recorded announcement (CRA)

"This service allows a call to be completed to a (customized) terminating announcement instead of a subscriber line. The served user may define different announcements for unsuccessful call completions due to different reasons (e.g. caller outside business hours, all lines are busy)." The switches in LESS can do the customization work. For example, a time-based customization is as below:

```
   <less>
     <incoming>
       <!-- Effective from 07/16/2004, if an incoming call is after
            05:00:00 PM Friday, and before 09:00:00 AM Monday, play
            the no_office_hour.au announcement -->
       <time-switch>
         <time dtstart="20040716T170000Z"
           duration="PT16H" freq="weekly" byday="MO,TU,WE,TH,FR">
           <Media:media media="audio" source="no_office_hour.au"
mode="sendonly">
             <accept/>
           </Media:media>
         </time>
       </time-switch>
     </incoming>
   </less>
```

17.1.20 Customized ringing (CRG)

"This service feature allows the subscriber to allocate a distinctive ringing to a list of calling parties."

This service can only be handled in an end system. The script below shows an enhanced version not only based on address but also based on priority of the call to perform customized ringing.

```
   <less>
     <incoming>
       <priority-switch>
         <priority equal="emergency">
           <alert priority="emergency"/>
         </priority>
         <otherwise>
           <address-switch field="origin">
             <address is="sip:bob@example.com">
               <alert audio="bob.au"/>
             </address>
             <otherwise>
               <alert/>
             </otherwise>
           </address-switch>
         </otherwise>
       </priority-switch>
     </incoming>
   </less>
```

17.1.21 Destination user prompter (DUP)

"This service feature enables to prompt the called party with a specific announcement. Such an announcement may ask the called party enter an extra numbering, e.g. through DTMF, or a voice instruction that can be used by the service logic to continue to process the call."

Though we can use LESS <Media:media> modifier to easily play an announcement, it is out of LESS's scope to handle DTMF inputs. The DTMF handling falls into VoiceXML's [20] domain. For example, UI:prompt script="input1.vxml"/>. More investigation is required on integrating VoiceXML into LESS.

17.1.22 Follow-me diversion (FMD)

"With this service feature, a user may register for incoming calls to any terminal access." This service should be in network servers.

17.1.23 Mass calling (MAS)

"This service feature allows processing of huge numbers of incoming calls, generated by broadcasted advertisings or games." This service should be handled by network servers.

17.1.24 Meet-me conference (MMC)

"This service feature allows the user to reserve a conference resource for making a multi-party call. At a specified date and time, each participant in the conference has to dial a designated number in order to have access to the conference." This is a standard conferencing service. We can use LESS to perform time and caller checking, and authentication.

```
<less>
  <incoming>
    <time-switch>
      <!-- a conference on every Monday from 17:00 to 19:00 -->
      <time dtstart="20040716T170000Z"
        duration="PT2H" freq="weekly" byday="MO">
        <address-switch field="origin">
          <address is="sip:bob@example.com">
            <authenticate user="{message.destination}" target="remote">
              <success>
                <accept/>
```

```
              </success>
            </authenticate>
          </address>
        </address-switch>
      </time>
    </incoming>
  </less>
```

17.1.25 Multi-way calling (MWC)

"This service feature allows the user to establish multiple, simultaneous telephone calls with other parties."

An Internet telephony end system can initiate as many simultaneous calls as it wishes as long as the network bandwidth and the CPU processing speed permit. A changed version of the service will be to put a limit on the number of simultaneous calls. The changed version can be handled by a LESS script below:

```
<less>
  <UI:command command="call">
    <status-switch status-name="active-calls">
      <status equal="2">
        <terminate>
          <next>
            <alert message="Only 2 active calls allowed"/>
          </next>
        </terminate>
      </status>
    </status-switch>
  </UI:command>
</less>
```

17.1.26 Off-net access (OFA)

"This service feature allows a VPN user to access his or her VPN from any non-VPN station in the PSTN by using a personal identification number (PIN)." This service should not be handled by using LESS scripts.

17.1.27 Off-net calling (ONC)

"This service feature allows the user to call outside the VPN

network."

This service is just a standard firewall traversal process, no service scripts required.

17.1.28 One number (ONE)

"This feature allows a subscriber with two or more terminating lines in any number of locations to have a single telephone number. This allows businesses to advertise just one telephone number throughout their market area and to maintain their operations in different locations to maximize efficiency. The subscriber can specify which calls are to be terminated on which terminating lines based on the area the calls originate."

This is a standard SIP proxy service, not handled by end systems.

17.1.29 Origin dependent routing (ODR)

"This service feature enables the subscriber to accept or reject a call, and in case of acceptance, to route this call, according to the calling party geographical location. This service feature allows the served user to specify the destination installations according to the geographical area from which the call was originated."

This service requires <Location:where-switch> to handle location-based call routing. A service script below can handle the service. For end systems, we use <redirect/> action to route calls. A proxy server is a more appropriate place to host the service, and use <proxy/> action to route calls.

```
<less>
  <incoming>
    <address-switch field="origin">
      <address is="sip:bob@example.com">
        <Location:where-switch type="civil" principle="sip:bob@example.com">
          <Location:where country="USA" A1="New York"
            A3="New York" A6="West 120th" HNO="450" LOC="Room 563">
            <location url="sip:bob-office@example.com">
              <redirect/>
            </location>
          </Location:where>
          <otherwise>
            <location url="sip:bob-mobile@example.com">
              <redirect/>
            </location>
          </otherwise>
```

```
          </Location:where-switch>
        </address>
      </address-switch>
    </incoming>
  </less>
```

17.1.30 Originating call screening (OCS)

"This service feature allows the served user to bar calls from certain areas based on the District code of the area from which the call is originated." This service can be handled by using address-switch.

```
<less>
  <incoming>
    <address-switch field="destination" subfield="host">
      <address is="example.com">
        <reject status="reject"/>
      </address>
    </address-switch>
  </incoming>
</less>
```

17.1.31 Originating user prompter (OUP)

"This service feature allows a served user to provide an announcement which will request the caller to enter a digit or series of digits via a DTMF phone or generator. The collected digits will provide additional information that can be used for direct routing or as a security check during call processing."

This service can be in a network server or a user agent. To enable a user agent to support this service, it requires the user agent to handle the interaction with remote users.

17.1.32 Personal numbering (PN)

"This service feature supports a UPT number that uniquely identifies each UPT user and is used by the caller to reach that UPT user." Reaching a user through a personal address is accomplished simply by a proxy or redirection server which locates the user.

17.1.33 Premium charging (PRMC)

"This service feature allows for the pay back of part of the cost of a call to the called party." This service is out of the scope of LESS.

17.1.34 Private numbering plan (PNP)

"This service feature allows the subscriber to maintain a numbering plan within his private network, which is separate from the public numbering plan." This service should usually be put in outbound proxy servers. If we provide such service in end systems, it would be similar to the Abbreviated dialing (ABD) service we introduced in Section 17.1.1.

17.1.35 Reverse charging (REVC)

"This service feature allows the service subscriber (e.g. freephone) to accept to receive calls at its expense and be charged for the entire cost of the call."

This service and the next service "Split charging" require a detailed VoIP charging schema and have the charging information available to users. Currently, LESS cannot handle this kind of service. However, if call signaling messages can provide charging information, LESS scripts can perform call decisions, such as automatically rejecting a reverse charging calls unless it is from certain users, based on the charging information.

17.1.36 Split charging (SPLC)

"This service feature allows for the separation of charges for a specific call, the calling and called party each being charged for one part of the call." This service is similar to the previous service, LESS cannot handle the service.

17.1.37 Terminating call screening (TCS)

"This service feature allows the user to screen calls based on the terminating telephone number dialed." Below is a LESS script example handling this service.

```
<less>
  <incoming>
    <address-switch field="destination" subfield="host">
      <address is="example.com">
        <reject status="reject"/>
      </address>
    </address-switch>
```

```
  </incoming>
</less>
```

17.1.38 Time dependent routing (TDR)

"This services feature allows the served user to apply different call treatments based on the time of day, day of week, day of year, holiday, etc." Usually, routing services should be put in network servers, but we can perform similar services for time dependent redirecting. This can be handled by LESS time-switch.

```
<less>
  <incoming>
    <time-switch>
      <time dtstart="20040716T170000Z"
        duration="PT8H" freq="weekly" byday="MO">
        <location url="sip:bob@voicemail.com>
          <redirect/>
        </location>
      </time>
    </time-switch>
  </incoming>
</less>
```

17.2 5ESS services

In this section, we investigate the services introduced in "AT&T 5ESS Switch The Premier Solution, Feature Handbook" [17]. 5ESS services including the following categories: BRCS (Business and Residence Custom Services) features, ISDN features (message and MBKS), Public Service features (emergency), Defense Switched Network Features, Toll and Tandem features, Interoffice Signaling and Control features, Operator Service Position System features, OA&M features. Among these categories, we are most interested in the BRCS features because many of BRCS features can be handled in end systems. For ISDN features, we will discuss the Message Services and the MBKS (Multibutton Key Telephone System). For Public Service features, we will discuss the group alerting service. We will not discuss the other service categories. We only list the services suitable for end systems.

17.2.1 Attendant call transfer (also known as Call splitting)

"The service allows the attendant to flash and dial a code before

dialing the third leg of a 3-way call. This inhibits the automatic connecting of all the parties to allow private consultation between the attendant and the third leg of the call. If the code is not dialed, the automatic 3-way is formed."

Since an Internet telephony end system usually can have multiple lines, it is easy to put one line on hold and initiate another call. However, to enable this service, an end system should be able to merge an active call and a held call into a 3-way call.

This service is not a programmable service, a user will manually use a second line to initiate the second call, and manually instruct his user agent to merge two calls.

17.2.2 Attendant camp-on

"This service allows incoming calls that the attendant attempts to complete to a busy station to be held waiting until the busy station becomes idle. The busy station receives a tone (indication of camp-on) each time the attendant attempts a completion. The call being transferred receives audible ringing, special tone followed by audible ringing, or optional silence, while waiting for the busy station to answer the call." The following script can handle the service.

```
<less>
  <incoming>
    <status-switch status-name="active-calls">
      <!-- We use 1+ to represent 1 or more active-calls -->
      <status greater="1">
        <Media:media source="wait.au" mode="sendonly">
          <accept>
            <next>
              <Queue:enqueue queue="hold"/>
            </next>
          </accept>
        </Media:media>
      </status>
    </status-switch>
  </incoming>
  <Event:notification>
    <status-switch status-name="active-calls">
      <status is="0">
        <Queue:dequeue queue="hold" number="1">
          <success>
            <Media:media media="audio" input="microphone" mode="sendrecv">
              <Media:mediaupdate/>
```

```
            </Media:media>
          </success>
        </Queue:dequeue>
      </status>
    </status-switch>
  </Event:notification>
</less>
```

17.2.3 Attendant conference

"This service enables an attendant to initiate a conference call involving up to six parties (including attendant). The selection of a special conference attendant can be done from any station within the same customer group by dialing a particular access code."

There are two ways to handle conferencing services in an end system, one is to have all conference participants to connect to a conference server (by REFER or third-party call control), the other is to use end system mixing. The first way is more scalable, but requires an external conference server. The second way requires an end system supporting mixing. The following scripts show both ways for a time-based conference initiation.

```
<less>
  <!-- Use SIP REFER -->
  <timer dtstart="20040716T170000Z"
    duration="PT2H" freq="weekly" byday="MO">
    <lookup source="sip:bob@example.com
            sip:tom@example.com sip:alice@abc.com">
      <success>
        <call>
          <success>
            <location url="sip:conf@example.com">
              <transfer/>
            </location>
          </success>
        </call>
      </success>
    </lookup>
  </timer>
</less>
```

The script will automatically send a call invitation to each URL in

the list, once a call established, the script will transfer the call to the conference server at sip:conf@example.com.

```
<less>
  <!-- Use local mixing -->
  <timer dtstart="20040716T170000Z"
    duration="PT2H" freq="weekly" byday="MO">
    <sub ref="IRTdef"/>
    <lookup source="sip:bob@ex.com sip:tom@ex.com">
      <success>
        <call subject="Group meeting">
          <success>
            <merge subject="Group meeting"/>
          </success>
        </call>
      </success>
    </lookup>
  </timer>
</less>
```

17.2.4 Authorization code

"The codes allow the station user to input an assigned code to change the restrictions associated with the originating station to those associated with the assigned authorization code. Thus, unauthorized use of facilities is avoid."

```
<less>
  <UI:command command="call">
    <address-switch field="destination">
      <address is="sip:conf123@example.com">
        <call>
          <success>
            <IM:sendmsg type="dtmf" digits="123456"/>
          </success>
        </call>
      </address>
    </address-switch>
  </UI:command>
</less>
```

17.2.5 Automatic recall

"This service lets the customer automatically call the LOCDN (last outgoing call directory number) currently associated with the customer's station when both stations become idle. It is different from Automaticall Callback, which automatically places a call on LICDN (last incoming call directory number)." The service script below shows how to program the service in LESS.

```
<less>
  <UI:command command="call">
    <call>
      <failed status="486,603">
        <Queue:enqueue queue="callback">
          <next>
            <Event:subscribe/>
          </next>
        </Queue:enqueue>
      </failed>
    </call>
  </UI:command>
  <Event:notification>
    <Event:event-switch>
      <Event:event activity="normal">
        <!-- User {message.origin} to represent the sender of the
notification-->
        <Queue:dequeue queue="callback" match="{message.origin}">
          <success>
            <call/>
          </success>
        </Queue:dequeue>
      </Event:event>
    </Event:event-switch>
  </Event:notification>
</less>
```

17.2.6 Call forwarding busy line (CFBL)

"This service permits calls attempting to terminate to a busy line to be redirected to another customer-specified line." The following script can handle the service.

```
<less>
  <incoming>
    <status-switch status-name="active-calls">
      <status greater="1">
        <location url="sip:phone2@example.com">
```

```
            <redirect/>
          </location>
        </status>
      </status-switch>
    </incoming>
  </less>
```

17.2.7 Call forwarding busy line--incoming only

"This service provides that only incoming DID calls are forwarded to the specified business group line on busy. Intragroup call attempts and attempts from private facilities to terminate to the busy line receive busy treatment." This is in fact address-based call forwarding services.

```
<less>
  <incoming>
    <status-switch status-name="active-calls">
      <status greater="1">
        <address-switch field="origin">
          <address isnot="sip:*@example.com">
            <location url="sip:phone2@example.com">
              <redirect/>
            </location>
          </address>
        </address-switch>
      </status>
    </status-switch>
  </incoming>
</less>
```

17.2.8 Call forwarding--don't answer

"This service forwards incoming calls to a station when the called station is not answered after a customer-specified number of ringing cycles." The script below can handle the service.

```
<less>
  <incoming>
    <alert timeout="2000">
      <timeout>
        <location url="sip:phone2@example.com">
```

```
            <redirect/>
          </location>
        </timeout>
      </alert>
    </incoming>
  </less>
```

17.2.9 Unconditional call forwarding

"This service forwards all incoming calls to another telephone." We can simply handle the service by using <redirect/> action.

```
<less>
  <incoming>
    <location url="sip:bob@room123.example.com">
      <redirect/>
    </location>
  </incoming>
</less>
```

17.2.10 Call hold

"This service allows a station user to 'hold' a call in progress by flashing and then dialing the call hold code. This frees the line for originating another call, answering a waiting call, or returning to a held call." In LESS, we use the action <Media:mediaupdate/> to change the media source for "hold" and "unhold" service.

```
  ...
  <!-- hold -->
  <Media:media media="audio" input="music.au" mode="sendonly">
    <Media:mediaupdate/>
  </Media:media>
  ...
  <!-- unhold -->
  <Media:media media="audio" input="microphone" mode="sendrecv">
    <Media:mediaupdate/>
  </Media:media>
```

17.2.11 Call transfer--individual--all calls

"This service allows a station user to transfer any established call to another station within or outside the PBX or business group without the assistance of the attendant. This is accomplished by flashing while on a stable 2-party call, dialing the desired party, and hanging up the telephone." The <transfer/> action can handle this service.

```
<less>
  <incoming>
    <media media="audio" input="transfer.au" mode="sendonly">
      <accept>
        <next>
          <location url="sip:bob@room123.example.com">
            <transfer>
              <failed>
                <media media="audio" input="failed.au" mode="sendonly">
                  <mediaupdate/>
                </media>
              </failed>
            </transfer>
          </location>
        </next>
      </accept>
    </media>
  </incoming>
</less>
```

17.2.12 Call waiting and cancel call waiting

Since an Internet telephony end system usually have multiple lines, users do not have to use flash button to switch calls. However, users should be able to control how many active calls an end device can handle. The following script can help to configure the maximum number of calls.

```
<less>
  <incoming>
    <status-switch status-name="active-calls">
      <status greater="3">
        <reject reason="busy"/>
      </status>
    </status-switch>
  </incoming>
</less>
```

17.2.13 Circle hunting

"This service allows all lines in a multiline hunt group to be tested for busy, regardless of the point of entry into the group. When a call is made to a line in an MLHG, a regular hunt is performed starting at the terminal associated with the dialed number. It continues to the last terminal in the MLHG, then proceeds to the first terminal in the group and continues to hunt sequentially through the remaining lines in the group. Busy tone is returnedif the called terminal is reached without finding one that is idle." This service is more suitable for a proxy server to handle. However, we can also handle it in an end system by using <transfer/> action.

```
<less>
  <incoming>
    <media type="audio" input="wait.au" mode="sendonly">
      <accept>
        <next>
          <lookup source="sip:bob@ex.com sip:tom@ex.com"
                  order="sequential">
            <success>
              <transfer>
                <succeed>
                  <stop/>
                </succeed>
              </transfer>
            </success>
          </lookup>
        </next>
      </accept>
    </media>
  </incoming>
</less>
```

17.2.14 Conference calling

"The service allows a station user to establish a conference call involving up to five other parties without attendant assistance." This service is almost the same as the attendant conferencing service we discussed before, it requires an end system to support mixing, and to merge an incoming call to an existing conference call. We use <merge/> action to handle the merge.

17.2.15 Customer-changeable speed calling

"This service allows subscribers to assign their own Speed Calling codes directly and immediately from their own telephone by dialing a change Speed Calling list access code, an abbreviated code and a new telephone number. It is available for 1- and or 2-digit Speed Calling list owners." In an end system, speed dialing modification should be done by editing service scripts from a local GUI. Below is a script for speed dialing handling.

```
<less>
  <UI:command command="call">
    <address-switch field="destination" subfield="user">
      <address is="1">
        <location url="sip:bob@example.com">
          <call/>
        </location>
      </address>
    </address-switch>
  </UI:command>
</less>
```

17.2.16 Direct connect

"This service automatically places a call to a preselected called number when a station goes off-hook. This feature can be used for introffice and interoffice calls and does not affect termination to a line." This service requires a specially configured UA. The service should not be handled by a service script.

17.2.17 Distinctive ringing

This service applies a distinctive ringing to determine the source of an incoming call. The following script handles the service.

```
<less>
  <incoming>
    <address-switch field="origin">
      <address is="sip:bob@example.com">
        <alert audio="bob.au"/>
      </address>
    </address-switch>
  </incoming>
</less>
```

17.2.18 Four- and eight-party

"This service provides POTS for up to four or eight customers sharing the same line. It provides fully selective ringing or semiselective ringing for up to four customers and semiselective ringing for five to eight customers." The following script plays distinctive ringing for different user sharing the same phone.

```
<less>
  <incoming>
    <address-switch field="destination">
      <address is="sip:bob@example.com">
        <alert audio="bob.au"/>
      </address>
      <otherwise>
        <address-switch field="destination">
          <address is="sip:tom@example.com">
            <alert audio="tom.au"/>
          </address>
          <otherwise>
            <address-switch field="destination">
              <address is="sip:mary@example.com">
                <alert audio="mary.au"/>
              </address>
            </address-switch>
          </otherwise>
        </address-switch>
      </otherwise>
    </address-switch>
  </incoming>
</less>
```

17.2.19 Recorded telephone dictation

"This service permits access to and control of customer-owned dictating equipment from a station in the customer group." This service requires an end system to be able to record and playback a phone call.

```
<less>
  <UI:command command="accept">
    <address-switch field="origin">
      <address is="sip:bob@example.com">
        <log record="true"/>
```

```
        </address>
      </address-switch>
    </UI:command>
  </less>
```

17.2.20 Time-of-Day features

"This kind of features can perform automatic actions based on time-of-day. For example, Slumber Service (also known as Do Not Disturb) can temporarily prohibit an individual customer station or a functional group of individual stations from receiving calls. It is used in hospitals to restrict incoming calls to patients during the night or any designated period." We can use LESS time-switch to handle the service.

```
<less>
  <incoming>
    <time-switch>
      <time dtstart="20040716T230000Z"
        duration="PT8H" freq="daily">
        <reject reason="Do not disturb"/>
      </time>
    </time-switch>
  </incoming>
</less>
```

17.2.21 Message services

ISDN Message Services provides centralized and personalized call coverage of message answering capabilities. For Basic Message Service, calls to a message service client are redirected via one of the following features.

Call forward -- busy line

Call forward -- don't answer

Call forward -- variable
To provide the personalized call coverage, message service attendants can be equipped with special ISDN station sets that display call information on calls directed to the message service center. The following information is available for display: Call type, Originating party DN, Calling party DN. This

service requires an end system can identify a message service, e.g., a voicemail server. The following script forward a call to a voicemail server when the user is busy.

```
<less>
  <incoming>
    <status-switch status-name="active-calls">
      <status greater="2">
        <location url="sip:bob@voicemail.example.com">
          <redirect/>
        </location>
      </status>
    </status-switch>
  </incoming>
</less>
```

There can be additional features for message services, for example, message waiting indicator, which can inform a client that he/she has message(s) waiting. The Message Waiting Indication Event Package for SIP [21] can provide required message information. An end system should be able to handle the event for the message waiting indicator service.

```
<less>
  <Event:notification>
    <Event:event-switch>
      <Event:event package="message-summary" message-waiting="yes">
        <alert message="There are new messages for you" audio="mwi.au"/>
      </Event:event>
    </Event:event-switch>
  </Event:notification>
</less>
```

17.2.22 Multibutton key telephone system (MBKS)

"Feature Function Buttons on the MBKS set can be assigned to activate certain features. For example, a user can press the function button assigned to Automatic Callback on Busy when a busy number is dialed." We can bind a script to a specific button for this kind of services.

17.2.23 Group alerting

"This service permits the controller (or alerting party), upon dialing a code, to signal a preselected number of telephones simultaneously. One-way communication is established from the controller to all members of the group that responded to the signal." The script below can handle the service.

```
<less>
  <UI:command command="call">
    <!-- define name="Alerting group"
         values="sip:security@abc.com sip:fire@abc.com" -->
    <address-switch field="destination">
      <address is="sip:alert@example.com">
        <lookup source="sip:security@abc.com sip:fire@abc.com">
          <media mode="sendonly">
            <call subject="Alert">
              <success>
                <merge subject="Alert"/>
              </success>
            </call>
          </media>
        </lookup>
      </address>
    </address-switch>
  </UI:command>
</less>
```

17.3 Services defined in CSTA Phase III

In this section, we investigate the services in ECMA standard ECMA-269, "Services for Computer Supported Telecommunications Applications (CSTA) Phase III" [18]. The call model of CSTA services is to use protocol messages to control end systems. Many CSTA services are in fact simply actions in LESS. For example, Accept Call is a CSTA control service, but just an action, <accept/>, in LESS.

CSTA Phase III categorizes services into System Services, Monitoring Services, Snapshot Services, Call Control Services and Events, Call Associated Features, Media Attachment Services and Events, Routing Services, Physical Device Features, Logical Device Features, Device Maintenance Events, I/O Services, Data Collection Services, Voice Services and Events, and Call Detail Record (CDR) Services. Among these services, System Services are used to handle the relationship between switch functions and CSTA service functions and are out of the scope of LESS, Monitoring Services are like to define the DPs (Detection Points) in Intelligent Network (IN) and are also out of

the scope of LESS. Snapshot Services are used to send device information to switch functions, not suitable for LESS. The Routing Services are not end system services and are also not suitable for LESS. The Device Maintenance Events, I/O Services, and Data Collection Services are too low-level for end user oriented service programming, thus they are not suitable for LESS. We will discuss the other services below.

17.3.1 Accept call

This service can be mapped to <alert/> action in LESS.

17.3.2 Alternate call

This service places an existing active call on hold and then retrieve a previously held call. This service in fact involves several actions. We can use the <Media:mediaupdate/> action to put a call held/unheld and put held calls into a queue by enqueue action.

17.3.3 Answer call

This service can be mapped to <accept/> action in LESS.

17.3.4 Call back call-related

The Call Back Call-Related service allows a computing function to request that the calling device retry the call to the called device when the called device is in an appropriate state to accept the call. This is the same as the ACB service described in Section 17.2.5.

17.3.5 Call back message call-related

The Call Back Message Call-Related service allows a computing function to request that the switching function leave a pre-defined message requesting that the called device call the calling device. For example, the called device may have been busy when called.

```
<less>
  <incoming>
    <status-switch status-name="activity">
      <status is="on-the-phone">
        <alert message="Please call back {message.origin}">
          <next>
            <Queue:enqueue queue="callback"/>
          </next>
        </alert>
      </status>
```

```
      </status-switch>
    </incoming>
  </less>
```

17.3.6 Camp on call

The Camp On Call service allows the computing function to queue a call for a device (that typically is busy) until that device becomes available. The script below can handle the service.

```
<less>
  <incoming>
    <status-switch status-name="active-name">
      <status greater="1">
        <media type="audio" input="wait.au" mode="sendonly">
          <accept>
            <next>
              <Queue:enqueue queue="callback"/>
            </next>
          </accept>
        </media>
      </status>
    </status-switch>
  </incoming>
</less>
```

17.3.7 Clear call

The Clear Call service releases all devices from an existing call. In the case of a conference call, this results in all devices in the conference call being released from the call. This service can be mapped to the <terminate/> action.

17.3.8 Clear connection

The Clear Connection service releases a specific device from a call. In the case of a two-party call, this may result in the call being torn down. In the case of a conference call, this results in the specific party being removed from the conference. For a two-party call, it is the same as Clear Call and can be represented by the <terminate/> action. For a conference call, we need to provide a parameter to the <terminate/> action to indicate which call leg should be disconnected. We can use a SIP url to identify the call

leg.

17.3.9 Conference call

The Conference Call service provides a conference of an existing held call and another active call at a conferencing device. This is the same as the 3-way calling service. It requires LESS to have a <merge/> command to merge multiple two-party calls into a conference call.

17.3.10 Consultation call

The Consultation Call service places an existing active call at a device on hold and initiates a new call from the same device. This is in fact two actions in LESS, using <Media:mediaupdate/> to put a call on hold and using <call/> to initiate a new call.

17.3.11 Deflect call

The Deflect Call service allows the computing function to divert a call to another destination that may be inside or outside the switching sub-domain. This can be mapped to LESS <transfer/> action.

17.3.12 Dial digits

The Dial Digits service allows the computing function to perform a dialing sequence that is associated with a call that has already been initiated. This service is used to send message out after a call invitation and we use <IM:sendmsg> to handle the action.

17.3.13 Directed pickup call

The Directed Pickup Call service moves a specified call and connects it at a new specified destination. This results in the connection being diverted to a new destination inside the switching sub-domain. This service should not be handled by end systems, but by an application server in the network.

17.3.14 Group pickup call

The Group Pickup Call service moves a call that is a member of a specified or default pickup group to a new specified destination. The same as Directed Pickup Call, this service should be handled by an application server in the network.

17.3.15 Hold call

The Hold Call service places a connected connection on hold at the

same device. The <Media:mediaupdate/> action in LESS handles this service and <enqueue> action can put a call in a queue.

17.3.16 Intrude call

The Intrude Call service adds the calling device to a call at a busy called device. Depending upon the switching function, the result will be that the calling device is either actively or silently participating in the called device's existing call or consulting with the called device with a new call. This can be handled by LESS <merge/> action with the <media> modifier set as mode="sendonly" for silently participating.

17.3.17 Join call

The Join Call service allows a computing function to request, on behalf of a device, that the device be joined into an existing call. This can also be handled by LESS <merge/> action.

17.3.18 Make call

The Make Call service allows the computing function to set up a call between a calling device and a called device. This can be handled by LESS <call/> action.

17.3.19 Make predictive call

The Make Predictive Call service shall originate a call between two devices by first creating a connection to the called device. The service returns a positive acknowledgement that provides the connection at the called device. Subsequent actions are taken depending upon the call progress and the actions requested. This service makes additional actions based on the result of <call/> action. For example,

```
<less>
  <UI:command command="call">
    <call>
      <success>
        ....
      </success>
      <redirected>
        ...
      </redirected>
      <rejected>
        ...
      </rejected>
```

```
      </call>
    </UI:command>
  </less>
```

17.3.20 Park call

The Park Call service moves a specified call at a device to a specified (parked-to) destination. Two ways to handle this in an end system, the first is to transfer the call to a resource server, the second is to change the media input of the call as described in Section 17.1.10.

17.3.21 Reconnect call

The Reconnect Call service will clear a specified connection at the reconnecting device and retrieve a specified held connection at the same device. This service can be handled by using the action <terminate/> to terminate an existing call and using <Media:mediaupdate/> and <dequeue> to retrieve a held call.

17.3.22 Retrieve call

The Retrieve Call service connects a specified held connection. This can be handled by <dequeue> and <Media:mediaupdate/> actions.

17.3.23 Send message

The Send Message service allows the computing function to send a message to one or more devices. The message, composed of one or more MIME body parts, is included in the Send Message service request. This service can be used to send messages for many different types of applications (Instant Messaging, Email, Pager, and Short Message Service (SMS), etc). In LESS, we use <Message:sendmsg/> action to send a message, and <Email:send/> to send an email.

17.3.24 Single step conference call

The Single Step Conference Call joins a new device into an existing call. This service can be repeated to make n-device conference calls (subject to switching function limits). This can be handled by the action <merge/>.

17.3.25 Single step transfer call

The Single Step Transfer Call service transfers an existing connection at a device to another device. This transfer is performed

in a single-step, that is the device doing the transfer does not have to place the existing call on hold before issuing the Single Step Transfer Call service. This can be handled by the action <transfer/>.

17.3.26 Transfer call

The Transfer Call service transfers a call held at a device to an active call at the same device. The held and active calls at the transferring device shall be merged into a new call. Also, the Connections of the held and active calls at the transferring device shall become Null and their ConnectionIDs shall be released (i.e., the transferring device is no longer involved with the call). This service is in fact merge a held call and an existing call into a 3-way calling. It should be handled by LESS <merge> action.

17.4 New services

One of the biggest advantage of Internet telephony is its ability to easily integrate other Internet services, such as presence information, email, and web. The integration can introduce many new services which are impossible for PSTN networks. These new services are not mentioned in Q.1211 or 5ESS service documents.

17.4.1 Email

For email, we usually care about three things, receivers, subject, and content. We can use <location> or <lookup> modifier to specify receivers, use subject parameter in <email> action for subject information, and put content as the data of an <email> tag. For example,

```
<location url="mailto:bob@example.com">
  <Email:send subject="This is a test">
    We simply send this as a test for our LESS scripts.
  </Email:send>
</location>
```

There is no output of the email action.

17.4.2 Web

There are two parameters for a web service action, one is the URI to visit, the other is the HTTP method. The URI parameter can be handled by the <location> and <lookup> modifiers. We use different action to

represent different HTTP methods. For example, <Web:get> represents HTTP GET method, <Web:post> represents HTTP POST method. Below is a service example,

```
<location uri="http://www.example.com">
  <Web:get/>
</location>
```

The output of the a web action can be succeed, redirect, or failed.

17.4.3 Instant messaging

People use instant messaging as a complementary communication method for voice call and email. We introduce two toplevel actions <IM::message-coming> for incoming message, <UI:command command="sendmsg"> for an outgoing message, and one action <IM::sendmsg> for sending a message out. For example,

```
<less>
  <IM:message-coming>
    <address-switch field="origin">
      <address is="sip:bob@example.com">
        <location url="sip:bob@example.com">
          <IM:sendmsg>Hi, Bob</IM:sendmsg>
        </location>
      </address>
    </address-switch>
  </IM:message-coming>
</less>
```

17.4.4 Event subscription and notification

Below is an example for address-based event subscription handling.

```
<Event:subscription>
  <address-switch field="origin">
    <address is="sip:bob@example.com">
      <Event:approve>
        <next>
          <Event:notify status="closed"/>
        </next>
```

```
        </Event:approve>
      </address>
    </address-switch>
  </Event:subscription>
```

17.4.5 Event notification and event-based services

Event notifications can trigger many new services. For example,

```
<Event:notification>
  <address-switch field="origin">
    <address is="sip:bob@example.com">
      <Event:event-switch>
        <Event:event status="online">
          <location url="sip:bob@example.com">
            <call/>
          </location>
        </Event:event>
      </Event:event-switch>
    </address>
  </address-switch>
</Event:notification>
```

17.4.6 Location-based services

Location information gets more and more important in people's communications. It is not only used for tracking, but also for guarding appropriate communication behaviors, triggering communication actions, and discovering available resources in a context. Below is an example showing location-based call decision making.

```
<incoming>
  <Location:where-switch>
    <Location:where placytype="movie-theatre">
      <alert style="vibrate"/>
    </Location:where>
  </Location:where-switch>
</incoming>
```

18 Feature Interaction Handling Algorithm for LESS Scripts

For LESS-based services, due to the tree-like structure of LESS, we consider it simple and efficient to design an algorithm to merge multiple LESS decision trees into one. After merging, for a specific trigger, there is only one active LESS script at a device. The merging algorithm is good for single component services. For service scripts on different devices, for example, the single user multiple components (SUMC) call control services the merging algorithm can only detect feature interactions, it cannot merge all the scripts into one to resolve the interactions. After merging, we will still keep the original scripts so users can modify them independently. This way, no conflicts in service execution because for a given trigger, it only needs to go through one decision tree to perform services. In the mean time, we can still ensure the service modularity so users can create their services efficiently.

18.1 Tree merging algorithm

The overall multi-script merging process is as below.

```
set base-rule-set empty
foreach LESS-tree {
 convert the LESS-tree into a rule set
 foreach rule in the rule set {
  normalize the rule
 }
 merge the normalized rule set into base-rule-set
}
convert base-rule-set into a decision tree
```

The merging operation is in fact to merge every set of rules into base-rule-set, then convert the base-rule-set back to a decision tree. We will explain every step of the process below.

A rule of a LESS decision tree is defined as which actions get executed under a certain condition for a specific trigger. Based on the LESS design principles we introduced in Section 2.1, the path from the root of a decision tree to each leaf node consists a decision rule. We construct a rule as a composition of a trigger, the actions in accordance to the trigger, and a list of switch nodes that in the path from the root to the action node, we name the list of switch nodes rule path. For example, for the script below, we can represent a decision rule as "{incoming,accept,{{string-switch,organization="ABC Inc."},{address-switch,

```
origin="sip:tom@abc.com"},{string-switch,subject="group meeting"}}}".


<less>
  <incoming>
    <string-switch field="organization">
      <string is="ABC Inc.">
        <address-switch field="origin">
          <address is="sip:tom@abc.com">
            <string-switch field="subject">
              <string is="group meeting">
                <accept/>
              </string>
            </string-switch>
          </address>
          <otherwise>
            <location url="sip:tom@voicemail.abc.com">
              <redirect/>
            </location>
          </otherwise>
        </address-switch>
      </string>
    </string-switch>
  </incoming>
</less>
```

To facilitate rule merging, we need to normalize the rules generated from LESS decision trees. The normalization process sorts rule paths to a specific order, e.g., in the order of "address-switch", "time-switch", "status-switch", "string-switch", "priority-switch", "where-switch", and "language-switch". It will also merge the switch nodes with the same switch name into one node in a rule path. For example, a normalized rule for the script above is "{incoming, accept, {address-switch, origin="sip:tom@abc.com"}, {string-switch, subject="ABC group meeting", organization="ABC Inc."}}". Because switches are independent of each other, normalized rules are functionally equal to original rules. After normalization, we can follow the process below to detect feature interactions between two normalized rules.

```
if (two rules have different triggers) {
 no rule conflict
} elseif actions in two rules do not conflict {
 no rule conflict
} elseif no overlap between rule path in two rules {
```

```
 no rule conflict
} else {
 two rules conflict with each other,
 return the rule path overlap and action conflict information
 prompt to the script owner to judge
}
```

During the process, we use the action conflict table defined in our technical report "Feature Interactions in Internet Telephony End Systems" [15]. to check whether two actions interact with each other or not. If actions in two rules conflict with each other, we need to check whether there are conditions matching both rule paths. We name this kind of conditions the overlap between two rule paths. We employ the following algorithm to get the overlap.

```
set overlap-set empty
foreach switch-node1 in rule-path1 {
 if there is a switch-node2 in rule-path2 has the same switch name {
  if the switch-overlap between switch-node1 and switch-node2 is empty {
   return empty overlap-set
  } else {
   insert the switch-overlap into overlap-set
  }
 } else {
  insert switch-node1 into overlap-set
 }
}
foreach switch-node2 in rule-path2 {
 if there is not a switch-node1 in rule-path1 has the same switch name {
  insert switch-node2 into overlap-set
 }
}
return overlap-set
```

During the overlap handling process, different switch types have different overlap detection method. We will not detail the overlap handling for each switch type in this paper.

Once we find the overlap-set and the conflicting actions, we can present the information to users to make decisions. We can record the decision made by the user and build a normalized rule set without conflicts.

To convert a set of normalized rules back to a decision tree is straightforward. The pseudo code below shows the algorithm.

```
set tree empty
foreach rule in the merged rule set {
 foreach switch node in the rule path {
  //since each rule is normalized, the switch node appears in order
  go along the tree from the root
  if the switch node matches a tree node {
   go to the matched branch, and continue for the next switch node
  } else {
   if there is an unmatched branch {
    go to unmatched branch, continue the test
   } else {
    create an unmatched branch
    put the rest rule path (including current switch node)
    in the unmatched branch
   }}}}
```

19 The XML Schema for LESS and Commonly Used Extensions

19.1 XML Schema for LESS

```
   <?xml version="1.0" encoding="UTF-8"?>
   <!-- edited with XMLSPY v2004 rel. 2 U (http://www.xmlspy.com) by Xiaotao Wu
(Columbia University Computer Science Dept) -->
   <xs:schema targetNamespace="urn:ietf:params:xml:ns:less" xmlns:xs="http://
www.w3.org/2001/XMLSchema" xmlns="urn:ietf:params:xml:ns:less"
elementFormDefault="qualified" attributeFormDefault="unqualified">
     <xs:complexType name="TriggerType" abstract="true">
       <xs:group ref="Node"/>
     </xs:complexType>
     <xs:element name="trigger" type="TriggerType"/>
     <xs:complexType name="ActionType" abstract="true" mixed="true">
       <xs:all>
         <xs:element name="next" minOccurs="0">
           <xs:complexType>
             <xs:group ref="Node"/>
           </xs:complexType>
         </xs:element>
       </xs:all>
     </xs:complexType>
     <xs:element name="action" type="ActionType"/>
     <xs:complexType name="SwitchType" abstract="true"/>
     <xs:element name="switch" type="SwitchType"/>
     <xs:complexType name="ModifierType" abstract="true"/>
     <xs:element name="modifier" type="ModifierType"/>
     <xs:complexType name="SubAction">
       <xs:attribute name="ref" type="xs:string" use="required"/>
     </xs:complexType>
     <xs:element name="sub" type="SubAction"/>
     <xs:complexType name="SubactionType">
       <xs:group ref="Node"/>
       <xs:attribute name="id" use="required"/>
     </xs:complexType>
     <xs:complexType name="AncillaryType"/>
     <xs:group name="Node">
       <xs:choice>
         <xs:element ref="switch" minOccurs="0"/>
         <xs:element ref="modifier" minOccurs="0" maxOccurs="unbounded"/>
         <xs:element ref="sub" minOccurs="0"/>
         <xs:element ref="action" minOccurs="0" maxOccurs="unbounded"/>
       </xs:choice>
     </xs:group>
     <xs:complexType name="OtherwiseAction">
       <xs:group ref="Node"/>
     </xs:complexType>
     <xs:complexType name="NotPresentAction">
       <xs:group ref="Node"/>
     </xs:complexType>
     <xs:complexType name="LESSType">
       <xs:sequence>
         <xs:element name="ancillary" type="AncillaryType" minOccurs="0"/>
```

```
         <xs:element name="subaction" type="SubactionType" minOccurs="0"
maxOccurs="unbounded"/>
         <xs:element ref="trigger" minOccurs="0" maxOccurs="unbounded">
           <xs:annotation>
             <xs:documentation>Any toplevel action MUST NOT appear more than
once.</xs:documentation>
           </xs:annotation>
         </xs:element>
       </xs:sequence>
     </xs:complexType>
     <xs:element name="less" type="LESSType"/>
     <xs:simpleType name="YesNoType">
       <xs:restriction base="xs:NMTOKEN">
         <xs:enumeration value="yes"/>
         <xs:enumeration value="no"/>
       </xs:restriction>
     </xs:simpleType>
     <xs:simpleType name="StatusType">
       <xs:union>
         <xs:simpleType>
           <xs:restriction base="xs:NMTOKEN">
             <xs:enumeration value="busy"/>
             <xs:enumeration value="notfound"/>
             <xs:enumeration value="reject"/>
             <xs:enumeration value="error"/>
           </xs:restriction>
         </xs:simpleType>
         <xs:simpleType>
           <xs:restriction base="xs:string"/>
         </xs:simpleType>
       </xs:union>
     </xs:simpleType>
     <xs:complexType name="IncomingType">
       <xs:complexContent>
         <xs:extension base="TriggerType"/>
       </xs:complexContent>
     </xs:complexType>
     <xs:element name="incoming" type="IncomingType"
substitutionGroup="trigger"/>
     <xs:complexType name="TimerType">
       <xs:complexContent>
         <xs:extension base="TriggerType">
           <xs:attribute name="dtstart" type="xs:string" use="required">
             <xs:annotation>
               <xs:documentation>RFC 2445 DATE-TIME</xs:documentation>
             </xs:annotation>
           </xs:attribute>
           <xs:attribute name="dtend" type="xs:string" use="optional">
             <xs:annotation>
               <xs:documentation>RFC 2445 DATE-TIME</xs:documentation>
             </xs:annotation>
```

```
          </xs:attribute>
          <xs:attribute name="duration" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>RFC 2445 DURATION</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="freq" type="FreqType" use="optional"/>
          <xs:attribute name="interval" type="xs:positiveInteger" default="1"/
>
          <xs:attribute name="until" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>RFC 2445 DATE-TIME</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="count" type="xs:positiveInteger" use="optional"/
>
          <xs:attribute name="bysecond" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Comma-separated list of seconds within a
minute. Valid values are 0 to 59.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="byminute" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Comma-separated list of minutes within an
hour. Valid values are 0 to 59.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="byhour" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Comma-separated list of hours of the day.
Valid values are 0 to 23.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="byday" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Comma-separated list of days of the week.
Valid values are "MO", "TU", "WE", "TH", "FR", "SA" and "SU". These values are
not case-sensitive. Each can be preceded by a positive (+n) or negative (-n)
integer.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="bymonthday" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Comma-separated list of days of the month.
Valid values are 1 to 31 or -31 to -1.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="byyearday" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Comma-separated list of days of the year.
Valid values are 1 to 366 or -366 to -1.</xs:documentation>
            </xs:annotation>
```

```
          </xs:attribute>
          <xs:attribute name="byweekno" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Comma-separated list of ordinals specifying
weeks of the year. Valid values are 1 to 53 or -53 to -1.</xs:documentation>
            </xs:annotation>
```

```
          </xs:attribute>
          <xs:attribute name="bymonth" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Comma-separated list of months of the year.
Valid values are 1 to 12.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="wkst" type="DayType" default="MO"/>
          <xs:attribute name="bysetpos" type="YearDayType"/>
          <xs:anyAttribute namespace="##any" processContents="lax"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="timer" type="TimerType" substitutionGroup="trigger"/>
    <xs:simpleType name="OrderingType">
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="parallel"/>
        <xs:enumeration value="sequential"/>
        <xs:enumeration value="first-only"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="AddressFieldType">
      <xs:union>
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="origin"/>
            <xs:enumeration value="destination"/>
            <xs:enumeration value="original-destination"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:union>
    </xs:simpleType>
    <xs:simpleType name="AddressSubfieldType">
      <xs:union>
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="address-type"/>
            <xs:enumeration value="user"/>
            <xs:enumeration value="host"/>
            <xs:enumeration value="port"/>
            <xs:enumeration value="tel"/>
            <xs:enumeration value="display"/>
            <xs:enumeration value="password"/>
            <xs:enumeration value="alias-type"/>
          </xs:restriction>
        </xs:simpleType>
```

```
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:union>
    </xs:simpleType>
    <xs:complexType name="AddressType">
      <xs:annotation>
        <xs:documentation>Exactly one of the three attributes must appear</
xs:documentation>
      </xs:annotation>
      <xs:group ref="Node"/>
      <xs:attribute name="is" type="xs:string" use="optional"/>
      <xs:attribute name="contains" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>for "display" only</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="subdomain-of" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>for "host", "tel" only</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="AddressSwitchType">
      <xs:complexContent>
        <xs:extension base="SwitchType">
          <xs:sequence>
            <xs:element name="address" type="AddressType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:sequence minOccurs="0">
              <xs:element name="not-present" type="NotPresentAction"/>
              <xs:element name="address" type="AddressType" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:element name="otherwise" type="OtherwiseAction" minOccurs="0"/
>
          </xs:sequence>
          <xs:attribute name="field" type="AddressFieldType" use="required"/>
          <xs:attribute name="subfield" type="AddressSubfieldType"
use="optional"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="address-switch" type="AddressSwitchType"
substitutionGroup="switch"/>
    <xs:simpleType name="StringFieldType">
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="subject"/>
        <xs:enumeration value="organization"/>
        <xs:enumeration value="user-agent"/>
        <xs:enumeration value="display"/>
      </xs:restriction>
```

```
    </xs:simpleType>
```

```
     <xs:complexType name="StringType">
       <xs:group ref="Node"/>
       <xs:attribute name="is" type="xs:string" use="optional"/>
       <xs:attribute name="contains" type="xs:string" use="optional"/>
       <xs:anyAttribute namespace="##any" processContents="lax"/>
     </xs:complexType>
     <xs:complexType name="StringSwitchType">
       <xs:complexContent>
         <xs:extension base="SwitchType">
           <xs:sequence>
             <xs:element name="string" type="StringType" minOccurs="0"
maxOccurs="unbounded"/>
             <xs:sequence minOccurs="0">
               <xs:element name="not-present" type="NotPresentAction"/>
               <xs:element name="string" type="StringType" minOccurs="0"
maxOccurs="unbounded"/>
             </xs:sequence>
             <xs:element name="otherwise" type="OtherwiseAction" minOccurs="0"/
>
           </xs:sequence>
           <xs:attribute name="field" type="StringFieldType" use="required">
             <xs:annotation>
               <xs:documentation>Strings are matched as case-insensitive
Unicode strings.</xs:documentation>
             </xs:annotation>
           </xs:attribute>
         </xs:extension>
       </xs:complexContent>
     </xs:complexType>
     <xs:element name="string-switch" type="StringSwitchType"
substitutionGroup="switch"/>
     <xs:complexType name="LanguageType">
       <xs:group ref="Node"/>
       <xs:attribute name="matches" type="xs:string" use="required">
         <xs:annotation>
           <xs:documentation>The value of one of these parameters is a
language-tag, as defined in RFC 3066.</xs:documentation>
         </xs:annotation>
       </xs:attribute>
       <xs:anyAttribute namespace="##any" processContents="lax"/>
     </xs:complexType>
     <xs:complexType name="LanguageSwitchType">
       <xs:complexContent>
         <xs:extension base="SwitchType">
           <xs:sequence>
             <xs:element name="language" type="LanguageType" minOccurs="0"
maxOccurs="unbounded"/>
             <xs:sequence minOccurs="0">
               <xs:element name="not-present" type="NotPresentAction"/>
               <xs:element name="language" type="LanguageType" minOccurs="0"
maxOccurs="unbounded"/>
             </xs:sequence>
             <xs:element name="otherwise" type="OtherwiseAction" minOccurs="0"/
```

```
>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
```

```
     </xs:complexType>
     <xs:element name="language-switch" type="LanguageSwitchType"
substitutionGroup="switch"/>
     <xs:simpleType name="FreqType">
       <xs:restriction base="xs:NMTOKEN">
         <xs:pattern value="[s|S][e|E][c|C][o|O][n|N][d|D][l|L][y|Y]"/>
         <xs:pattern value="[m|M][i|I][n|N][u|U][t|T][e|E][l|L][y|Y]"/>
         <xs:pattern value="[h|H][o|O][u|U][r|R][l|L][y|Y]"/>
         <xs:pattern value="[d|D][a|A][i|I][l|L][y|Y]"/>
         <xs:pattern value="[w|W][e|E][e|E][k|K][l|L][y|Y]"/>
         <xs:pattern value="[m|M][o|N][n|N][t|T][h|H][l|L][y|Y]"/>
         <xs:pattern value="[y|Y][e|E][a|A][r|R][l|L][y|Y]"/>
       </xs:restriction>
     </xs:simpleType>
     <xs:simpleType name="YearDayType">
       <xs:union>
         <xs:simpleType>
           <xs:restriction base="xs:integer">
             <xs:minInclusive value="-366"/>
             <xs:maxInclusive value="-1"/>
           </xs:restriction>
         </xs:simpleType>
         <xs:simpleType>
           <xs:restriction base="xs:integer">
             <xs:minInclusive value="1"/>
             <xs:maxExclusive value="366"/>
           </xs:restriction>
         </xs:simpleType>
       </xs:union>
     </xs:simpleType>
     <xs:simpleType name="DayType">
       <xs:restriction base="xs:NMTOKEN">
         <xs:pattern value="[m|M][o|O]"/>
         <xs:pattern value="[t|T][u|U]"/>
         <xs:pattern value="[w|W][e|E]"/>
         <xs:pattern value="[t|T][h|H]"/>
         <xs:pattern value="[f|F][r|R]"/>
         <xs:pattern value="[s|S][a|A]"/>
         <xs:pattern value="[s|S][u|U]"/>
       </xs:restriction>
     </xs:simpleType>
     <xs:complexType name="TimeType">
       <xs:annotation>
         <xs:documentation>Exactly one of the two attributes "dtend" and
"duration" must occur. None of the attributes following freq are meaningful
unless freq appears. </xs:documentation>
       </xs:annotation>
       <xs:group ref="Node"/>
       <xs:attribute name="dtstart" type="xs:string" use="required">
         <xs:annotation>
           <xs:documentation>RFC 2445 DATE-TIME</xs:documentation>
```

```
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="dtend" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>RFC 2445 DATE-TIME</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="duration" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>RFC 2445 DURATION</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="freq" type="FreqType" use="optional"/>
      <xs:attribute name="interval" type="xs:positiveInteger" default="1"/>
      <xs:attribute name="until" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>RFC 2445 DATE-TIME</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="count" type="xs:positiveInteger" use="optional"/>
      <xs:attribute name="bysecond" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>Comma-separated list of seconds within a minute.
Valid values are 0 to 59.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="byminute" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>Comma-separated list of minutes within an hour.
Valid values are 0 to 59.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="byhour" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>Comma-separated list of hours of the day. Valid
values are 0 to 23.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="byday" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>Comma-separated list of days of the week. Valid
values are "MO", "TU", "WE", "TH", "FR", "SA" and "SU". These values are not
case-sensitive. Each can be preceded by a positive (+n) or negative (-n)
integer.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="bymonthday" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>Comma-separated list of days of the month. Valid
values are 1 to 31 or -31 to -1.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="byyearday" type="xs:string" use="optional">
```

```
         <xs:annotation>
           <xs:documentation>Comma-separated list of days of the year. Valid
values are 1 to 366 or -366 to -1.</xs:documentation>
```

```
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="byweekno" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>Comma-separated list of ordinals specifying weeks
of the year. Valid values are 1 to 53 or -53 to -1.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="bymonth" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>Comma-separated list of months of the year. Valid
values are 1 to 12.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="wkst" type="DayType" default="MO"/>
      <xs:attribute name="bysetpos" type="YearDayType"/>
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>
    <xs:simpleType name="TZIDType">
      <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="TZURLType">
      <xs:restriction base="xs:anyURI"/>
    </xs:simpleType>
    <xs:complexType name="TimeSwitchType">
      <xs:complexContent>
        <xs:extension base="SwitchType">
          <xs:sequence>
            <xs:element name="time" type="TimeType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:sequence minOccurs="0">
              <xs:element name="not-present" type="NotPresentAction"/>
              <xs:element name="time" type="TimeType" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:element name="otherwise" type="OtherwiseAction" minOccurs="0"/
>
          </xs:sequence>
          <xs:attribute name="tzid" type="TZIDType"/>
          <xs:attribute name="tzurl" type="TZURLType"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="time-switch" type="TimeSwitchType"
substitutionGroup="switch"/>
    <xs:simpleType name="PriorityValues">
      <xs:restriction base="xs:NMTOKEN">
        <xs:pattern value="[e|E][m|M][e|E][r|R][g|G][e|E][n|N][c|C][y|Y]"/>
        <xs:pattern value="[u|U][r|R][g|G][e|E][n|N][t|T]"/>
        <xs:pattern value="[n|N][o|O][r|R][m|M][a|A][l|L]"/>
        <xs:pattern value="[n|N][o|O][n|N]-[u|U][r|R][g|G][e|E][n|N][t|T]"/>
      </xs:restriction>
    </xs:simpleType>
```

```
     <xs:complexType name="PriorityType">
```

```
       <xs:annotation>
         <xs:documentation>Exactly one of the three attributes must appear </
xs:documentation>
       </xs:annotation>
       <xs:group ref="Node"/>
       <xs:attribute name="less" type="PriorityValues"/>
       <xs:attribute name="greater" type="PriorityValues"/>
       <xs:attribute name="equal" type="xs:string">
         <xs:annotation>
           <xs:documentation>case-insensitive</xs:documentation>
         </xs:annotation>
       </xs:attribute>
       <xs:anyAttribute namespace="##any" processContents="lax"/>
     </xs:complexType>
     <xs:complexType name="PrioritySwitchType">
       <xs:complexContent>
         <xs:extension base="SwitchType">
           <xs:sequence>
             <xs:element name="priority" type="PriorityType" minOccurs="0"
maxOccurs="unbounded"/>
             <xs:sequence minOccurs="0">
               <xs:element name="not-present" type="NotPresentAction"/>
               <xs:element name="priority" type="PriorityType" minOccurs="0"
maxOccurs="unbounded"/>
             </xs:sequence>
             <xs:element name="otherwise" type="OtherwiseAction" minOccurs="0"/
>
           </xs:sequence>
         </xs:extension>
       </xs:complexContent>
     </xs:complexType>
     <xs:element name="priority-switch" type="PrioritySwitchType"
substitutionGroup="switch"/>
     <xs:complexType name="UserStatusType">
       <xs:group ref="Node"/>
       <xs:attribute name="less" type="xs:integer"/>
       <xs:attribute name="greater" type="xs:integer"/>
       <xs:attribute name="equal" type="xs:integer"/>
       <xs:attribute name="is">
         <xs:simpleType>
           <xs:list itemType="xs:string"/>
         </xs:simpleType>
       </xs:attribute>
       <xs:attribute name="contains">
         <xs:simpleType>
           <xs:list itemType="xs:string"/>
         </xs:simpleType>
       </xs:attribute>
     </xs:complexType>
     <xs:complexType name="StatusSwitchType">
       <xs:complexContent>
         <xs:extension base="SwitchType">
           <xs:sequence>
```

```
            <xs:element name="status" type="UserStatusType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:sequence minOccurs="0">
              <xs:element name="not-present" type="NotPresentAction"/>
              <xs:element name="status" type="UserStatusType" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:element name="otherwise" type="OtherwiseAction" minOccurs="0"/
>
          </xs:sequence>
          <xs:attribute name="uri" type="xs:anyURI" use="optional"/>
          <xs:attribute name="status-name">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="active-calls"/>
                <xs:enumeration value="presence"/>
                <xs:enumeration value="activity"/>
                <xs:enumeration value="mood"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="status-switch" type="StatusSwitchType"
substitutionGroup="switch"/>
    <xs:simpleType name="LocationPriorityType">
      <xs:restriction base="xs:float">
        <xs:minInclusive value="0.0"/>
        <xs:maxInclusive value="1.0"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="LocationType">
      <xs:complexContent>
        <xs:extension base="ModifierType">
          <xs:group ref="Node"/>
          <xs:attribute name="url" type="xs:anyURI" use="required"/>
          <xs:attribute name="priority" type="LocationPriorityType"
use="optional" default="1.0"/>
          <xs:attribute name="clear" type="YesNoType" default="no"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="location" type="LocationType"
substitutionGroup="modifier"/>
    <xs:complexType name="LookupType">
      <xs:complexContent>
        <xs:extension base="ModifierType">
          <xs:all>
            <xs:element name="success" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
```

```
            </xs:element>
```

```
            <xs:element name="notfound" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="failure" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
          </xs:all>
          <xs:attribute name="source" type="xs:string" use="required"/>
          <xs:attribute name="order" use="optional" default="parallel">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="parallel"/>
                <xs:enumeration value="sequential"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="timeout" type="xs:positiveInteger" default="30"/
>
          <xs:attribute name="clear" type="YesNoType" default="no"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="lookup" type="LookupType" substitutionGroup="modifier"/>
    <xs:complexType name="RemoveLocationType">
      <xs:complexContent>
        <xs:extension base="ModifierType">
          <xs:group ref="Node"/>
          <xs:attribute name="location" type="xs:string" use="optional"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="remove-location" type="RemoveLocationType"
substitutionGroup="modifier"/>
    <xs:complexType name="LogAction">
      <xs:complexContent>
        <xs:extension base="ActionType">
          <xs:group ref="Node"/>
          <xs:attribute name="name" type="xs:string" use="optional"/>
          <xs:attribute name="comment" type="xs:string" use="optional"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="log" type="LogAction" substitutionGroup="action"/>
    <xs:complexType name="MailAction">
      <xs:complexContent>
        <xs:extension base="ActionType">
```

```
          <xs:group ref="Node"/>
          <xs:attribute name="url" type="xs:anyURI" use="required"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="mail" type="MailAction" substitutionGroup="action"/>
    <xs:complexType name="WaitAction">
      <xs:complexContent>
        <xs:extension base="ActionType">
          <xs:group ref="Node"/>
          <xs:attribute name="duration" type="xs:duration" use="required"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="wait" type="WaitAction" substitutionGroup="action"/>
    <xs:complexType name="AcceptActionType">
      <xs:complexContent>
        <xs:extension base="ActionType"/>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="accept" type="AcceptActionType"
substitutionGroup="action"/>
    <xs:complexType name="RedirectAction">
      <xs:complexContent>
        <xs:extension base="ActionType">
          <xs:attribute name="permanent" type="YesNoType" default="no"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="redirect" type="RedirectAction"
substitutionGroup="action"/>
    <xs:complexType name="RejectAction">
      <xs:complexContent>
        <xs:extension base="ActionType">
          <xs:attribute name="status" type="StatusType" use="required"/>
          <xs:attribute name="reason" type="xs:string" use="optional"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="reject" type="RejectAction" substitutionGroup="action"/>
    <xs:complexType name="AuthenticateAction">
      <xs:complexContent>
        <xs:extension base="ActionType">
          <xs:all>
            <xs:element name="success" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="failure" minOccurs="0">
```

```
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
          </xs:all>
          <xs:attribute name="method" use="optional" default="digest">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="digest"/>
                <xs:enumeration value="basic"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="realm" type="xs:string"/>
          <xs:attribute name="user" type="xs:string"/>
          <xs:attribute name="target">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="remote"/>
                <xs:enumeration value="local"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="authenticate" type="AuthenticateAction"
substitutionGroup="action"/>
    <xs:complexType name="CallActionType">
      <xs:complexContent>
        <xs:extension base="ActionType">
          <xs:all>
            <xs:element name="accepted" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="busy" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="noanswer" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="failure" minOccurs="0">
              <xs:complexType>
```

```
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="redirection" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
          </xs:all>
          <xs:attribute name="timeout" type="xs:positiveInteger"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="call" type="CallActionType" substitutionGroup="action"/>
    <xs:complexType name="TerminateActionType">
      <xs:complexContent>
        <xs:extension base="ActionType">
          <xs:group ref="Node"/>
          <xs:attribute name="uri">
            <xs:simpleType>
              <xs:list itemType="xs:anyURI"/>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="subject">
            <xs:simpleType>
              <xs:list itemType="xs:string"/>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="calls">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="all"/>
                <xs:enumeration value="last"/>
                <xs:enumeration value="this"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="terminate" type="TerminateActionType"
substitutionGroup="action"/>
  </xs:schema>
```

19.2 XML Schema for LESS Media Extension

```
   <?xml version="1.0" encoding="UTF-8"?>
   <!-- edited with XMLSPY v2004 rel. 2 U (http://www.xmlspy.com) by Xiaotao Wu
(Columbia University Computer Science Dept) -->
   <xs:schema targetNamespace="urn:ietf:params:xml:ns:less" xmlns:xs="http://
www.w3.org/2001/XMLSchema" xmlns="urn:ietf:params:xml:ns:less"
elementFormDefault="qualified" attributeFormDefault="unqualified">
     <xs:complexType name="TriggerType" abstract="true">
       <xs:group ref="Node"/>
     </xs:complexType>
     <xs:element name="trigger" type="TriggerType"/>
     <xs:complexType name="ActionType" abstract="true" mixed="true">
       <xs:all>
         <xs:element name="next" minOccurs="0">
           <xs:complexType>
             <xs:group ref="Node"/>
           </xs:complexType>
         </xs:element>
       </xs:all>
     </xs:complexType>
     <xs:element name="action" type="ActionType"/>
     <xs:complexType name="SwitchType" abstract="true"/>
     <xs:element name="switch" type="SwitchType"/>
     <xs:complexType name="ModifierType" abstract="true"/>
     <xs:element name="modifier" type="ModifierType"/>
     <xs:complexType name="SubAction">
       <xs:attribute name="ref" type="xs:string" use="required"/>
     </xs:complexType>
     <xs:element name="sub" type="SubAction"/>
     <xs:complexType name="SubactionType">
       <xs:group ref="Node"/>
       <xs:attribute name="id" use="required"/>
     </xs:complexType>
     <xs:complexType name="AncillaryType"/>
     <xs:group name="Node">
       <xs:choice>
         <xs:element ref="switch" minOccurs="0"/>
         <xs:element ref="modifier" minOccurs="0" maxOccurs="unbounded"/>
         <xs:element ref="sub" minOccurs="0"/>
         <xs:element ref="action" minOccurs="0" maxOccurs="unbounded"/>
       </xs:choice>
     </xs:group>
     <xs:complexType name="OtherwiseAction">
       <xs:group ref="Node"/>
     </xs:complexType>
     <xs:complexType name="NotPresentAction">
       <xs:group ref="Node"/>
     </xs:complexType>
     <xs:complexType name="LESSType">
       <xs:sequence>
         <xs:element name="ancillary" type="AncillaryType" minOccurs="0"/>
```

```
         <xs:element name="subaction" type="SubactionType" minOccurs="0"
maxOccurs="unbounded"/>
         <xs:element ref="trigger" minOccurs="0" maxOccurs="unbounded">
           <xs:annotation>
             <xs:documentation>Any toplevel action MUST NOT appear more than
once.</xs:documentation>
           </xs:annotation>
         </xs:element>
       </xs:sequence>
     </xs:complexType>
     <xs:element name="less" type="LESSType"/>
     <xs:simpleType name="YesNoType">
       <xs:restriction base="xs:NMTOKEN">
         <xs:enumeration value="yes"/>
         <xs:enumeration value="no"/>
       </xs:restriction>
     </xs:simpleType>
     <xs:simpleType name="StatusType">
       <xs:union>
         <xs:simpleType>
           <xs:restriction base="xs:NMTOKEN">
             <xs:enumeration value="busy"/>
             <xs:enumeration value="notfound"/>
             <xs:enumeration value="reject"/>
             <xs:enumeration value="error"/>
           </xs:restriction>
         </xs:simpleType>
         <xs:simpleType>
           <xs:restriction base="xs:string"/>
         </xs:simpleType>
       </xs:union>
     </xs:simpleType>
     <xs:complexType name="IncomingType">
       <xs:complexContent>
         <xs:extension base="TriggerType"/>
       </xs:complexContent>
     </xs:complexType>
     <xs:element name="incoming" type="IncomingType"
substitutionGroup="trigger"/>
     <xs:complexType name="TimerType">
       <xs:complexContent>
         <xs:extension base="TriggerType">
           <xs:attribute name="dtstart" type="xs:string" use="required">
             <xs:annotation>
               <xs:documentation>RFC 2445 DATE-TIME</xs:documentation>
             </xs:annotation>
           </xs:attribute>
           <xs:attribute name="dtend" type="xs:string" use="optional">
             <xs:annotation>
               <xs:documentation>RFC 2445 DATE-TIME</xs:documentation>
             </xs:annotation>
```

```
          </xs:attribute>
          <xs:attribute name="duration" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>RFC 2445 DURATION</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="freq" type="FreqType" use="optional"/>
          <xs:attribute name="interval" type="xs:positiveInteger" default="1"/
>
          <xs:attribute name="until" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>RFC 2445 DATE-TIME</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="count" type="xs:positiveInteger" use="optional"/
>
          <xs:attribute name="bysecond" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Comma-separated list of seconds within a
minute. Valid values are 0 to 59.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="byminute" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Comma-separated list of minutes within an
hour. Valid values are 0 to 59.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="byhour" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Comma-separated list of hours of the day.
Valid values are 0 to 23.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="byday" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Comma-separated list of days of the week.
Valid values are "MO", "TU", "WE", "TH", "FR", "SA" and "SU". These values are
not case-sensitive. Each can be preceded by a positive (+n) or negative (-n)
integer.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="bymonthday" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Comma-separated list of days of the month.
Valid values are 1 to 31 or -31 to -1.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="byyearday" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Comma-separated list of days of the year.
Valid values are 1 to 366 or -366 to -1.</xs:documentation>
            </xs:annotation>
```

```
          </xs:attribute>
          <xs:attribute name="byweekno" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Comma-separated list of ordinals specifying
weeks of the year. Valid values are 1 to 53 or -53 to -1.</xs:documentation>
            </xs:annotation>
```

```
          </xs:attribute>
          <xs:attribute name="bymonth" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Comma-separated list of months of the year.
Valid values are 1 to 12.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="wkst" type="DayType" default="MO"/>
          <xs:attribute name="bysetpos" type="YearDayType"/>
          <xs:anyAttribute namespace="##any" processContents="lax"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="timer" type="TimerType" substitutionGroup="trigger"/>
    <xs:simpleType name="OrderingType">
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="parallel"/>
        <xs:enumeration value="sequential"/>
        <xs:enumeration value="first-only"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="AddressFieldType">
      <xs:union>
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="origin"/>
            <xs:enumeration value="destination"/>
            <xs:enumeration value="original-destination"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:union>
    </xs:simpleType>
    <xs:simpleType name="AddressSubfieldType">
      <xs:union>
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="address-type"/>
            <xs:enumeration value="user"/>
            <xs:enumeration value="host"/>
            <xs:enumeration value="port"/>
            <xs:enumeration value="tel"/>
            <xs:enumeration value="display"/>
            <xs:enumeration value="password"/>
            <xs:enumeration value="alias-type"/>
          </xs:restriction>
        </xs:simpleType>
```

```
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:union>
    </xs:simpleType>
    <xs:complexType name="AddressType">
      <xs:annotation>
        <xs:documentation>Exactly one of the three attributes must appear</
xs:documentation>
      </xs:annotation>
      <xs:group ref="Node"/>
      <xs:attribute name="is" type="xs:string" use="optional"/>
      <xs:attribute name="contains" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>for "display" only</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="subdomain-of" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>for "host", "tel" only</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="AddressSwitchType">
      <xs:complexContent>
        <xs:extension base="SwitchType">
          <xs:sequence>
            <xs:element name="address" type="AddressType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:sequence minOccurs="0">
              <xs:element name="not-present" type="NotPresentAction"/>
              <xs:element name="address" type="AddressType" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:element name="otherwise" type="OtherwiseAction" minOccurs="0"/
>
          </xs:sequence>
          <xs:attribute name="field" type="AddressFieldType" use="required"/>
          <xs:attribute name="subfield" type="AddressSubfieldType"
use="optional"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="address-switch" type="AddressSwitchType"
substitutionGroup="switch"/>
    <xs:simpleType name="StringFieldType">
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="subject"/>
        <xs:enumeration value="organization"/>
        <xs:enumeration value="user-agent"/>
        <xs:enumeration value="display"/>
      </xs:restriction>
```

```
    </xs:simpleType>
```

```
     <xs:complexType name="StringType">
       <xs:group ref="Node"/>
       <xs:attribute name="is" type="xs:string" use="optional"/>
       <xs:attribute name="contains" type="xs:string" use="optional"/>
       <xs:anyAttribute namespace="##any" processContents="lax"/>
     </xs:complexType>
     <xs:complexType name="StringSwitchType">
       <xs:complexContent>
         <xs:extension base="SwitchType">
           <xs:sequence>
             <xs:element name="string" type="StringType" minOccurs="0"
maxOccurs="unbounded"/>
             <xs:sequence minOccurs="0">
               <xs:element name="not-present" type="NotPresentAction"/>
               <xs:element name="string" type="StringType" minOccurs="0"
maxOccurs="unbounded"/>
             </xs:sequence>
             <xs:element name="otherwise" type="OtherwiseAction" minOccurs="0"/
>
           </xs:sequence>
           <xs:attribute name="field" type="StringFieldType" use="required">
             <xs:annotation>
               <xs:documentation>Strings are matched as case-insensitive
Unicode strings.</xs:documentation>
             </xs:annotation>
           </xs:attribute>
         </xs:extension>
       </xs:complexContent>
     </xs:complexType>
     <xs:element name="string-switch" type="StringSwitchType"
substitutionGroup="switch"/>
     <xs:complexType name="LanguageType">
       <xs:group ref="Node"/>
       <xs:attribute name="matches" type="xs:string" use="required">
         <xs:annotation>
           <xs:documentation>The value of one of these parameters is a
language-tag, as defined in RFC 3066.</xs:documentation>
         </xs:annotation>
       </xs:attribute>
       <xs:anyAttribute namespace="##any" processContents="lax"/>
     </xs:complexType>
     <xs:complexType name="LanguageSwitchType">
       <xs:complexContent>
         <xs:extension base="SwitchType">
           <xs:sequence>
             <xs:element name="language" type="LanguageType" minOccurs="0"
maxOccurs="unbounded"/>
             <xs:sequence minOccurs="0">
               <xs:element name="not-present" type="NotPresentAction"/>
               <xs:element name="language" type="LanguageType" minOccurs="0"
maxOccurs="unbounded"/>
             </xs:sequence>
             <xs:element name="otherwise" type="OtherwiseAction" minOccurs="0"/
```

```
>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
```

```
     </xs:complexType>
     <xs:element name="language-switch" type="LanguageSwitchType"
substitutionGroup="switch"/>
     <xs:simpleType name="FreqType">
       <xs:restriction base="xs:NMTOKEN">
         <xs:pattern value="[s|S][e|E][c|C][o|O][n|N][d|D][l|L][y|Y]"/>
         <xs:pattern value="[m|M][i|I][n|N][u|U][t|T][e|E][l|L][y|Y]"/>
         <xs:pattern value="[h|H][o|O][u|U][r|R][l|L][y|Y]"/>
         <xs:pattern value="[d|D][a|A][i|I][l|L][y|Y]"/>
         <xs:pattern value="[w|W][e|E][e|E][k|K][l|L][y|Y]"/>
         <xs:pattern value="[m|M][o|N][n|N][t|T][h|H][l|L][y|Y]"/>
         <xs:pattern value="[y|Y][e|E][a|A][r|R][l|L][y|Y]"/>
       </xs:restriction>
     </xs:simpleType>
     <xs:simpleType name="YearDayType">
       <xs:union>
         <xs:simpleType>
           <xs:restriction base="xs:integer">
             <xs:minInclusive value="-366"/>
             <xs:maxInclusive value="-1"/>
           </xs:restriction>
         </xs:simpleType>
         <xs:simpleType>
           <xs:restriction base="xs:integer">
             <xs:minInclusive value="1"/>
             <xs:maxExclusive value="366"/>
           </xs:restriction>
         </xs:simpleType>
       </xs:union>
     </xs:simpleType>
     <xs:simpleType name="DayType">
       <xs:restriction base="xs:NMTOKEN">
         <xs:pattern value="[m|M][o|O]"/>
         <xs:pattern value="[t|T][u|U]"/>
         <xs:pattern value="[w|W][e|E]"/>
         <xs:pattern value="[t|T][h|H]"/>
         <xs:pattern value="[f|F][r|R]"/>
         <xs:pattern value="[s|S][a|A]"/>
         <xs:pattern value="[s|S][u|U]"/>
       </xs:restriction>
     </xs:simpleType>
     <xs:complexType name="TimeType">
       <xs:annotation>
         <xs:documentation>Exactly one of the two attributes "dtend" and
"duration" must occur. None of the attributes following freq are meaningful
unless freq appears. </xs:documentation>
       </xs:annotation>
       <xs:group ref="Node"/>
       <xs:attribute name="dtstart" type="xs:string" use="required">
         <xs:annotation>
           <xs:documentation>RFC 2445 DATE-TIME</xs:documentation>
```

```
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="dtend" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>RFC 2445 DATE-TIME</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="duration" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>RFC 2445 DURATION</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="freq" type="FreqType" use="optional"/>
      <xs:attribute name="interval" type="xs:positiveInteger" default="1"/>
      <xs:attribute name="until" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>RFC 2445 DATE-TIME</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="count" type="xs:positiveInteger" use="optional"/>
      <xs:attribute name="bysecond" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>Comma-separated list of seconds within a minute.
Valid values are 0 to 59.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="byminute" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>Comma-separated list of minutes within an hour.
Valid values are 0 to 59.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="byhour" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>Comma-separated list of hours of the day. Valid
values are 0 to 23.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="byday" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>Comma-separated list of days of the week. Valid
values are "MO", "TU", "WE", "TH", "FR", "SA" and "SU". These values are not
case-sensitive. Each can be preceded by a positive (+n) or negative (-n)
integer.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="bymonthday" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>Comma-separated list of days of the month. Valid
values are 1 to 31 or -31 to -1.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="byyearday" type="xs:string" use="optional">
```

```
         <xs:annotation>
           <xs:documentation>Comma-separated list of days of the year. Valid
values are 1 to 366 or -366 to -1.</xs:documentation>
```

```
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="byweekno" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>Comma-separated list of ordinals specifying weeks
of the year. Valid values are 1 to 53 or -53 to -1.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="bymonth" type="xs:string" use="optional">
        <xs:annotation>
          <xs:documentation>Comma-separated list of months of the year. Valid
values are 1 to 12.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="wkst" type="DayType" default="MO"/>
      <xs:attribute name="bysetpos" type="YearDayType"/>
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>
    <xs:simpleType name="TZIDType">
      <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="TZURLType">
      <xs:restriction base="xs:anyURI"/>
    </xs:simpleType>
    <xs:complexType name="TimeSwitchType">
      <xs:complexContent>
        <xs:extension base="SwitchType">
          <xs:sequence>
            <xs:element name="time" type="TimeType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:sequence minOccurs="0">
              <xs:element name="not-present" type="NotPresentAction"/>
              <xs:element name="time" type="TimeType" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:element name="otherwise" type="OtherwiseAction" minOccurs="0"/
>
          </xs:sequence>
          <xs:attribute name="tzid" type="TZIDType"/>
          <xs:attribute name="tzurl" type="TZURLType"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="time-switch" type="TimeSwitchType"
substitutionGroup="switch"/>
    <xs:simpleType name="PriorityValues">
      <xs:restriction base="xs:NMTOKEN">
        <xs:pattern value="[e|E][m|M][e|E][r|R][g|G][e|E][n|N][c|C][y|Y]"/>
        <xs:pattern value="[u|U][r|R][g|G][e|E][n|N][t|T]"/>
        <xs:pattern value="[n|N][o|O][r|R][m|M][a|A][l|L]"/>
        <xs:pattern value="[n|N][o|O][n|N]-[u|U][r|R][g|G][e|E][n|N][t|T]"/>
      </xs:restriction>
    </xs:simpleType>
```

```
     <xs:complexType name="PriorityType">
```

```
       <xs:annotation>
         <xs:documentation>Exactly one of the three attributes must appear </
xs:documentation>
       </xs:annotation>
       <xs:group ref="Node"/>
       <xs:attribute name="less" type="PriorityValues"/>
       <xs:attribute name="greater" type="PriorityValues"/>
       <xs:attribute name="equal" type="xs:string">
         <xs:annotation>
           <xs:documentation>case-insensitive</xs:documentation>
         </xs:annotation>
       </xs:attribute>
       <xs:anyAttribute namespace="##any" processContents="lax"/>
     </xs:complexType>
     <xs:complexType name="PrioritySwitchType">
       <xs:complexContent>
         <xs:extension base="SwitchType">
           <xs:sequence>
             <xs:element name="priority" type="PriorityType" minOccurs="0"
maxOccurs="unbounded"/>
             <xs:sequence minOccurs="0">
               <xs:element name="not-present" type="NotPresentAction"/>
               <xs:element name="priority" type="PriorityType" minOccurs="0"
maxOccurs="unbounded"/>
             </xs:sequence>
             <xs:element name="otherwise" type="OtherwiseAction" minOccurs="0"/
>
           </xs:sequence>
         </xs:extension>
       </xs:complexContent>
     </xs:complexType>
     <xs:element name="priority-switch" type="PrioritySwitchType"
substitutionGroup="switch"/>
     <xs:complexType name="UserStatusType">
       <xs:group ref="Node"/>
       <xs:attribute name="less" type="xs:integer"/>
       <xs:attribute name="greater" type="xs:integer"/>
       <xs:attribute name="equal" type="xs:integer"/>
       <xs:attribute name="is">
         <xs:simpleType>
           <xs:list itemType="xs:string"/>
         </xs:simpleType>
       </xs:attribute>
       <xs:attribute name="contains">
         <xs:simpleType>
           <xs:list itemType="xs:string"/>
         </xs:simpleType>
       </xs:attribute>
     </xs:complexType>
     <xs:complexType name="StatusSwitchType">
       <xs:complexContent>
         <xs:extension base="SwitchType">
           <xs:sequence>
```

```
            <xs:element name="status" type="UserStatusType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:sequence minOccurs="0">
              <xs:element name="not-present" type="NotPresentAction"/>
              <xs:element name="status" type="UserStatusType" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:element name="otherwise" type="OtherwiseAction" minOccurs="0"/
>
          </xs:sequence>
          <xs:attribute name="uri" type="xs:anyURI" use="optional"/>
          <xs:attribute name="status-name">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="active-calls"/>
                <xs:enumeration value="presence"/>
                <xs:enumeration value="activity"/>
                <xs:enumeration value="mood"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="status-switch" type="StatusSwitchType"
substitutionGroup="switch"/>
    <xs:simpleType name="LocationPriorityType">
      <xs:restriction base="xs:float">
        <xs:minInclusive value="0.0"/>
        <xs:maxInclusive value="1.0"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="LocationType">
      <xs:complexContent>
        <xs:extension base="ModifierType">
          <xs:group ref="Node"/>
          <xs:attribute name="url" type="xs:anyURI" use="required"/>
          <xs:attribute name="priority" type="LocationPriorityType"
use="optional" default="1.0"/>
          <xs:attribute name="clear" type="YesNoType" default="no"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="location" type="LocationType"
substitutionGroup="modifier"/>
    <xs:complexType name="LookupType">
      <xs:complexContent>
        <xs:extension base="ModifierType">
          <xs:all>
            <xs:element name="success" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
```

```
            </xs:element>
```

```
            <xs:element name="notfound" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="failure" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
          </xs:all>
          <xs:attribute name="source" type="xs:string" use="required"/>
          <xs:attribute name="order" use="optional" default="parallel">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="parallel"/>
                <xs:enumeration value="sequential"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="timeout" type="xs:positiveInteger" default="30"/
>
          <xs:attribute name="clear" type="YesNoType" default="no"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="lookup" type="LookupType" substitutionGroup="modifier"/>
    <xs:complexType name="RemoveLocationType">
      <xs:complexContent>
        <xs:extension base="ModifierType">
          <xs:group ref="Node"/>
          <xs:attribute name="location" type="xs:string" use="optional"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="remove-location" type="RemoveLocationType"
substitutionGroup="modifier"/>
    <xs:complexType name="LogAction">
      <xs:complexContent>
        <xs:extension base="ActionType">
          <xs:group ref="Node"/>
          <xs:attribute name="name" type="xs:string" use="optional"/>
          <xs:attribute name="comment" type="xs:string" use="optional"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="log" type="LogAction" substitutionGroup="action"/>
    <xs:complexType name="MailAction">
      <xs:complexContent>
        <xs:extension base="ActionType">
```

```
          <xs:group ref="Node"/>
          <xs:attribute name="url" type="xs:anyURI" use="required"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="mail" type="MailAction" substitutionGroup="action"/>
    <xs:complexType name="WaitAction">
      <xs:complexContent>
        <xs:extension base="ActionType">
          <xs:group ref="Node"/>
          <xs:attribute name="duration" type="xs:duration" use="required"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="wait" type="WaitAction" substitutionGroup="action"/>
    <xs:complexType name="AcceptActionType">
      <xs:complexContent>
        <xs:extension base="ActionType"/>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="accept" type="AcceptActionType"
substitutionGroup="action"/>
    <xs:complexType name="RedirectAction">
      <xs:complexContent>
        <xs:extension base="ActionType">
          <xs:attribute name="permanent" type="YesNoType" default="no"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="redirect" type="RedirectAction"
substitutionGroup="action"/>
    <xs:complexType name="RejectAction">
      <xs:complexContent>
        <xs:extension base="ActionType">
          <xs:attribute name="status" type="StatusType" use="required"/>
          <xs:attribute name="reason" type="xs:string" use="optional"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="reject" type="RejectAction" substitutionGroup="action"/>
    <xs:complexType name="AuthenticateAction">
      <xs:complexContent>
        <xs:extension base="ActionType">
          <xs:all>
            <xs:element name="success" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="failure" minOccurs="0">
```

```
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
          </xs:all>
          <xs:attribute name="method" use="optional" default="digest">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="digest"/>
                <xs:enumeration value="basic"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="realm" type="xs:string"/>
          <xs:attribute name="user" type="xs:string"/>
          <xs:attribute name="target">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="remote"/>
                <xs:enumeration value="local"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="authenticate" type="AuthenticateAction"
substitutionGroup="action"/>
    <xs:complexType name="CallActionType">
      <xs:complexContent>
        <xs:extension base="ActionType">
          <xs:all>
            <xs:element name="accepted" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="busy" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="noanswer" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="failure" minOccurs="0">
              <xs:complexType>
```

```
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="redirection" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
          </xs:all>
          <xs:attribute name="timeout" type="xs:positiveInteger"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="call" type="CallActionType" substitutionGroup="action"/>
    <xs:complexType name="TerminateActionType">
      <xs:complexContent>
        <xs:extension base="ActionType">
          <xs:group ref="Node"/>
          <xs:attribute name="uri">
            <xs:simpleType>
              <xs:list itemType="xs:anyURI"/>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="subject">
            <xs:simpleType>
              <xs:list itemType="xs:string"/>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="calls">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="all"/>
                <xs:enumeration value="last"/>
                <xs:enumeration value="this"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="terminate" type="TerminateActionType"
substitutionGroup="action"/>
  </xs:schema>
```

19.3 XML Schema for LESS Mid-call handling Extension

```
   <?xml version="1.0" encoding="UTF-8"?>
   <!-- edited with XMLSPY v2004 rel. 2 U (http://www.xmlspy.com) by Xiaotao Wu
(Columbia University Computer Science Dept) -->
   <!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Xiaotao Wu
(Columbia University) -->
   <xs:schema targetNamespace="urn:ietf:params:xml:ns:less:midcall"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://
www.w3.org/2001/XMLSchema" xmlns:LESS="urn:ietf:params:xml:ns:less"
xmlns="urn:ietf:params:xml:ns:less:midcall">
     <xs:import namespace="urn:ietf:params:xml:ns:less"
schemaLocation="less.xsd"/>
     <xs:complexType name="TransferActionType">
       <xs:complexContent>
         <xs:extension base="LESS:ActionType">
           <xs:all>
             <xs:element name="accepted" minOccurs="0">
               <xs:complexType>
                 <xs:group ref="Node"/>
               </xs:complexType>
             </xs:element>
             <xs:element name="busy" minOccurs="0">
               <xs:complexType>
                 <xs:group ref="Node"/>
               </xs:complexType>
             </xs:element>
             <xs:element name="noanswer" minOccurs="0">
               <xs:complexType>
                 <xs:group ref="Node"/>
               </xs:complexType>
             </xs:element>
             <xs:element name="failure" minOccurs="0">
               <xs:complexType>
                 <xs:group ref="Node"/>
               </xs:complexType>
             </xs:element>
             <xs:element name="redirection" minOccurs="0">
               <xs:complexType>
                 <xs:group ref="Node"/>
               </xs:complexType>
             </xs:element>
           </xs:all>
           <xs:attribute name="timeout" type="xs:positiveInteger"
use="optional" default="20"/>
         </xs:extension>
       </xs:complexContent>
     </xs:complexType>
     <xs:element name="transfer" type="TransferActionType"
substitutionGroup="LESS:action"/>
     <xs:complexType name="MergeAction">
       <xs:complexContent>
         <xs:extension base="LESS:ActionType">
           <xs:attribute name="uri">
```

```
            <xs:simpleType>
              <xs:list itemType="xs:anyURI"/>
            </xs:simpleType>
```

```
          </xs:attribute>
          <xs:attribute name="subject">
            <xs:simpleType>
              <xs:list itemType="xs:string"/>
            </xs:simpleType>
          </xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="merge" type="MergeAction"
substitutionGroup="LESS:action"/>
  </xs:schema>
```

19.4 XML Schema for LESS User Interaction Extension

```
   <?xml version="1.0" encoding="UTF-8"?>
   <!-- edited with XMLSPY v2004 rel. 2 U (http://www.xmlspy.com) by Xiaotao Wu
(Columbia University Computer Science Dept) -->
   <!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Xiaotao Wu
(Columbia University) -->
   <xs:schema targetNamespace="urn:ietf:params:xml:ns:less:ui"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://
www.w3.org/2001/XMLSchema" xmlns:LESS="urn:ietf:params:xml:ns:less"
xmlns="urn:ietf:params:xml:ns:less:ui">
     <xs:import namespace="urn:ietf:params:xml:ns:less"
schemaLocation="less.xsd"/>
     <xs:complexType name="CommandType">
       <xs:complexContent>
         <xs:extension base="LESS:TriggerType">
           <xs:attribute name="command" type="xs:string" use="required"/>
         </xs:extension>
       </xs:complexContent>
     </xs:complexType>
     <xs:element name="command" type="CommandType"
substitutionGroup="LESS:trigger"/>
     <xs:complexType name="AlertActionType">
       <xs:complexContent>
         <xs:extension base="LESS:ActionType">
           <xs:all>
             <xs:element name="timeout" minOccurs="0">
               <xs:complexType>
                 <xs:group ref="Node"/>
               </xs:complexType>
             </xs:element>
           </xs:all>
           <xs:attribute name="duration" type="xs:dateTime" use="optional"/>
           <xs:attribute name="priority" type="LESS:PriorityValues"
use="optional"/>
           <xs:attribute name="message" type="xs:string" use="optional"/>
           <xs:attribute name="icon" type="xs:string" use="optional"/>
           <xs:attribute name="input" type="xs:anyURI" use="optional"/>
           <xs:attribute name="style" use="optional" default="sound">
             <xs:simpleType>
               <xs:list>
                 <xs:simpleType>
                   <xs:restriction base="xs:string">
                     <xs:enumeration value="viberate"/>
                     <xs:enumeration value="flash"/>
                     <xs:enumeration value="sound"/>
                     <xs:enumeration value="text"/>
                   </xs:restriction>
                 </xs:simpleType>
               </xs:list>
             </xs:simpleType>
           </xs:attribute>
         </xs:extension>
       </xs:complexContent>
```

```
     </xs:complexType>
     <xs:element name="alert" type="AlertActionType"
substitutionGroup="LESS:action"/>
     <xs:complexType name="GetinputActionType">
```

```
      <xs:complexContent>
        <xs:extension base="LESS:ActionType">
          <xs:all>
            <xs:element name="noanswer" minOccurs="0">
              <xs:complexType>
                <xs:group ref="Node"/>
              </xs:complexType>
            </xs:element>
          </xs:all>
          <xs:attribute name="timeout" type="xs:duration" use="optional"/>
          <xs:attribute name="source" use="optional">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="screen"/>
                <xs:enumeration value="audio"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="target" use="optional">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="remote"/>
                <xs:enumeration value="local"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="getinput" type="GetinputActionType"
substitutionGroup="LESS:action"/>
  </xs:schema>
```

19.5 XML Schema for LESS Instant Messaging Extension

```
   <?xml version="1.0" encoding="UTF-8"?>
   <!-- edited with XMLSPY v2004 rel. 2 U (http://www.xmlspy.com) by Xiaotao Wu
(Columbia University Computer Science Dept) -->
   <!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Xiaotao Wu
(Columbia University) -->
   <xs:schema targetNamespace="urn:ietf:params:xml:ns:less:im"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://
www.w3.org/2001/XMLSchema" xmlns:LESS="urn:ietf:params:xml:ns:less"
xmlns="urn:ietf:params:xml:ns:less:im">
     <xs:import namespace="urn:ietf:params:xml:ns:less"
schemaLocation="less.xsd"/>
     <xs:complexType name="MessageType">
       <xs:complexContent>
         <xs:extension base="LESS:TriggerType"/>
       </xs:complexContent>
     </xs:complexType>
     <xs:element name="message" type="MessageType"
substitutionGroup="LESS:trigger"/>
     <xs:complexType name="SendmsgActionType">
       <xs:complexContent>
         <xs:extension base="LESS:ActionType">
           <xs:attribute name="type">
             <xs:simpleType>
               <xs:union>
                 <xs:simpleType>
                   <xs:restriction base="xs:NMTOKEN">
                     <xs:enumeration value="dtmf"/>
                     <xs:enumeration value="text"/>
                     <xs:enumeration value="image"/>
                     <xs:enumeration value="file"/>
                   </xs:restriction>
                 </xs:simpleType>
                 <xs:simpleType>
                   <xs:restriction base="xs:string"/>
                 </xs:simpleType>
               </xs:union>
             </xs:simpleType>
           </xs:attribute>
           <xs:attribute name="message" type="xs:string"/>
         </xs:extension>
       </xs:complexContent>
     </xs:complexType>
     <xs:element name="sendmsg" type="SendmsgActionType"
substitutionGroup="LESS:action"/>
   </xs:schema>
```

19.6 XML Schema for LESS Event Handling Extension

```
   <?xml version="1.0" encoding="UTF-8"?>
   <!-- edited with XMLSPY v2004 rel. 2 U (http://www.xmlspy.com) by Xiaotao Wu
(Columbia University Computer Science Dept) -->
   <xs:schema targetNamespace="urn:ietf:params:xml:ns:less:event"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://
www.w3.org/2001/XMLSchema" xmlns:LESS="urn:ietf:params:xml:ns:less"
xmlns="urn:ietf:params:xml:ns:less:event" elementFormDefault="qualified"
attributeFormDefault="unqualified">
     <xs:import namespace="urn:ietf:params:xml:ns:less"
schemaLocation="less.xsd"/>
     <xs:element name="subscription" type="SubscriptionType"
substitutionGroup="LESS:trigger"/>
     <xs:complexType name="SubscriptionType">
       <xs:complexContent>
         <xs:extension base="LESS:TriggerType"/>
       </xs:complexContent>
     </xs:complexType>
     <xs:element name="notification" type="NotificationType"
substitutionGroup="LESS:trigger"/>
     <xs:complexType name="NotificationType">
       <xs:complexContent>
         <xs:extension base="LESS:TriggerType"/>
       </xs:complexContent>
     </xs:complexType>
     <xs:complexType name="EventType">
       <xs:group ref="LESS:Node"/>
       <xs:attribute name="package" use="required">
         <xs:simpleType>
           <xs:list itemType="xs:string"/>
         </xs:simpleType>
       </xs:attribute>
       <xs:attribute name="is" use="optional">
         <xs:simpleType>
           <xs:list itemType="xs:string"/>
         </xs:simpleType>
       </xs:attribute>
     </xs:complexType>
     <xs:complexType name="EventSwitchType">
       <xs:sequence>
         <xs:element name="event" type="EventType" minOccurs="0"
maxOccurs="unbounded"/>
         <xs:sequence minOccurs="0">
           <xs:element name="not-present" type="LESS:NotPresentAction"/>
           <xs:element name="event" type="EventType" minOccurs="0"
maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:element name="otherwise" type="LESS:OtherwiseAction"
minOccurs="0"/>
       </xs:sequence>
     </xs:complexType>
     <xs:element name="event-switch" type="EventSwitchType"
substitutionGroup="LESS:switch"/>
```

```
    <xs:complexType name="ApproveActionType">
      <xs:complexContent>
        <xs:extension base="LESS:ActionType">
          <xs:attribute name="expires" type="xs:integer">
            <xs:annotation>
              <xs:documentation>Expiration in seconds</xs:documentation>
            </xs:annotation>
```

```
          </xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="approve" type="ApproveActionType"
substitutionGroup="LESS:action"/>
    <xs:complexType name="DenyActionType">
      <xs:complexContent>
        <xs:extension base="LESS:ActionType">
          <xs:attribute name="reason" type="xs:string" use="optional"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="deny" type="DenyActionType"
substitutionGroup="LESS:action"/>
    <xs:complexType name="DeferActionType">
      <xs:complexContent>
        <xs:extension base="LESS:ActionType"/>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="defer" type="DeferActionType"
substitutionGroup="LESS:action"/>
    <xs:complexType name="SubscribeActionType">
      <xs:complexContent>
        <xs:extension base="LESS:ActionType">
          <xs:all>
            <xs:element name="approved" minOccurs="0">
              <xs:complexType>
                <xs:group ref="LESS:Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="denied" minOccurs="0">
              <xs:complexType>
                <xs:group ref="LESS:Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="pending" minOccurs="0">
              <xs:complexType>
                <xs:group ref="LESS:Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="redirection" minOccurs="0">
              <xs:complexType>
                <xs:group ref="LESS:Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="noanswer" minOccurs="0">
              <xs:complexType>
                <xs:group ref="LESS:Node"/>
              </xs:complexType>
            </xs:element>
```

```
            <xs:element name="failure" minOccurs="0">
              <xs:complexType>
                <xs:group ref="LESS:Node"/>
              </xs:complexType>
            </xs:element>
          </xs:all>
          <xs:attribute name="timeout" type="xs:positiveInteger"
use="optional" default="20"/>
          <xs:attribute name="expires" type="xs:integer">
            <xs:annotation>
              <xs:documentation>Expiration in seconds</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="package" type="xs:string" use="required"/>
          <xs:attribute name="content" type="xs:string" use="optional"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="subscribe" type="SubscribeActionType"
substitutionGroup="LESS:action"/>
    <xs:complexType name="NotifyActionType">
      <xs:complexContent>
        <xs:extension base="LESS:ActionType">
          <xs:all>
            <xs:element name="accepted" minOccurs="0">
              <xs:complexType>
                <xs:group ref="LESS:Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="redirection" minOccurs="0">
              <xs:complexType>
                <xs:group ref="LESS:Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="noanswer" minOccurs="0">
              <xs:complexType>
                <xs:group ref="LESS:Node"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="failure" minOccurs="0">
              <xs:complexType>
                <xs:group ref="LESS:Node"/>
              </xs:complexType>
            </xs:element>
          </xs:all>
          <xs:attribute name="timeout" type="xs:positiveInteger"
use="optional" default="20"/>
          <xs:attribute name="package" type="xs:string" use="required"/>
          <xs:attribute name="event" type="xs:string" use="required"/>
          <xs:anyAttribute namespace="##any" processContents="lax"/>
        </xs:extension>
```

```
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="notify" type="NotifyActionType"
substitutionGroup="LESS:action"/>
  </xs:schema>
```

19.7 XML Schema for LESS Location-based Services Extension

```
   <?xml version="1.0" encoding="UTF-8"?>
   <!-- edited with XMLSPY v2004 rel. 2 U (http://www.xmlspy.com) by Xiaotao Wu
(Columbia University Computer Science Dept) -->
   <xs:schema targetNamespace="urn:ietf:params:xml:ns:less:location"
xmlns="urn:ietf:params:xml:ns:less:location" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:LESS="urn:ietf:params:xml:ns:less" elementFormDefault="qualified"
attributeFormDefault="unqualified">
     <xs:import namespace="urn:ietf:params:xml:ns:less"
schemaLocation="less.xsd"/>
     <xs:complexType name="WhereSwitchType">
       <xs:sequence>
         <xs:element name="where" type="WhereType" minOccurs="0"
maxOccurs="unbounded"/>
         <xs:sequence minOccurs="0">
           <xs:element name="not-present" type="LESS:NotPresentAction"/>
           <xs:element name="where" type="WhereType" minOccurs="0"
maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:element name="otherwise" type="LESS:OtherwiseAction"
minOccurs="0"/>
       </xs:sequence>
       <xs:attribute name="type" use="required">
         <xs:simpleType>
           <xs:restriction base="xs:string">
             <xs:enumeration value="geospatial"/>
             <xs:enumeration value="civil"/>
           </xs:restriction>
         </xs:simpleType>
       </xs:attribute>
       <xs:attribute name="uri" type="xs:anyURI"/>
     </xs:complexType>
     <xs:element name="where-switch" type="WhereSwitchType"
substitutionGroup="LESS:switch"/>
     <xs:complexType name="WhereType">
       <xs:group ref="LESS:Node"/>
       <xs:attribute name="changed" type="xs:boolean" default="true"/>
       <xs:attribute name="longitude" type="xs:string">
         <xs:annotation>
           <xs:documentation>Valid only if type is geospatial</
xs:documentation>
         </xs:annotation>
       </xs:attribute>
       <xs:attribute name="latitude" type="xs:string">
         <xs:annotation>
           <xs:documentation>Valid only if type is geospatial</
xs:documentation>
         </xs:annotation>
       </xs:attribute>
       <xs:attribute name="altitude" type="xs:string">
         <xs:annotation>
           <xs:documentation>Valid only if type is geospatial</
```

```
xs:documentation>
         </xs:annotation>
       </xs:attribute>
       <xs:attribute name="A1" type="xs:string">
         <xs:annotation>
           <xs:documentation>Valid only if type is civil</xs:documentation>
         </xs:annotation>
       </xs:attribute>
```

```
      <xs:attribute name="A2" type="xs:string">
        <xs:annotation>
          <xs:documentation>Valid only if type is civil</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="A3" type="xs:string">
        <xs:annotation>
          <xs:documentation>Valid only if type is civil</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="A4" type="xs:string">
        <xs:annotation>
          <xs:documentation>Valid only if type is civil</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="A5" type="xs:string">
        <xs:annotation>
          <xs:documentation>Valid only if type is civil</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="A6" type="xs:string">
        <xs:annotation>
          <xs:documentation>Valid only if type is civil</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="PRD" type="xs:string">
        <xs:annotation>
          <xs:documentation>Valid only if type is civil</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="POD" type="xs:string">
        <xs:annotation>
          <xs:documentation>Valid only if type is civil</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="STS" type="xs:string">
        <xs:annotation>
          <xs:documentation>Valid only if type is civil</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="HNO" type="xs:string">
        <xs:annotation>
          <xs:documentation>Valid only if type is civil</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="HNS" type="xs:string">
        <xs:annotation>
          <xs:documentation>Valid only if type is civil</xs:documentation>
```

```
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="LMK" type="xs:string">
        <xs:annotation>
          <xs:documentation>Valid only if type is civil</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="LOC" type="xs:string">
        <xs:annotation>
          <xs:documentation>Valid only if type is civil</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="FLR" type="xs:string">
        <xs:annotation>
          <xs:documentation>Valid only if type is civil</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="NAM" type="xs:string">
        <xs:annotation>
          <xs:documentation>Valid only if type is civil</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="PC" type="xs:string">
        <xs:annotation>
          <xs:documentation>Valid only if type is civil</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="distance">
        <xs:annotation>
          <xs:documentation>Valid only if geospatial or civil location
presented</xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:decimal">
            <xs:minInclusive value="0"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="condition" default="at">
        <xs:annotation>
          <xs:documentation>Valid only if distance presented</
xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="in"/>
            <xs:enumeration value="out"/>
            <xs:enumeration value="at"/>
          </xs:restriction>
        </xs:simpleType>
```

```
      </xs:attribute>
      <xs:attribute name="unit" default="m">
        <xs:annotation>
          <xs:documentation>Valid only if distance presented</
xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="m"/>
            <xs:enumeration value="km"/>
            <xs:enumeration value="mi"/>
            <xs:enumeration value="in"/>
            <xs:enumeration value="ft"/>
            <xs:enumeration value="yd"/>
            <xs:enumeration value="naut mi"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="direction">
        <xs:annotation>
          <xs:documentation>Valid only if geospatial or civil location
presented</xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="south"/>
            <xs:enumeration value="north"/>
            <xs:enumeration value="east"/>
            <xs:enumeration value="west"/>
            <xs:enumeration value="above"/>
            <xs:enumeration value="below"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="WhereRelationSwitchType">
      <xs:sequence>
        <xs:element name="where-relation" type="WhereRelationType"
minOccurs="0" maxOccurs="unbounded"/>
        <xs:sequence minOccurs="0">
          <xs:element name="not-present" type="LESS:NotPresentAction"/>
          <xs:element name="where-relation" type="WhereRelationType"
minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:element name="otherwise" type="LESS:OtherwiseAction"
minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="uri1" type="xs:anyURI" use="required"/>
      <xs:attribute name="uri2" type="xs:anyURI"/>
    </xs:complexType>
    <xs:element name="where-relation-switch" type="WhereRelationSwitchType"
substitutionGroup="LESS:switch"/>
```

```
     <xs:simpleType name="WhereAttributes">
```

```
       <xs:restriction base="xs:string">
         <xs:enumeration value="latitude"/>
         <xs:enumeration value="longitude"/>
         <xs:enumeration value="altitude"/>
         <xs:enumeration value="country"/>
         <xs:enumeration value="A1"/>
         <xs:enumeration value="A2"/>
         <xs:enumeration value="A3"/>
         <xs:enumeration value="A4"/>
         <xs:enumeration value="A5"/>
         <xs:enumeration value="A6"/>
         <xs:enumeration value="PRD"/>
         <xs:enumeration value="POD"/>
         <xs:enumeration value="STS"/>
         <xs:enumeration value="HNO"/>
         <xs:enumeration value="HNS"/>
         <xs:enumeration value="LMK"/>
         <xs:enumeration value="LOC"/>
         <xs:enumeration value="FLR"/>
         <xs:enumeration value="NAM"/>
         <xs:enumeration value="PC"/>
       </xs:restriction>
     </xs:simpleType>
     <xs:complexType name="WhereRelationType">
       <xs:group ref="LESS:Node"/>
       <xs:attribute name="distance">
         <xs:annotation>
           <xs:documentation>Valid only if geospatial or civil location
presented</xs:documentation>
         </xs:annotation>
         <xs:simpleType>
           <xs:restriction base="xs:decimal">
             <xs:minInclusive value="0"/>
           </xs:restriction>
         </xs:simpleType>
       </xs:attribute>
       <xs:attribute name="condition" default="at">
         <xs:annotation>
           <xs:documentation>Valid only if distance presented</
xs:documentation>
         </xs:annotation>
         <xs:simpleType>
           <xs:restriction base="xs:string">
             <xs:enumeration value="in"/>
             <xs:enumeration value="out"/>
             <xs:enumeration value="at"/>
           </xs:restriction>
         </xs:simpleType>
       </xs:attribute>
       <xs:attribute name="unit" default="m">
```

```
         <xs:annotation>
           <xs:documentation>Valid only if distance presented</
xs:documentation>
         </xs:annotation>
         <xs:simpleType>
           <xs:restriction base="xs:string">
             <xs:enumeration value="m"/>
             <xs:enumeration value="km"/>
             <xs:enumeration value="mi"/>
             <xs:enumeration value="in"/>
             <xs:enumeration value="ft"/>
             <xs:enumeration value="yd"/>
             <xs:enumeration value="naut mi"/>
           </xs:restriction>
         </xs:simpleType>
       </xs:attribute>
       <xs:attribute name="direction">
         <xs:annotation>
           <xs:documentation>Valid only if geospatial or civil location
presented</xs:documentation>
         </xs:annotation>
         <xs:simpleType>
           <xs:restriction base="xs:string">
             <xs:enumeration value="south"/>
             <xs:enumeration value="north"/>
             <xs:enumeration value="east"/>
             <xs:enumeration value="west"/>
             <xs:enumeration value="above"/>
             <xs:enumeration value="below"/>
           </xs:restriction>
         </xs:simpleType>
       </xs:attribute>
       <xs:attribute name="same">
         <xs:simpleType>
           <xs:list itemType="WhereAttributes"/>
         </xs:simpleType>
       </xs:attribute>
       <xs:attribute name="difference">
         <xs:simpleType>
           <xs:list itemType="WhereAttributes"/>
         </xs:simpleType>
       </xs:attribute>
       <xs:anyAttribute namespace="##other" processContents="lax"/>
     </xs:complexType>
   </xs:schema>
```

19.8 XML Schema for LESS Queue Handling Extension

```
   <?xml version="1.0" encoding="UTF-8"?>
   <!-- edited with XMLSPY v2004 rel. 2 U (http://www.xmlspy.com) by Xiaotao Wu
(Columbia University Computer Science Dept) -->
   <xs:schema targetNamespace="urn:ietf:params:xml:ns:less:queue"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://
www.w3.org/2001/XMLSchema" xmlns:LESS="urn:ietf:params:xml:ns:less"
xmlns="urn:ietf:params:xml:ns:less:queue" elementFormDefault="qualified"
attributeFormDefault="unqualified">
     <xs:import namespace="urn:ietf:params:xml:ns:less"
schemaLocation="less.xsd"/>
     <xs:simpleType name="QueueName">
       <xs:union>
         <xs:simpleType>
           <xs:restriction base="xs:NMTOKEN">
             <xs:enumeration value="callback"/>
             <xs:enumeration value="hold"/>
           </xs:restriction>
         </xs:simpleType>
         <xs:simpleType>
           <xs:restriction base="xs:string"/>
         </xs:simpleType>
       </xs:union>
     </xs:simpleType>
     <xs:complexType name="EnqueueActionType">
       <xs:complexContent>
         <xs:extension base="LESS:ActionType">
           <xs:all>
             <xs:element name="full" minOccurs="0">
               <xs:complexType>
                 <xs:group ref="Node"/>
               </xs:complexType>
             </xs:element>
           </xs:all>
           <xs:attribute name="queue" type="QueueName" use="optional"/>
           <xs:attribute name="maxlen" type="xs:positiveInteger"
use="optional"/>
         </xs:extension>
       </xs:complexContent>
     </xs:complexType>
     <xs:element name="enqueue" type="EnqueueActionType"
substitutionGroup="LESS:action"/>
     <xs:complexType name="DequeueActionType">
       <xs:complexContent>
         <xs:extension base="LESS:ActionType">
           <xs:all>
             <xs:element name="success" minOccurs="0">
               <xs:complexType>
                 <xs:group ref="Node"/>
               </xs:complexType>
             </xs:element>
             <xs:element name="failure" minOccurs="0">
               <xs:complexType>
```

```
            <xs:group ref="Node"/>
          </xs:complexType>
        </xs:element>
```

```
          </xs:all>
          <xs:attribute name="queue" type="QueueName" use="optional"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="dequeue" type="DequeueActionType"
substitutionGroup="LESS:action"/>
  </xs:schema>
```

20 References

20.1 Normative References

[1] S. Bradner, "Key words for use in RFCs to indicate requirement levels," RFC 2119, Internet Engineering Task Force, Mar. 1997.

[2] J. Lennox, X. Wu, and H. Schulzrinne, "Call processing language (CPL): a language for user control of Internet telephony services," rfc, Internet Engineering Task Force, Oct. 2004. Standard.

[3] T. Bray, J. Paoli, and C. M. Sperberg-McQueen, "Extensible markup language (XML) 1.0 (second edition)," W3C Recommendation REC-xml-20001006, World Wide Web Consortium (W3C), Oct. 2000. Available at http://www.w3.org/XML/.

[4] D. C. Fallside, "XML schema part 0: Primer," W3C Candidate Recommendation CR-xmlschema-0-20001024, World Wide Web Consortium (W3C), Oct. 2000. Available at http://www.w3.org/TR/xmlschema-0/.

[5] H. Schulzrinne, "RPID -- rich presence information data format," Internet Draft draft-ietf-simple-rpid-01, Internet Engineering Task Force, Feb. 2004. Work in progress.

[6] R. Moats, "URN syntax," RFC 2141, Internet Engineering Task Force, May 1997.

[7] R. Moats, "A URN namespace for IETF documents," RFC 2648, Internet Engineering Task Force, Aug. 1999.

[8] R. Mahy, "The ietf xml registry," RFC 3688, Internet Engineering Task Force, Jan. 2004.

[9] M. Murata, S. S. Laurent, and D. Kohn, "XML media types," RFC 3023, Internet Engineering Task Force, Jan. 2001.

[10] T. Bray, D. Hollander, and A. Layman, "Namespaces in XML," W3C Recommendation REC-xml-names-19900114, World Wide Web Consortium (W3C), Jan. 1999. Available at http://www.w3.org/TR/REC-xml-names/.

20.2 Informative References

[11] X. Wu and H. Schulzrinne, "Where should services reside in Internet telephony systems?," in IP Telecom Services Workshop, (Atlanta, Georgia), Sept. 2000.

[12] X. Wu and H. Schulzrinne, "Programmable end system services using SIP," in Conference Record of the International Conference on Communications (ICC), May 2003.

[13] R. Stansifer, Study of Programming Languages, vol. 1. Florida Institute of Technology: Prentice Hall, 1994.

[14] A. D. Falkoff and K. E. Iverson, "The design of APL," SIGAPL APL Quote Quad, vol. 6, no. 1, pp. 5--14, 1975.

[15] X. Wu and H. Schulzrinne, "Feature interactions in Internet telephony end systems," tech. rep., Department of Computer Science, Columbia University, Jan. 2004.

[16] International Telecommunication Union, "General recommendations on telephone switching and signaling -- intelligent network: Introduction to intelligent network capability set 1," Recommendation Q.1211, International Telecommunication Union, Geneva, Switzerland, Mar. 1993.

[17] AT&T, 5ESS Switch, The Premier Solution, Feature Handbook, Issue 4. AT&T, Sept. 1987.

[18] E. International, "Services for computer supported telecommunications applications (CSTA) Phase III," Standard 269, Ecma International, June 2004.

[19] J. Lennox, H. Schulzrinne, and T. L. Porta, "Implementing intelligent network services with the session initiation protocol," Technical Report CUCS-002-99, Columbia University, New York, New York, Jan. 1999.

[20] VoiceXML Forum, "Voicexml home page." http://www.voicexml.org/.

[21] R. Mahy, "A message summary and message waiting indication event package for the session initiation protocol (SIP)," RFC 3842, Internet Engineering Task Force, Aug. 2004.

Authors' Addresses

Xiaotao Wu
Dept. of Computer Science

Columbia University
1214 Amsterdam Avenue, MC 0401
New York, NY 10027
USA
electronic mail: xiaotaow@cs.columbia.edu

Henning Schulzrinne
Dept. of Computer Science
Columbia University
1214 Amsterdam Avenue, MC 0401
New York, NY 10027
USA
electronic mail: schulzrinne@cs.columbia.edu

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.