TestML Tags

February 17, 2009

Touchstone Technologies, Inc.
228 North York Road, Suites C & D
Hatboro, PA 19040
Tel: 215-672-6550
Fax: 215-672-6551

www.touchstone-inc.com

Table of Contents

Structured Elements

Tag: blueprint

Attributes:

name - the name of the blueprint variable.
description - brief description of the blueprint that appears on user interface.

Example:

```
<blueprint name="Default" description="Basic SIP UAC & UAS Functionality">

<dictionary name = "system" file="[system.dictionary]" />

<template name="new invite" type="request">
<![CDATA[
  "INVITE"{='
'[last.invite.request.transport]':'}{=':'[last.invite.target.id]'@'}{='@'[last.invite.target.
address]':| '}{?':'[last.invite.target.port]' '}{='
'[last.invite.request.protocol]'/'}{='/'[last.invite.request.version]}

"Via:"{=[last.invite.via.protocol]'/'}{='/'[last.invite.via.version]'/'}{='/'[last.invite.via.pr
otocol]' '}{='
'[last.invite.via.address]':|;|,|'}{?':'[last.invite.via.port]';|,|'}{'^;branch='[last.via.branc
h]':|;|,|'}{=',#repeat}

"From:"{?[last.invite.remote.name]'<'}{='<'[last.invite.sip.transport]':'}{=':'[last.invite
.remote.id]'@'}{='@'[last.invite.remote.address]':|;|,|'}{?':'[last.invite.remote.port]';|'}
{^'tag='[last.invite.remote.tag]';|'}

"To:"{?[last.invite.local.name]'<'}{='<'[last.invite.local.transport]':'}{=':'[last.invite.loc
al.id]'@'}{='@'[last.invite.local.address]':|;|'}{'?:'[last.invite.local.port]"|;|'}
  "Call-ID:"{=[last.invite.callid]}
      "Cseq:"{=[last.invite.remote.cseq]' '}{=' '[last.invite.method]}

"Contact:"{=[last.invite.contact.protocol]':'}{=':'[last.invite.contact.id]'@'}{'@'[last.in
vite.contact.address]':|;|'}{?':'[last.invite.contact.port]':|;|'}
  "Authorization:"{=' '[authorization.method]'
'}{^user="[authorization.username]',|'}{^'realm='[authorization.realm]',|'}{^'nonce='[
authorization.nonce]',|'}{^'uri='[authorization.uri]',|'}{^'response='[authorization.res
ponse]',|'}{^'algorithm='[authorization.algorithm]',|'}
```

```
  "Max-Forwards:"{=[last.invite.maxforwards]}
  "Subject:"{=[last.invite.subject]}
  "Content-Type:"{=[last.invite.content.type]}
  "Content-Length:"{=[last.invite.content.length]}
  {=[last.invite.content.payload]}
]]>
</template>

<transaction name="Register" method="REGISTER">

<message name="register">
<![CDATA[
  REGISTER
[system.sip.transport]:[system.registrar.address]:[system.registrar.port]
[system.sip.protocol]/[system.sip.version]
  Via: [system.sip.protocol]/[system.sip.version]/[system.ip.protocol]
[input.local.address]:[input.local.port];branch=[input.branch]
  From: [input.local.name]
<[system.sip.transport]:[input.local.id]@[system.registrar.address]:[system.registr
ar.port]>;tag=[input.local.tag]
  To: [input.remote.name]
<[system.sip.transport]:[input.remote.id]@[system.registrar.address]:[system.regi
strar.port]>
  Max-Forwards: [transaction.maxforwards]
  Call-ID: [transaction.callid]
  Cseq: [transaction.local.cseq] REGISTER
  Contact:
[system.sip.protocol]:[input.local.id]@[input.local.address]:[input.local.port];expire
s=[session.registration.ttl]
  Expires: [session.registration.ttl]
  <conditional var="session.authorization.length" op="greater" value="0"
type="integer">
  Authorization: [authorization.method] user="[authorization.username]",
realm="[authorization.realm]", nonce="[authorization.nonce]",
uri="[authorization.uri]", response="[authorization.response]",
algorithm=[authorization.algorithm]
  </conditional>

]]>
</message>

<set var="transaction.cseq" value="[registration.cseq]" />
<set var="transaction.callid"        value="abc-[new.randhex.16]-xyz" />
<set var="transaction.branch"       value="[new.randhex.16]" />
```

```
<send message="register" protocol="sip" ip="[system.registrar.address]"
port="[system.registrar.address]" key="" retransmit="[system.sip.t1]">

<error>
<log level="Error" text="A socket error occured sending [transaction.name]
([transaction.method] [transaction.type]) to
[system.registrar.address]:[system.registrar.port]" />
</error>

<success>
<log level="Trace" text="Successfully sent a [transaction.name]
([transaction.method] [transaction.type]) to
[system.registrar.address]:[system.registrar.port]" />
</success>

  </send>
<incr var="registration.cseq" increment="1" />

<log level="all" text="Entering Register WaitEvents" />
<waitevents>
<receive protocol="sip" method="REGISTER" type="response"
code="100..179,180..189,190,191..199" complete="false"/>

<receive protocol="sip" method="REGISTER" type="response" code="200..299"
complete="true">
<set var="session.registered" value="true"/>
<execute type="internal" id="session.registration.StartExpirationTimer" />
</receive>


<receive protocol="sip" method="REGISTER" type="response" code="401"
complete="true">
    <if var="session.authorization.length" op="greater" value="0" type="integer">
<execute type="internal" id="session.Authenticate.MD5"/>
   <if var="errorcode" op="equal" value="0" type="integer">
<execute type="transaction" id="Register"/>
</if>
</if>
</receive >

<receive protocol="sip" method="REGISTER" type="response" code="407"
complete="true">
    <if var="session.authorization.length" op="greater" value="0" type="integer">
<execute type="internal" id="session.ProxyAuthenticate.MD5"/>
   <if var="errorcode" op="equal" value="0" type="integer">
<execute type="transaction" id="Unregister"/>
```

```
</if>
</if>
</receive >

<receive protocol="sip" method="REGISTER" type="response" code="300..399"
complete="true">
<set var="transaction.error" value="true" />
</receive >

<event id="user.stop" complete="true"/>
<event id="user.terminate" complete="true"/>

</waitevents>
<log level="all" text="Exiting Register WaitEvents" />
</transaction>

<transaction name="Unregister" method="REGISTER">

<message name="unregister">
    <![CDATA[

  REGISTER
[system.sip.transport]:[input.remote.id]@[input.remote.address]:[input.remote.por
t] [system.sip.protocol]/[system.sip.version]
  Via: [system.sip.protocol]/[system.sip.version]/[system.ip.protocol]
[input.local.address]:[input.local.port];branch=[input.branch]
  From: [input.local.name]
<[system.sip.transport]:[input.local.id]@[input.local.address]:[input.local.port]>;tag
=[input.local.tag]
  To: [input.remote.name]
<[system.sip.transport]:[input.remote.id]@[input.remote.address]:[input.remote.p
ort]>
  Max-Forwards: 70
  Call-ID: [transaction.callid]
      Cseq: [transaction.local.cseq] REGISTER
  Contact:
[system.sip.protocol]:[input.local.id]@[input.local.address]:[input.local.port];expire
s=[session.registration.ttl]
  Expires: 0
   <conditional var="session.authorization.length" op="greater" value="0"
type="integer">
  Authorization: [session.authorization]
  </conditional>

    ]]>
</message>
```

```
<set var="transaction.cseq" value="[registration.cseq]" />
<set var="transaction.callid" value="abc-[new.randhex.16]-xyz" />
<set var="transaction.branch"       value="[new.randhex.16]" />

<send message="unregister" protocol="sip" ip="[system.registrar.address]"
port="[system.registrar.address]" key="" retransmit="[system.sip.t1]">

<error>
<log level="Error" text="A socket error occured sending [transaction.name]
([transaction.method] [transaction.type]) to
[system.registrar.address]:[system.registrar.port]" />
</error>

<success>
<log level="Trace" text="Successfully sent a [transaction.name]
([transaction.method] [transaction.type]) to
[system.registrar.address]:[system.registrar.port]" />
</success>

  </send>
<incr var="registration.cseq" increment="1" />

<log level="all" text="Entering Unregister WaitEvents" />
<waitevents timeout="60">
<receive protocol="sip" method="REGISTER" type="response" code="100..199"
complete="false"/>


<receive protocol="sip" method="REGISTER" type="response" code="200..299"
complete="true">
<set var="session.registered" value="false"/>
<execute type="internal" id="session.registration.StopExpirationTimer" />
</receive >


<receive protocol="sip" method="REGISTER" type="response" code="401"
complete="true">
 <if var="session.authorization.length" op="equal" value="0" type="integer">
<execute type="internal" id="session.Authenticate.MD5"/>
    <if var="errorcode" op="equal" value="0" type="integer">
<execute type="transaction" id="Unregister"/>
</if>
</if>
</receive >
```

```
<receive protocol="sip" method="REGISTER" type="response" code="407"
complete="true">
 <if var="session.authorization.length" op="equal" value="0" type="integer">
<execute type="internal" id="session.ProxyAuthenticate.MD5"/>
    <if var="errorcode" op="equal" value="0" type="integer">
<execute type="transaction" id="Unregister"/>
</if>
</if>
</receive >

<receive protocol="sip" method="REGISTER" type="response" code="300..699"
complete="true"/>

<event id="user.stop" complete="true" />
<event id="user.terminate" complete="true" />
</waitevents>
<log level="all" text="Exiting Unregister WaitEvents" />
</transaction>

<session name="Basic UAC">

<dialog name="Basic Outgoing Telephony Call">

  <!-- The following are set by default when a dialog is created   -->
<!-- This is simply an example of how you can set data elements. -->


<set var="dialog.ip.protocol"          value="[system.ip.protocol]" />
<set var="dialog.sip.transport"  value="[system.sip.transport]" />
<set var="dialog.sip.protocol"   value="[system.sip.protocol]" />
<set var="dialog.sip.version"    value="[system.sip.version]" />
<set var="dialog.local.id"   value="[input.local.id]" />
<set var="dialog.local.address"  value="[input.local.address]" />
<set var="dialog.local.port"     value="[input.local.address]" />
<set var="dialog.remote.id"            value="[input.remote.id]" />
<set var="dialog.remote.address" value="[input.remote.address]" />
<set var="dialog.remote.port"    value="[input.remote.address]" />
<set var="dialog.callid"     value="abc-[dialog.new.callid]-xyz"/>
<set var="dialog.branch"         value="[dialog.new.branch]" />
<set var="dialog.local.tag"     value="[dialog.new.tag]" />

<!-- Step 1:                              -->
  <!-- Delay for initial delay if specified -->
<!--                                      -->
 <if var="input.initialdelay" op="not equal" value="0" type="integer">
<delay duration="[input.initialdelay]"/>
```

```
</if>

<!-- Step 2:                          -->
  <!-- Execute the INVITE transaction  -->
<!--                                  -->

<execute type="transaction" id="Basic Invite"/>

<!-- Step 3:                          -->
  <!-- Check code.  If success, start media -->
<!-- and call timers                      -->

    <if var="errorcode" op="equal" value="0" type="integer">
    <if var="input.duration" op="equal" value="0" type="integer">
<execute type="internal" id="dialog.StartCallDurationTimer" />
</if>
    <if var="errorcode" op="equal" value="0" type="integer">
<execute type="internal" id="dialog.Media.Start"/>
</if>
    <if var="errorcode" op="equal" value="0" type="integer">
<execute type="internal" id="dialog.DTMF.Start"/>
</if>
</if>

<!-- Step 4:                          -->
  <!-- wait for dialog events  -->
<!--                     -->

<log level="all" text="Entering Basic Outgoing Telephony Call WaitEvents" />
<waitevents timeout="60">
<receive protocol="sip" method="BYE" type="request" complete="true">
<respond code="100" text="Trying">
<delay duration="100"/>
</respond>
<respond code="200" text="OK" />
</receive >
<event id="dialog.calltimer" complete="true">
<execute type="transaction" id="Call Teardown"/>
</event>
</waitevents>
<log level="all" text="Exiting Basic Outgoing Telephony Call WaitEvents" />

<transaction name="Basic Invite" method="INVITE">

<set var="transaction.local.cseq" value="[session.cseq]"/>
<set var="transaction.content.payload" value="[dialog.new.sdp]" />
```

```
<send protocol="sip" ip="[dialog.remoteaddress]" port="[dialog.remoteport]"
key="" retransmit="[system.sip.t1]">

<![CDATA[
  INVITE
[dialog.sip.transport]:[dialog.remote.id]@[dialog.remote.address]:[dialog.remote.p
ort] [dialog.sip.protocol]/[dialog.sip.version]
  Via: [dialog.sip.protocol]/[dialog.sip.version]/[dialog.ip.protocol]
[dialog.local.address]:[dialog.local.port];branch=[dialog.branch]
  From: [input.local.name]
<[dialog.sip.transport]:[dialog.local.id]@[dialog.local.address]:[dialog.local.port]>;t
ag=[dialog.local.tag]
  To: [dialog.remote.name]
<[dialog.sip.transport]:[dialog.remote.id]@[dialog.remote.address]:[dialog.remote.
port]>
  Call-ID: [dialog.callid]
        Cseq: [transaction.local.cseq] INVITE
  Contact: [sip_protocol]:[local_id]@[local_addr]:[local_port]
     <conditional var="dialog.authorization.length" op="greater" value="0"
type="integer">
  Authorization: [dialog.authorization]
  </conditional>
  Max-Forwards: 70
     <conditional var="transaction.subject.length" op="greater" value="0"
type="integer">
  Subject: [transaction.subject]
  </conditional>
     <conditional var="transaction.content.type.length" op="greater" value="0"
type="integer">
Content-Type: [transaction.content.type]
  </conditional>
     <conditional var="transaction.content.length" op="greater" value="0"
type="integer">
  Content-Length: [transaction.content.length]
  </conditional>

     <conditional var="transaction.content.length" op="greater" value="0"
type="integer">
  [transaction.content.payload]
  </conditional>
  ]]>
  </send>

<if var="errorcode" op="equal" value="0" type="integer">
<set var="dialog.local.sdpsent" value="true"/>
```

```
<incr var="session.cseq" increment="1" />
<incr var="dialog.local.cseq" increment="1" />
</if>


<if var="errorcode" op="not equal" value="0" type="integer">
<log level="all" text="An error occured send a Basic Invite message ([errorcode])"
/>
</if>

<log level="all" text="Entering Basic Invite WaitEvents" />
<waitevents timeout="60">
<receive protocol="sip" method="INVITE" type="response" code="100..199"
complete="false">

<execute type="internal" id="dialog.get.remote.tag"/>
        <if var="last.response" op="contains" value="100rel" type="string">
<execute type="transaction" id="PRACK"/>
</if>
        <if var="last.response" op="contains" value="Record-Route:" type="string">
<execute type="internal" id="dialog.RecordRouteSet"/>
</if>
        <if var="last.response" op="contains" value="application/sdp:"
type="string">
<execute type="internal" id="dialog.GetSDP"/>
</if>
</receive >

<receive protocol="sip" method="INVITE" type="response" code="200..299"
complete="true">
<execute type="internal" id="dialog.get.remote.tag"/>
        <if var="last.response" op="contains" value="Record-Route:" type="string">
<execute type="internal" id="dialog.RecordRouteSet"/>
</if>
        <if var="last.response" op="contains" value="application/sdp:"
type="string">
<execute type="internal" id="dialog.GetSDP"/>
</if>
<execute type="transaction" id="Success Ack"/>
</receive >


<receive protocol="sip" method="INVITE" type="response" code="300..699"
complete="true">
  <set var="dialog.errorcode" value="[response.code]"/>
<execute type="transaction" id="Failure Ack"/>
```

```
</receive >
</waitevents>
<log level="all" text="Exiting Basic Invite WaitEvents" />
</transaction>

<transaction name="Success Ack" method="ACK">

<send protocol="sip" ip="[dialog.remoteaddress]" port="[dialog.remoteport]"
key="" retransmit="[system.sip.t1]">

<![CDATA[
 ACK
[dialog.sip.transport]:[dialog.remote.id]@[dialog.remote.address]:[dialog.remote.p
ort] [dialog.sip.protocol]/[dialog.sip.version]
     <conditional var="dialog.routeset.length" op="not equal" value="0"
type="integer">
 [dialog.routeset]
 </conditional>
 Via: [dialog.sip.protocol]/[dialog.sip.version]/[dialog.ip.protocol]
[dialog.local.address]:[dialog.local.port];branch=[dialog.new.branch]
 From: [input.local.name]
<[dialog.sip.transport]:[dialog.local.id]@[dialog.local.address]:[dialog.local.port]>;t
ag=[dialog.local.tag]
 To: [dialog.remote.name]
<[dialog.sip.transport]:[dialog.remote.id]@[dialog.remote.address]:[dialog.remote.
port]>
 Call-ID: [dialog.callid]
      Cseq: [transaction.local.cseq] ACK
 Contact:
[dialog.sip.transport]:[dialog.local.id]@[dialog.local.address]:[dialog.local.port]
 Max-Forwards: 70
 ]]>

</send>

</transaction>

<transaction name="Failure Ack" method="ACK">

<send protocol="sip" ip="[dialog.remoteaddress]" port="[dialog.remoteport]"
key="" retransmit="[system.sip.t1]">

<![CDATA[
 ACK
[dialog.sip.transport]:[dialog.remote.id]@[dialog.remote.address]:[dialog.remote.p
ort] [dialog.sip.protocol]/[dialog.sip.version]
```

```
      <conditional var="dialog.routeset.length" op="not equal" value="0"
type="integer">
  [dialog.routeset]
  </conditional>
    Via: [dialog.sip.protocol]/[dialog.sip.version]/[dialog.ip.protocol]
[dialog.local.address]:[dialog.local.port];branch=[dialog.branch]
  From: [input.local.name]
<[dialog.sip.transport]:[dialog.local.id]@[dialog.local.address]:[dialog.local.port]>;t
ag=[dialog.local.tag]
  To: [dialog.remote.name]
<[dialog.sip.transport]:[dialog.remote.id]@[dialog.remote.address]:[dialog.remote.
port]>
  Call-ID: [dialog.callid]
        Cseq: [transaction.local.cseq] ACK
  Contact:
[dialog.sip.transport]:[dialog.local.id]@[dialog.local.address]:[dialog.local.port]
  Max-Forwards: 70
      <conditional var="dialog.local.sdp.sent" op="not equal" value="true"
type="boolean">
  Content-Type: [transaction.content.type]
Content-Length: [transaction.content.length]

[transaction.content.payload]
  </conditional>
]]>

        <if var="dialog.local.sdp.sent" op="not equal" value="true" type="boolean">
<set var="dialog.local.sdpsent" value="true"/>
</if>

</send>

</transaction>

<transaction name="Call Teardown" method="BYE">

<send protocol="sip" ip="[dialog.remoteaddress]" port="[dialog.remoteport]"
key="" retransmit="[system.sip.t1]">

<![CDATA[
  BYE
[dialog.sip.transport]:[dialog.remote.id]@[dialog.remote.address]:[dialog.remote.p
ort] [dialog.sip.protocol]/[dialog.sip.version]
      <conditional var="dialog.routeset.length" op="not equal" value="0"
type="integer">
  [dialog.routeset]
```

```
  </conditional>
  Via: [dialog.sip.protocol]/[dialog.sip.version]/[dialog.ip.protocol]
[dialog.local.address]:[dialog.local.port];branch=[dialog.new.branch]
  From: [input.local.name]
<[dialog.sip.transport]:[dialog.local.id]@[dialog.local.address]:[dialog.local.port]>;t
ag=[dialog.local.tag]
  To: [dialog.remote.name]
<[dialog.sip.transport]:[dialog.remote.id]@[dialog.remote.address]:[dialog.remote.
port]>
  Call-ID: [dialog.callid]
       Cseq: [transaction.local.cseq] BYE
  Contact:
[dialog.sip.transport]:[dialog.local.id]@[dialog.local.address]:[dialog.local.port]
  Max-Forwards: 70

]]>

</send>

<log level="all" text="Entering Call Teardown WaitEvents" />
<waitevents timeout="60">
<receive protocol="sip" method="BYE" type="response" code="100..199"
complete="false"/>
<receive protocol="sip" method="BYE" type="response" code="200..299"
complete="true"/>
<receive protocol="sip" method="BYE" type="response" code="300..699"
complete="true">
  <set var="dialog.errorcode" value="[response.code]"/>
</receive >
</waitevents>
<log level="all" text="Exiting Call Teardown WaitEvents" />

</transaction>

</dialog>


<plan>
<log level="debug" text="Starting UAC..."/>
<execute type="transaction" id="Register">
<log level="trace" text="1. Registered"/>
    <if var="last.transaction.result" op="less" value="300" type="integer">

<log level="trace" text="2. Entering Loop..."/>
<repeat iterations="[input.frequency]">
```

```
<log level="trace" text="3. Executing Dialog..."/>
<execute type="dialog" id="Basic Outgoing Telephony Call">
</execute>
</repeat>
<log level="trace" text="4. Exited Loop."/>

<execute type="transaction" id="Unregister"/>
<log level="trace" text="5. Unregistered"/>
</if>
</execute>
<log level="debug" text="Finished UAC."/>
</plan>

</session>

<session name="Basic UAS">


<dialog name="Basic Incoming Telephony Call">

  <!-- The following are set by default when a dialog is created   -->
<!-- This is simply an example of how you can set data elements. -->


<set var="dialog.ip.protocol" value="[system.ip.protocol]" />
<set var="dialog.sip.transport" value="[system.sip.transport]" />
<set var="dialog.sip.protocol"  value="[system.sip.protocol]" />
<set var="dialog.sip.version"   value="[system.sip.version]" />
<set var="dialog.local.id"        value="[input.local.id]" />
<set var="dialog.local.address" value="[input.local.address]" />
<set var="dialog.local.port"    value="[input.local.port]" />
<set var="dialog.remote.id" value="[input.remote.id]" />
<set var="dialog.remote.address"value="[input.remote.address]" />
<set var="dialog.remote.port"   value="[input.remote.address]" />
<set var="dialog.callid"        value="abc-[dialog.new.callid]-xyz"/>
<set var="dialog.branch"        value="[dialog.new.branch]" />
<set var="dialog.local.tag"     value="[dialog.new.tag]" />

<log level="all" text="Entering Basic UAS WaitEvents" />
<waitevents>
<receive protocol="sip" method="INVITE" template="new invite" type="request"
complete="false">

<!-- Step 1:                          -->
  <!-- Delay for initial delay if specified -->
<!--                                  -->
```

```
    <if var="input.postdialdelay" op="greater" value="0" type="integer">
<delay duration="[input.postdialdelay]"/>
</if>

<!-- Step 2:                          -->
  <!-- Respond with trying transaction        -->
<!--                                 -->
<respond code="100" text="Trying"/>
<respond code="180" text="Ring-a-ling-ling">
    <if var="input.ringduration" op="greater" value="0" type="integer">
<delay duration="[input.ringduration]"/>
</if>
</respond>
<respond code="200" text="Here's to your success!" retransmit="500" />
</receive>
<receive protocol="sip" method="ACK" type="request" complete="false">
<execute type="internal" id="dialog.StartCallDurationTimer" />
<execute type="internal" id="dialog.Media.Start"/>
<execute type="internal" id="dialog.DTMF.Start"/>
</receive >
<receive protocol="sip" method="CANCEL" type="request" complete="true">
</receive >
<event id="call.duration.timer" complete="true">
<execute type="transaction" id="Call Teardown"/>
</event>
</waitevents>

<log level="all" text="Exited Basic UAS WaitEvents" />

<transaction name="UAS Call Teardown" method="BYE">

<message name="bye">

   <![CDATA[
  BYE
[dialog.sip.transport]:[dialog.remote.id]@[dialog.remote.address]:[dialog.remote.p
ort] [dialog.sip.protocol]/[dialog.sip.version]
  [dialog.routeset]
    Via: [dialog.sip.protocol]/[dialog.sip.version]/[dialog.ip.protocol]
[dialog.local.address]:[dialog.local.port];branch=[new.branch]
  From: [input.local.name]
<[dialog.sip.transport]:[dialog.local.id]@[dialog.local.address]:[dialog.local.port]>;t
ag=[dialog.local.tag]
  To: [dialog.remote.name]
<[dialog.sip.transport]:[dialog.remote.id]@[dialog.remote.address]:[dialog.remote.
port]>;tag=[dialog.remote.tag]
```

```
  Call-ID: [dialog.callid]
       Cseq: [transaction.local.cseq] BYE
  Max-Forwards: 70

]]>
</message>

<send message="bye" ip="[dialog.remote.address]" port="[dialog.remote.port]"
retransmit="500" />

<log level="all" text="Entering UAS CallTeardown WaitEvents" />
<waitevents timeout="60">
<log level="all" text="Entering Call Teardown WaitEvents" />
<receive protocol="sip" method="BYE" type="response" code="100..199"
complete="false"/>
<receive protocol="sip" method="BYE" type="response" code="200..299"
complete="true"/>
<receive protocol="sip" method="BYE" type="response" code="300..699"
complete="true">
  <set var="dialog.errorcode" value="[response.code]"/>
</receive >
</waitevents>
<log level="all" text="Exiting UAS CallTeardown WaitEvents" />
</transaction>
</dialog>

<plan>
<log level="debug" text="Starting UAS..."/>
<execute type="transaction" id="Register"/>
<log level="trace" text="1. Registered [input.local.id]"/>

 <if var="last.transaction.result" op="less" value="300" type="integer">

<log level="trace" text="2. Entering Loop..."/>
<repeat iterations="[input.frequency]">

<log level="trace" text="3. Executing Dialog..."/>
<execute type="dialog" id="Basic Incoming Telephony Call">
</execute>

</repeat>

<log level="trace" text="4. Exited Loop."/>
<execute type="transaction" id="Unregister"/>
<log level="trace" text="5. Unregistered"/>
</if>
```

```
<log level="debug" text="Finished UAS."/>
</plan>

</session>

<plan>
<log level="info" text="Starting Blueprint..."/>
<execute type="session" id="Basic UAS"/>
<!-- <execute type="session" id="Basic UAS"/> -->
<log level="info" text="Finished Blueprint."/>
</plan>

</blueprint>
```

Description:

A series of sessions.

Comments:

This can be used to execute up multiple sessions. Parent node of all other VoIPML nodes. There can only be one blueprint per file, and it must contain a plan to control it.

Tag: session

Attributes:

name - the name of the session variable.

Example:

```
<session name="Basic UAS">


<dialog name="Basic Incoming Telephony Call">

  <!-- The following are set by default when a dialog is created   -->
<!-- This is simply an example of how you can set data elements. -->


<set var="dialog.ip.protocol"value="[system.ip.protocol]" />
<set var="dialog.sip.transport" value="[system.sip.transport]" />
<set var="dialog.sip.protocol"  value="[system.sip.protocol]" />
<set var="dialog.sip.version"   value="[system.sip.version]" />
<set var="dialog.local.id"        value="[input.local.id]" />
<set var="dialog.local.address" value="[input.local.address]" />
<set var="dialog.local.port"    value="[input.local.port]" />
<set var="dialog.remote.id" value="[input.remote.id]" />
<set var="dialog.remote.address"value="[input.remote.address]" />
<set var="dialog.remote.port"   value="[input.remote.address]" />
<set var="dialog.callid"        value="abc-[dialog.new.callid]-xyz"/>
<set var="dialog.branch"        value="[dialog.new.branch]" />
<set var="dialog.local.tag"     value="[dialog.new.tag]" />

<log level="all" text="Entering Basic UAS WaitEvents" />
<waitevents>
<receive protocol="sip" method="INVITE" template="new invite" type="request"
complete="false">

<!-- Step 1:                          -->
  <!-- Delay for initial delay if specified -->
<!--                                  -->
    <if var="input.postdialdelay" op="greater" value="0" type="integer">
<delay duration="[input.postdialdelay]"/>
</if>

<!-- Step 2:                          -->
```

```
  <!-- Respond with trying transaction        -->
<!--                                  -->
<respond code="100" text="Trying"/>
<respond code="180" text="Ring-a-ling-ling">
    <if var="input.ringduration" op="greater" value="0" type="integer">
<delay duration="[input.ringduration]"/>
</if>
</respond>
<respond code="200" text="Here's to your success!" retransmit="500" />
</receive>
<receive protocol="sip" method="ACK" type="request" complete="false">
<execute type="internal" id="dialog.StartCallDurationTimer" />
<execute type="internal" id="dialog.Media.Start"/>
<execute type="internal" id="dialog.DTMF.Start"/>
</receive >
<receive protocol="sip" method="CANCEL" type="request" complete="true">
</receive >
<event id="call.duration.timer" complete="true">
<execute type="transaction" id="Call Teardown"/>
</event>
</waitevents>

<log level="all" text="Exited Basic UAS WaitEvents" />

<transaction name="UAS Call Teardown" method="BYE">

<message name="bye">

   <![CDATA[
  BYE
[dialog.sip.transport]:[dialog.remote.id]@[dialog.remote.address]:[dialog.remote.p
ort] [dialog.sip.protocol]/[dialog.sip.version]
  [dialog.routeset]
   Via: [dialog.sip.protocol]/[dialog.sip.version]/[dialog.ip.protocol]
[dialog.local.address]:[dialog.local.port];branch=[new.branch]
  From: [input.local.name]
<[dialog.sip.transport]:[dialog.local.id]@[dialog.local.address]:[dialog.local.port]>;t
ag=[dialog.local.tag]
  To: [dialog.remote.name]
<[dialog.sip.transport]:[dialog.remote.id]@[dialog.remote.address]:[dialog.remote.
port]>;tag=[dialog.remote.tag]
  Call-ID: [dialog.callid]
       Cseq: [transaction.local.cseq] BYE
  Max-Forwards: 70

]]>
```

```
</message>

<send message="bye" ip="[dialog.remote.address]" port="[dialog.remote.port]"
retransmit="500" />

<log level="all" text="Entering UAS CallTeardown WaitEvents" />
<waitevents timeout="60">
<log level="all" text="Entering Call Teardown WaitEvents" />
<receive protocol="sip" method="BYE" type="response" code="100..199"
complete="false"/>
<receive protocol="sip" method="BYE" type="response" code="200..299"
complete="true"/>
<receive protocol="sip" method="BYE" type="response" code="300..699"
complete="true">
  <set var="dialog.errorcode" value="[response.code]"/>
</receive >
</waitevents>
<log level="all" text="Exiting UAS CallTeardown WaitEvents" />
</transaction>
</dialog>

<plan>
<log level="debug" text="Starting UAS..."/>
<execute type="transaction" id="Register"/>
<log level="trace" text="1. Registered [input.local.id]"/>

 <if var="last.transaction.result" op="less" value="300" type="integer">

<log level="trace" text="2. Entering Loop..."/>
<repeat iterations="[input.frequency]">

<log level="trace" text="3. Executing Dialog..."/>
<execute type="dialog" id="Basic Incoming Telephony Call">
</execute>

</repeat>

<log level="trace" text="4. Exited Loop."/>
<execute type="transaction" id="Unregister"/>
<log level="trace" text="5. Unregistered"/>
</if>
<log level="debug" text="Finished UAS."/>
</plan>

</session>
```

Description:

A series of dialogs.

Comments:

This can be used to execute multiple dialogs.

Tag: plan

Attributes:

none

Example:

<session>
<plan>

<log level=”all” text=”Hello” />

</plan >
</session>

Description:

Required element for a session in a blueprint. This is what is actually performed by the session, while the rest is its description of variables and other settings.

Comments:

Basically, this is what a session is actively doing with the information it is given.

Tag: dialog

Attributes:

name - the name of the dialogue variable.

Example:

```
<dialog name="Basic Incoming Telephony Call">

  <!-- The following are set by default when a dialog is created   -->
<!-- This is simply an example of how you can set data elements. -->


<set var="dialog.ip.protocol"value="[system.ip.protocol]" />
<set var="dialog.sip.transport" value="[system.sip.transport]" />
<set var="dialog.sip.protocol"  value="[system.sip.protocol]" />
<set var="dialog.sip.version"   value="[system.sip.version]" />
<set var="dialog.local.id"        value="[input.local.id]" />
<set var="dialog.local.address" value="[input.local.address]" />
<set var="dialog.local.port"    value="[input.local.port]" />
<set var="dialog.remote.id" value="[input.remote.id]" />
<set var="dialog.remote.address"value="[input.remote.address]" />
<set var="dialog.remote.port"   value="[input.remote.address]" />
<set var="dialog.callid"        value="abc-[dialog.new.callid]-xyz"/>
<set var="dialog.branch"        value="[dialog.new.branch]" />
<set var="dialog.local.tag"     value="[dialog.new.tag]" />

<log level="all" text="Entering Basic UAS WaitEvents" />
<waitevents>
<receive protocol="sip" method="INVITE" template="new invite" type="request"
complete="false">

<!-- Step 1:                          -->
  <!-- Delay for initial delay if specified -->
<!--                                  -->
    <if var="input.postdialdelay" op="greater" value="0" type="integer">
<delay duration="[input.postdialdelay]"/>
</if>

<!-- Step 2:                          -->
  <!-- Respond with trying transaction        -->
<!--                                  -->
<respond code="100" text="Trying"/>
```

```
<respond code="180" text="Ring-a-ling-ling">
    <if var="input.ringduration" op="greater" value="0" type="integer">
<delay duration="[input.ringduration]"/>
</if>
</respond>
<respond code="200" text="Here's to your success!" retransmit="500" />
</receive>
<receive protocol="sip" method="ACK" type="request" complete="false">
<execute type="internal" id="dialog.StartCallDurationTimer" />
<execute type="internal" id="dialog.Media.Start"/>
<execute type="internal" id="dialog.DTMF.Start"/>
</receive >
<receive protocol="sip" method="CANCEL" type="request" complete="true">
</receive >
<event id="call.duration.timer" complete="true">
<execute type="transaction" id="Call Teardown"/>
</event>
</waitevents>

<log level="all" text="Exited Basic UAS WaitEvents" />

<transaction name="UAS Call Teardown" method="BYE">

<message name="bye">

   <![CDATA[
  BYE
[dialog.sip.transport]:[dialog.remote.id]@[dialog.remote.address]:[dialog.remote.p
ort] [dialog.sip.protocol]/[dialog.sip.version]
  [dialog.routeset]
    Via: [dialog.sip.protocol]/[dialog.sip.version]/[dialog.ip.protocol]
[dialog.local.address]:[dialog.local.port];branch=[new.branch]
  From: [input.local.name]
<[dialog.sip.transport]:[dialog.local.id]@[dialog.local.address]:[dialog.local.port]>;t
ag=[dialog.local.tag]
  To: [dialog.remote.name]
<[dialog.sip.transport]:[dialog.remote.id]@[dialog.remote.address]:[dialog.remote.
port]>;tag=[dialog.remote.tag]
  Call-ID: [dialog.callid]
       Cseq: [transaction.local.cseq] BYE
  Max-Forwards: 70

]]>
</message>
```

```
<send message="bye" ip="[dialog.remote.address]" port="[dialog.remote.port]"
retransmit="500" />

<log level="all" text="Entering UAS CallTeardown WaitEvents" />
<waitevents timeout="60">
<log level="all" text="Entering Call Teardown WaitEvents" />
<receive protocol="sip" method="BYE" type="response" code="100..199"
complete="false"/>
<receive protocol="sip" method="BYE" type="response" code="200..299"
complete="true"/>
<receive protocol="sip" method="BYE" type="response" code="300..699"
complete="true">
  <set var="dialog.errorcode" value="[response.code]"/>
</receive >
</waitevents>
<log level="all" text="Exiting UAS CallTeardown WaitEvents" />
</transaction>
</dialog>
```

Description:

A series of transactions.

Comments:

Can be used to execute up multiple transactions at once.

Tag: transaction

Attributes:

name - the name of the transaction variable.

method - type of the message. For example, for INVITE message the method will be “INVITE” and for ACK message the method will be “ACK”.

Example:

```
<transaction name="Basic Invite" method="INVITE">

<set var="transaction.local.cseq" value="[session.cseq]"/>
<set var="transaction.content.payload" value="[dialog.new.sdp]" />

<send protocol="sip" ip="[dialog.remoteaddress]" port="[dialog.remoteport]"
key="" retransmit="[system.sip.t1]">

<![CDATA[
  INVITE
[dialog.sip.transport]:[dialog.remote.id]@[dialog.remote.address]:[dialog.remote.p
ort] [dialog.sip.protocol]/[dialog.sip.version]
  Via: [dialog.sip.protocol]/[dialog.sip.version]/[dialog.ip.protocol]
[dialog.local.address]:[dialog.local.port];branch=[dialog.branch]
  From: [input.local.name]
<[dialog.sip.transport]:[dialog.local.id]@[dialog.local.address]:[dialog.local.port]>;t
ag=[dialog.local.tag]
  To: [dialog.remote.name]
<[dialog.sip.transport]:[dialog.remote.id]@[dialog.remote.address]:[dialog.remote.
port]>
  Call-ID: [dialog.callid]
        Cseq: [transaction.local.cseq] INVITE
  Contact: [sip_protocol]:[local_id]@[local_addr]:[local_port]
     <conditional var="dialog.authorization.length" op="greater" value="0"
type="integer">
  Authorization: [dialog.authorization]
  </conditional>
  Max-Forwards: 70
     <conditional var="transaction.subject.length" op="greater" value="0"
type="integer">
  Subject: [transaction.subject]
  </conditional>
```

```
    <conditional var="transaction.content.type.length" op="greater" value="0"
type="integer">
Content-Type: [transaction.content.type]
 </conditional>
    <conditional var="transaction.content.length" op="greater" value="0"
type="integer">
 Content-Length: [transaction.content.length]
 </conditional>

    <conditional var="transaction.content.length" op="greater" value="0"
type="integer">
 [transaction.content.payload]
 </conditional>
 ]]>
 </send>

<if var="errorcode" op="equal" value="0" type="integer">
<set var="dialog.local.sdpsent" value="true"/>
<incr var="session.cseq" increment="1" />
<incr var="dialog.local.cseq" increment="1" />
</if>


<if var="errorcode" op="not equal" value="0" type="integer">
<log level="all" text="An error occured send a Basic Invite message ([errorcode])"
/>
</if>

<log level="all" text="Entering Basic Invite WaitEvents" />
<waitevents timeout="60">
<receive protocol="sip" method="INVITE" type="response" code="100..199"
complete="false">

<execute type="internal" id="dialog.get.remote.tag"/>
      <if var="last.response" op="contains" value="100rel" type="string">
<execute type="transaction" id="PRACK"/>
</if>
      <if var="last.response" op="contains" value="Record-Route:" type="string">
<execute type="internal" id="dialog.RecordRouteSet"/>
</if>
      <if var="last.response" op="contains" value="application/sdp:"
type="string">
<execute type="internal" id="dialog.GetSDP"/>
</if>
</receive >
```

```
<receive protocol="sip" method="INVITE" type="response" code="200..299"
complete="true">
<execute type="internal" id="dialog.get.remote.tag"/>
        <if var="last.response" op="contains" value="Record-Route:" type="string">
<execute type="internal" id="dialog.RecordRouteSet"/>
</if>
        <if var="last.response" op="contains" value="application/sdp:"
type="string">
<execute type="internal" id="dialog.GetSDP"/>
</if>
<execute type="transaction" id="Success Ack"/>
</receive >


<receive protocol="sip" method="INVITE" type="response" code="300..699"
complete="true">
  <set var="dialog.errorcode" value="[response.code]"/>
<execute type="transaction" id="Failure Ack"/>
</receive >
</waitevents>
<log level="all" text="Exiting Basic Invite WaitEvents" />
</transaction>
```

Description:

Used to either send something and receive something in response or vice versa.

Comments:

Useful for making send/receive transactions.

Messaging and Control Elements

Tag: execute

Attributes:

type - selects the type of command to execute (“internal,” “session,” dialogue,” “transaction,” “function”).

id - selects the command to execute (constraints are yet to be fully defined).

Example:

<execute type="internal" id="session.registration.StopExpirationTimer" />

Description:

Used to execute commands.

Comments:

Executes commands from an inward out direction, starting with the file the tag itself is written on, moving outwards to its parents and their parents, et cetera until it reaches the command or it runs out of files.

Tag: template

Attributes:

name - the name of the template variable

type - the type of message the template is designed for

Example:

```
<template name="new invite" type="request">
<![CDATA[
  "INVITE"{='
'[last.invite.request.transport]':'}{=':'[last.invite.target.id]'@'}{='@'[last.invite.target.
address]':| '}{?':'[last.invite.target.port]' '}{='
'[last.invite.request.protocol]'/'}{='/'[last.invite.request.version]}

"Via:"{=[last.invite.via.protocol]'/'}{='/'[last.invite.via.version]'/'}{='/'[last.invite.via.pr
otocol]' '}{='
'[last.invite.via.address]':|;|,|'}{?':'[last.invite.via.port]';|,|'}{'^;branch='[last.via.branc
h]':|;|,|'}{=',#repeat}

"From:"{?[last.invite.remote.name]'<'}{='<'[last.invite.sip.transport]':'}{=':'[last.invite
.remote.id]'@'}{='@'[last.invite.remote.address]':|;|,|'}{?':'[last.invite.remote.port]';|'}
{^'tag='[last.invite.remote.tag]';|'}

"To:"{?[last.invite.local.name]'<'}{='<'[last.invite.local.transport]':
'}{=':'[last.invite.local.id]'@'}{='@'[last.invite.local.address]':|;|'
}{'?:'[last.invite.local.port]"|;|'}
  "Call-ID:"{=[last.invite.callid]}
      "Cseq:"{=[last.invite.remote.cseq]' '}{=' '[last.invite.method]}

"Contact:"{=[last.invite.contact.protocol]':'}{=':'[last.invite.contact.id]'@'}{'@'[last.in
vite.contact.address]':|;|'}{?':'[last.invite.contact.port]':|;|'}
  "Authorization:"{=' '[authorization.method]'
'}{^user="[authorization.username]',|'}{^'realm='[authorization.realm]',|'}{^'nonce='[
authorization.nonce]',|'}{^'uri='[authorization.uri]',|'}{^'response='[authorization.res
ponse]',|'}{^'algorithm='[authorization.algorithm]',|'}
  "Max-Forwards:"{=[last.invite.maxforwards]}
  "Subject:"{=[last.invite.subject]}
  "Content-Type:"{=[last.invite.content.type]}
  "Content-Length:"{=[last.invite.content.length]}
  {=[last.invite.content.payload]}
]]>
</template>
```

Description:

Takes certain sections of a message and loads them into predefined variables

Comments:

Useful for parsing and separating the sections of a message or for decoding. This is also an optional field for the receive tag.

Tag: header

Attributes:

name - the name of the header variable.

tokens - marks the beginning and end of what is read in as well as what variable to read it into. See Comments section below for important syntax notes.

multiple - optional field, when set to true allows for multiple instances of the same information to be read in.

counter - when using the multiple attribute, the variable where the counter is stored.

Example:

```
<template name="maxi invite" type="request" separators="' ',':','='">
  <header name="INVITE
"tokens="{="[last.invite.request.transport]':'}{=':'[last.invite.target.id]'@'}{='@'[last.i
nvite.target.address]':| '}{?':'[last.invite.target.port]' '}{='
'[last.invite.request.protocol]'/'}{='/'[last.invite.request.version]}" />
  <!--<header
name="Via:"tokens="{="[last.invite.via.counter.protocol]'/'}{='/'[last.invite.via.count
er.version]'/'}{='/'[last.invite.via.counter.protocol]' '}{='
'[last.invite.via.counter.address]':|;|,|'}{?':'[last.invite.via.counter.port]';|,|'}{^';branch
='[last.via.counter.branch]':|;|,|'}" multiple="true"
counter="last.invite.request.vias"/> -->
  <header name="Via:"tokens="{="[last.via]"}" multiple="true"
counter="last.invite.request.vias"/>
  <header
name="From:"tokens="{?"[last.invite.remote.name]'&lt'}{='&lt'[last.invite.sip.transp
ort]':'}{=':'[last.invite.remote.id]'@'}{='@'[last.invite.remote.address]':|;|,|'}{?':'[last.in
vite.remote.port]'&gt|;|'}{^'tag='[last.invite.remote.tag]';|'}" />
  <header
name="To:"tokens="{?"[last.invite.local.name]'&lt'}{='&lt'[last.invite.local.transport]
':'}{=':'[last.invite.local.id]'@'}{='@'[last.invite.local.address]':|;|'}{?':'[last.invite.local
.port]'&gt|;|'}" />
  <header name="Call-ID:"tokens="{="[last.invite.callid]}" />
  <header name="CSeq:"tokens="{="[last.invite.remote.cseq]' '}{='
'[last.invite.method]"}" />
  <header
name="Contact:"tokens="{="[last.invite.contact.counter.protocol]':'}{=':'[last.invite.
```

```
contact.counter.id]'@'}{='@'[last.invite.contact.counter.address]':|;|'}{?':'[last.invite.
contact.counter.port]':|;|'}{^'q='[last.invite.contact.counter.q]',|'}{^'received='[last.in
vite.contact.counter.received]',|'}" multiple="true"
counter="last.invite.request.contacts" />
  <header name="Authorization:" tokens="{="[authorization.method]'
'}{^'user='[authorization.username]',|'}{^'realm='[authorization.realm]',|'}{^'nonce='[
authorization.nonce]',|'}{^'uri='[authorization.uri]',|'}{^'response='[authorization.res
ponse]',|'}{^'algorithm='[authorization.algorithm]',|'}" />
  <header name="Max-Forwards:"tokens="{="[last.invite.maxforwards]}" />
  <header name="Subject:"tokens="{="[last.invite.subject]}" />
  <header name="Content-Type:"tokens="{="[last.invite.content.type]}" />
  <header name="Content-Length:"tokens="{="[last.invite.content.length]}" />
  <header name="Record-Route:"tokens="{='&lt'[last.invite.record-
route.counter.transport]':'}{=':'[last.invite.record-
route.counter.id]'@'}{='@'[last.invite.record-
route.counter.address]':|;|,|'}{?':'[last.invite.record-
route.counter.port]'&gt|;|'}{^'ftag='[last.invite.record-
route.counter.ftag]'&gt|;|'}{^'lr'[last.invite.record-route.counter.ftag]'&gt|;|lr'}"
multiple="true" counter="last.invite.request.record-routes" />
  <header name="v="tokens="{="[last.invite.sdp.version]"}" />
  <header name="o="tokens="{="[last.invite.sdp.id]' '}{=' '[last.invite.sdp.t1]' '}{='
'[last.invite.sdp.t2]' '}{=' '[last.invite.sdp.inout]' '}{=' '[last.invite.sdp.ipversion]' '}{='
'[last.invite.sdp.ipaddress]"}" />
  <header name="s="tokens="{="[last.invite.sdp.subject]"}" />
  <header name="i="tokens="{="[last.invite.sdp.info]"}" />
  <header name="c="tokens="{="[last.invite.sdp.connection.inout]' '}{='
'[last.invite.sdp.connection.ipversion]' '}{='
'[last.invite.sdp.connection.ipaddress]"}" />
  <header name="t="tokens="{="[last.invite.sdp.time.t1]' '}{='
'[last.invite.sdp.time.t2]"}" />
  <!-- <header
name="a="tokens="{="[last.invite.sdp.attribute.counter.attr]':|'}{?':'[last.invite.sdp.a
ttribute.counter.id]' '}{?' '[last.invite.sdp.attribute.counter.desc]"}" multiple="true"
counter="last.invite.media.attributes" /> -->
  <group  name="m="         tokens="{="[last.invite.sdp.media.counter.name]' '}{='
'[last.invite.sdp.media.counter.port]' '}{=' '[last.invite.sdp.media.counter.type]'
'}{r[last.invite.sdp.media.counter.codecs]'
'[last.invite.sdp.media.counter.codec.rcount]' |'}" multiple="true"
counter="last.invite.sdp.medias" groupname="last.invite.sdp.media.counter">
  <item
name="a="tokens="{="[groupname.attr.counter]':|'}{?':'[groupname.id.counter]'
'}{?' '[groupname.desc.counter]"}" multiple="true" counter="groupname.attributes"
/>
  </group>
  <!--
```

do the groups!!!

```
                "m=audio 40000 RTP/AVP 8 0 9 4 4 15 18 102 101\n"
                "a=sendrecv\n"
                "a=rtpmap:0 PCMU/8000\n"
                "a=rtpmap:8 PCMA/8000\n"
                "a=rtpmap:9 G722/8000\n"
                "a=rtpmap:4 G723/8000\n"
                "a=rtpmap:15 G728/8000\n"
                "a=rtpmap:18 G729/8000\n"
                "a=rtpmap:102 AMR/8000\n"
                "a=rtpmap:101 telephone-event/8000\n"
                "a=fmtp:101 0-15,16\n"
                "m=video 40002 RTP/AVP 31 34\n"
                "a=sendrecv\n"
                "a=rtpmap:34 H263/90000\n"
                "a=rtpmap:31 H261/90000\n");
-->
</template>
```

Description:

Used within a template tag to select and store various sections of a message that is received within several variables.

Comments:

Useful for parsing and separating the sections of a message line for decoding.

Token syntax:

The start and end what a variable is to store is marked within single quotations (for example ‘:’).

If there are multiple possibilities to start and/or stop the syntax, each possibility is separated by a | within the quotation marks (for example “.|:” denotes that it will start or stop if it sees either a period or a colon).

If the section of text must start with the designated character(s), an equal sign precedes the single quotes at the beginning.

If the section of text does not have to start with the designated character(s), but still may, a question mark precedes the single quotes in the beginning.

If the section of text may or may not contain the character(s), but may have them in any order, a q= precedes the single quotes at the beginning.

If a section of text must start with the designated character(s) and repeats itself, the letter r precedes the single quotes at the beginning.

Tag: message

Attributes:

name - the name of the message variable

Example:

```
<message name=”sample”>

<![CDATA[
  REGISTER
[system.sip.transport]:[system.registrar.address]:[system.registrar.port]
[system.sip.protocol]/[system.sip.version]
  Via: [system.sip.protocol]/[system.sip.version]/[system.ip.protocol]
[input.local.address]:[input.local.port];branch=[input.branch]
  From: [input.local.name]
<[system.sip.transport]:[input.local.id]@[system.registrar.address]:[system.registr
ar.port]>;tag=[input.local.tag]
  To: [input.remote.name]
<[system.sip.transport]:[input.remote.id]@[system.registrar.address]:[system.regi
strar.port]>
  Max-Forwards: [transaction.maxforwards]
  Call-ID: [transaction.callid]
  Cseq: [transaction.local.cseq] REGISTER
  Contact:
[system.sip.protocol]:[input.local.id]@[input.local.address]:[input.local.port];expire
s=[session.registration.ttl]
  Expires: [session.registration.ttl]
  <conditional var="session.authorization.length" op="greater" value="0"
type="integer">
  Authorization: [authorization.method] user="[authorization.username]",
realm="[authorization.realm]", nonce="[authorization.nonce]",
uri="[authorization.uri]", response="[authorization.response]",
algorithm=[authorization.algorithm]
  </conditional>

]]>

</message>
```

Description:

Defines message types used by the message attribute in the send and receive tags

Comments:

These are needed to define what the messages do

Tag: send

Attributes:

message - optional field, sends the message, must be selected from the message table.

protocol - the protocol that the message is using.

ip - ip address that is sent to.

port - the port which the message is being sent to.

key - optional field, yet to be defined.

retransmit - time to retransmit in milliseconds.

Example:

<send message=”unregister” protocol=”sip” ip=”120.123.333.22” port=”231000” key=”” retramsmit=”300”>

</send>

Description:

Sends the message to a receiver that is defined in the ip and port protocols.

Comments:

Necessary for the host to communicate with other clients.

Tag: waitevents

Attributes:

None

Example:

<waitevents>

</waitevents>

Description:

Starts a wait event state.

Comments:

This is needed to receive as well as other wait events such as timers and other passive operations.

Tag: receive

Attributes:

protocol - protocol from which to receive.

method - the type of message you are receiving.

type - either request or response, what type of message are you responding to. To receive a response, you must first send a request.

code – the code for the message being received.

complete – if set to true, ends the wait events state.

Example:

```
<receive protocol="sip" method="REGISTER" type="response"
code="100..199" complete="false"/>
```

Description:

This is used to receive messages from a sender.

Comments:

This is needed to obtain messages from other clients.

Tag: event

Attributes:

id - selects the event to occur (constraints on this list are yet to be fully defined).

complete - when set to “true,” ends the wait event state

Example:

<event id=”user.stop” complete=”true” />

Description:

Used to detect an event in the wait event state and when doing so, take some action, such as break out of the state

Comments:

Useful for stopping a program when errors occur.

Tag: timer

Attributes:

id - starts the timer with selected id.

duration – sets the duration of the selected timer.

Example:

<timer id="call.duration.timer" duration="[input.call.duration]" />

Description:

Used to start a particular timer.

Comments:

With this tag you can start a timer with a particular id and you set the duration of the timer.

Tag: stoptimer

Attributes:

id - stops the selected timer.

Example:

<stoptimer id="call.duration.timer" />

Description:

Used to stop a particular timer.

Comments:

With this tag you can stop a particular timer id which was started before.

Tag: senddtmf

Attributes:

events – sends the string of DTMF events.

Example:

<senddtmf events="12345" />

Description:

Used to send DTMF digits.

Comments:

You can send string of DTMF events or single DTMF event.

Tag: recvdtmf

Attributes:

digits – receives the string of DTMF events.

complete - when set to “true,” ends the wait event state

beep - when set to “true” beeps at the end of the DTMF string.

Example:

<recvdtmf digits="1234" complete="true" beep="true">

</recvdtmf >

Description:

Used to receive DTMF digits.

Comments:

You can receive string of DTMF events or single DTMF event.

Tag: break

Attributes:

none

Examples:

<break />

Description:

Used to break from a loop.

Comments:

With this tag you can break from a loop.

Tag: exit

Attributes:

label - sets the status of the call.

Examples:

<exit label="cancel handler" />

Description:

Used to set the label for exit.

Comments:

With this tag you can set label for the exit and it will be shown in the error notification.

Tag: status

Attributes:

status - sets the status of the call.

Examples:

<status status="trying" />
<status status="ringing" />
<status status="connected" />

Description:

Used to set the status of the calls.

Comments:

With this tag you can change the status of the calls.

Tag: repeat

Attributes:

iterations - the number of times to repeat the section of code.

Example:

<repeat iterations=”4”>

<log level=”info” text=”sample run 4 times” />

</repeat>

Description:

Used to create loops that run a specific number of times defined in its iteration attribute.

Comments:

Useful for increasing and decreasing counters, defining array variables, and other loop required actions.

Tag: delay

Attributes:

duration - sets the duration of the delay.

Example:

<delay duration=”1000” />

Description:

Pauses the program for an amount of time set by the duration attribute.

Comments:

The duration value is measured in milliseconds.

Tag: linger

Attributes:

duration - sets the duration of the delay.

Example:

<linger duration=”1000” />

Description:

Pauses the program after it has executed for an amount of time set by the duration attribute

Comments:

This can only be used at the end of an execution. The duration value is measured in milliseconds

Tag: setmedia

Attributes:

send – sets to send or not to send media out.

receive – sets to receive or not to receive media in.

addr – sets the address of media.

Example:

<set media send="true" receive="true" addr=” [input.local.addr]” />

Description:

Used to set the send, receive and attribute in SDP.

Comments:

With this tag you set the SDP attribute to send and receive or send only receive only. You can also set the address of the media channel.

Tag: startmedia

Attributes:

none

Example:

<startmedia />

Description:

Used to start the media

Comments:

With this tag, you can start the media when ever you want within the call.

Tag: stopmedia

Attributes:

none

Example:

<stopmedia />

Description:

Used to stop the media

Comments:

With this tag, you can stop the media when ever you want within the call if the media is already started.

Data Elements

Tag: dictionary

Attributes:

name - the name of the dictionary variable.

file - the file to where the dictionary will be saved and loaded from.

Example:

<dictionary name=”miscellaneous” file=”[misc.dictionary]”>

</dictionary>

Description:

A basic mapping of all the variables in the file and their values

Comments:

All variables are defined and initialized in this area

Tag: set

Attributes:

var - this is the variable to be altered.

value - this is the value to which the variable will be set equal to.

Example:

<set var=”person.name.1” value=”Quincy” />

Description:

Used to set a variable to a specific value, can only define specific elements, not arrays.

Comments:

This is the only way to define a variable.

Tag: increment

Attributes:

var - this selects the variable to be incremented

increment - sets the amount by which the value shall increase (can be set to negative values)

Example:

<incr var=”num.1” increment=”2” />

Description:

Used to change a variable based on its original value by adding a set number or variable defined in the dictionary to it. May be incremented by a negative number to decrement. May not increment by an equation, only a number or variable.

Comments:

Useful for producing counters and keeping track of loops as well as other repeating sections of code. Also provides the basic set-up for addition and subtraction

Conditional Elements

Tag: if

Attributes:

var - defines the variable that is referenced by the conditional, may be a user defined variable or an actual value.

op - defines how the variable is referenced by which operand is used (“equal,” “greater,” “less,” “greater or equal,” “less or equal,” “not equal,” “contains”).

value - the value that the variable is compared to for the statement.

type - defines what variable type the variables will be compared as.

Example:

<if var=”num.1” op=”greater” value=”num.2” type=”integer”>

</if>

Description:

Used to create conditional statements that execute commands based on the values of the variables and what the user inputs. This cannot be used in a “Message” section of code.

Comments:

Useful in restricting events to certain triggers thus reducing processing power commitment.

Tag: ifnot

Attributes:

var - defines the variable that is referenced by the conditional, may be a user defined variable or an actual value.

op - defines how the variable is referenced by which operand is used (“equal,” “greater,” “less,” “greater or equal,” “less or equal,” “not equal,” “contains”).

value - the value that the variable is compared to for the statement.

type - defines what variable type the variables will be compared as.

Example:

<ifnot var=”num.1” op=”greater” value=”num.2” type=”integer”>

</ifnot>

Description:

Used to create conditional statements that execute commands based on the values of the variables and what the user inputs. This cannot be used in a “Message” section of code.

Comments:

Useful in restricting events to certain triggers thus reducing processing power commitment

Tag: conditional

Attributes:

var - defines the variable that is referenced by the conditional, may be a user defined variable or an actual value.

op - defines how the variable is referenced by which operand is used (“equal,” “greater,” “less,” “greater or equal,” “less or equal,” “not equal,” “contains”).

value - the value that the variable is compared to for the statement.

type - defines what variable type the variables will be compared as.

Example:

<conditional var=”num.1” op=”greater” value=”num.2” type=”integer”>

</conditional>

Description:

Used to create conditional statements that execute commands based on the values of the variables and what the user inputs. This can only be used in a “Message” section of code.

Comments:

Useful in restricting events to certain triggers thus reducing processing power commitment

Utility Elements

Tag: log

Attributes:

level – this is the level of the log file where the information will be recorded (“all,” “trace,” “debug,” “info,” “warning,” “error”).

text – this is what is recorded into the log file

Example:

<log level=”error” text=”There is an error type 1234, miscellaneous failure” />

Description:

Logs information dictated by the “text” attribute on the level of the log file specified by the user.

Comments:

Useful for keeping track of events and what and where exactly things are occurring in the program.

Tag: beep

Attributes:

none

Example:

<beep=”true” />

Description:

Used to beep tone.

Comments:

Useful for checking or indicating the selected code is executed or not.