XML User Directory

About

The FreeSWITCH `$FS_ROOT/conf/directory/` contains accounts (i.e., XML files) for all users (i.e., SIP phone extensions) that may register to FS.

Note: This is not the same syntax which is used in the Dialplan

User Settings

conf/directory/default/1000.xml Example

```
<include>
  <user id="1000" cidr="12.34.56.78/32,20.0.0.0/8"> <!--ID is the required SIP username. CIDR is optional -- if
specified, these IPs automatically authenticate as this user-->
    <params>
      <param name="password" value="correcthorsebatterystaple"/> <!--SIP password-->
      <param name="vm-password" value="8761"/>
    </params>
    <variables> <!--these variable are accessible in the channel-->
      <variable name="accountcode" value="1000"/> <!--Use this in your dialplan for authorization and limits.
Also, cdr_csv can use it for separate CDR files-->
      <variable name="user_context" value="default"/> <!--magic variable: specifies the context, which
translates to which dialplan will route outbound calls-->
      <variable name="effective_caller_id_name" value="Extension 1000"/> <!--magic variable: used for outbound
caller ID name-->
      <variable name="effective_caller_id_number" value="1000"/> <!--magic variable: used for outbound caller
ID name/number-->
    </variables>
  </user>
</include>
```

Note that the "user" in the above directory snippet can be called in the dialplan bridge app with the user pseudo-endpoint.

```
<action application="bridge" data="user/1000"/>
```

Basic User

The basic configuration is quite simple, you just need to add user names and passwords.

In the file `$FS_ROOT/conf/directory/mike_x2239.xml`, for example, add:

```
<domain name="$${sip_profile}">
  <user id="mike">
    <params>
      <param name="password" value="micke"/>
    </params>
  </user>
</domain>
```

The domain tag tells FS to which domain this user belongs. Note that all users should be in the same domain tag, unless you are using multiple domains. The user name is the part left of @ in the sip address (in this case "mike" in "mike@sub.mydomain.com"). $${sip_profile} will be replaced with the domain that is specified in vars.xml.

a1-hash

A plaintext password may replaced with an "a1-hash". An a1-hash is generated by applying the MD5 digest function to the string "`username:domain:password`" (sans quotes). In Linux, the encoded password may be generated thusly:

```
openssl dgst -md5 < filename, or echo -n "username:domain:password" | openssl dgst -md5
```

conf/directory/mike_x2239.xml

```
<domain name="$${sip_profile}">
  <user id="mike">
    <params>
      <param name="a1-hash" value="c6440e5de50b403206989679159de89a"/>
    </params>
  </user>
</domain>
```

Reverse Authentication

Some endpoints require authentication for certain types of requests (ex. SIP NOTIFY for resync). You can specify the credentials used for this digest authentication.

conf/directory/default/jim_ra.xml

```
<domain name="$${sip_profile}">
  <user id="jim">
    <params>
       <param name="reverse-auth-user" value="jim" />
       <param name="reverse-auth-pass" value="foo123" />
    </params>
  </user>
</domain>
```

VCards

You can also add support for vcards if you are using mod_dingaling. Then you add information in the following format:

conf/directory/default/peters_dingaling.xml

```
<domain name="$${sip_profile}">
  <user id="peter">
    <params>
      <param name="password" value="thepassword"/>
    </params>

    <!-- This is only for mod_dingaling so it can deliver vcards in component mode-->
    <vcard xmlns='vcard-temp'>
      <FN>Peter Saint-Andre</FN>
      <N>
        <FAMILY>Saint-Andre</FAMILY>
        <GIVEN>Peter</GIVEN>
        <MIDDLE/>
      </N>
      <NICKNAME>stpeter</NICKNAME>
      <URL>http://www.jabber.org/people/stpeter.php</URL>
      <BDAY>1966-08-06</BDAY>
      <ORG>
        <ORGNAME>Jabber Software Foundation</ORGNAME>
        <ORGUNIT>Jabber Software Foundation</ORGUNIT>
      </ORG>
      <TITLE>Executive Director</TITLE>
      <ROLE>Patron Saint</ROLE>
      <TEL><WORK/><VOICE/><NUMBER>303-308-3282</NUMBER></TEL>
      <TEL><WORK/><FAX/><NUMBER/></TEL>
      <TEL><WORK/><MSG/><NUMBER/></TEL>
      <ADR>
        <WORK/>
        <EXTADD>Suite 600</EXTADD>
        <STREET>1899 Wynkoop Street</STREET>
        <LOCALITY>Denver</LOCALITY>
        <REGION>CO</REGION>
        <PCODE>80202</PCODE>
        <CTRY>USA</CTRY>
      </ADR>
      <TEL><HOME/><VOICE/><NUMBER>303-555-1212</NUMBER></TEL>
      <TEL><HOME/><FAX/><NUMBER/></TEL>
      <TEL><HOME/><MSG/><NUMBER/></TEL>
      <ADR>
        <HOME/>
        <EXTADD/>
        <STREET/>
        <LOCALITY>Denver</LOCALITY>
        <REGION>CO</REGION>
        <PCODE>80209</PCODE>
        <CTRY>USA</CTRY>
      </ADR>
      <EMAIL><INTERNET/><PREF/><USERID>stpeter@jabber.org</USERID></EMAIL>
      <JABBERID>stpeter@jabber.org</JABBERID>
      <DESC>
        More information about me is located on my
        personal website: http://www.saint-andre.com/
      </DESC>
    </vcard>

  </user>
</domain>
```

Groups

A group is a logical collection of users that FreeSWITCH can use to bridge calls in a serial or parallel fashion, depending on the arguments to the group call application. Using groups is optional -- you can put your users straight into the domain section if you desire.

This is specially useful if you use mod_xml_curl to provide the user directory to FreeSWITCH and want to group some users in a logical structure. The following group '200' gathers four users. Note the "dial-string" param, which is used to bridge the calls to those users. Users 1000 and 1001 will use the default "dial-string" while user 2014 uses a loopback channel so FreeSWITCH can actually query the dialplan to figure out how to reach that user (this also works for external numbers through OpenZAP and gateways):

The `type="pointer"` permits the same user to appear in multiple groups. It basically means to keep searching for the user in the directory.

```
<!-- Note: this example starts at the root of the FreeSWITCH XML tree -->
<document type="freeswitch/xml">
 <section name="directory">
   <domain name="sip.example.com">
     <users>
       <user id="1000">
         <params>
           <param name="dial-string" value="{presence_id=${dialed_user}@${dialed_domain}}${sofia_contact
(${dialed_user}@${dialed_domain})}"/>
         </params>
         <variables>
           <variable name="user_context" value="default"/>
         </variables>
       </user>
       <user id="1001">
         <params>
           <param name="dial-string" value="{presence_id=${dialed_user}@${dialed_domain}}${sofia_contact
(${dialed_user}@${dialed_domain})}"/>
         </params>
         <variables>
           <variable name="user_context" value="default"/>
         </variables>
       </user>
     </users>
     <groups>
       <group name="200">
         <users>
           <user id="2014">
             <params>
               <param name="dial-string" value="loopback/2014/default/XML"/>
             </params>
             <variables>
               <variable name="user_context" value="default"/>
             </variables>
           </user>
           <user id="1000" type="pointer"/>
           <user id="1001" type="pointer"/>
         </users>
       </group>
     </groups>
   </domain>
 </section>
</document>
```

The dialplan for the above group can be defined like this:

```
     <extension name="Group 200">
       <condition field="destination_number" expression="200">
         <action application="set" data="hangup_after_bridge=true"/>
         <action application="set" data="continue_on_fail=true"/>
         <action application="set" data="originate_continue_on_timeout=true"/>
         <action application="set" data="call_timeout=15"/>
         <action application="bridge" data="${group_call(200@${domain_name}+F)}"/>
         <action application="transfer" data="200 XML default"/>
         <action application="hangup"/>
       </condition>
     </extension>
```

The extension 200 will ring all the users defined in the user directory for group 200 in a serial fashion (specified by the +F argument, if you want to ring all the users at once use the +A argument) for 15 seconds, then it will transfer the call to the same group again so the call will ring the group infinitely.

To explain the variables set prior to the bridge:

- The hangup_after_bridge is set to true for this effect: if the bridge is successfully answered and the B-leg later hangs up, the A-leg will also be hung up.
- The continue_on_fail is set to true for this: if the bridge fails, dialplan execution will continue and the transfer will be executed.
- The originate_continue_on_timeout is set to true for this: if the bridge's dial string specifies several destinations separated by the "|" (this is for failover), the bridge will time out on an unanswered destination and will attempt the next specified destination. Without originate_continue_on_timeout set to true, the bridge will time out on the first destination it tries and the bridge itself will fail. (In the example above, the bridge string is generated by group_call with the +F option; this specifies a dial string with all the group's destinations separated by "|". So originate_continue_on_timeout needs to be set to true for serial calling behavior.)
- The call_timeout is set so that the bridge attempt to a destination that goes unanswered will time out. NOTE: If the destination sends early media, the bridge will be answered (pre-answered) and will NOT time out! To time out a bridge attempt to a destination sending early media, set ignore_early_media to true.

The dialplan for user 2014, which in this example happens to be terminated through a gateway defined via mod_lcr, can be defined like this:

```
    <extension name="Extension 2014">
      <condition field="destination_number" expression="2014">
        <action application="lcr" data="2014"/>
        <action application="set" data="dialed_ext=${lcr_auto_route}"/>
        <action application="export" data="dialed_ext=${lcr_auto_route}"/>
        <action application="set" data="hangup_after_bridge=true"/>
        <action application="set" data="call_timeout=120"/>
        <action application="bridge" data="${lcr_auto_route}"/>
        <action application="hangup"/>
      </condition>
    </extension>
```

Alphanumeric to Numeric User Mapping

Say you want a user's id to be alphanumeric (like an email username), such as `johnsmith@pbx.example.com` These users have alphanumeric usernames in their sip phone config, but you want to map them from their sip username to a numeric extension, and vice versa.

As of version 1.0.4, FreeSWITCH makes this trivial to accomplish. A user's ID can be any alphanumeric string, and this can be simply tied to an extension number using the 'number-alias' property. This property creates an aliased directory entry that points to the alphanumeric user entry.

Be sure to edit your dialplan/default.xml file to include not only the user id (johnsmith in our example), but also the number-alias (1001). Keep in mind that user IDs are case sensitive.

When using this attribute, you must be careful not to create a directory collision by having another user whose ID is the same as another user's alias

Here is an example from the user directory:

number-alias Example

```
  <user id="johnsmith" number-alias="1001">
    <!-- Insert the usual user configuration variables and params here,
         including user password, voicemail password, caller ID info, etc -->
     <variables>
       <variable name="mailbox" value="1001"/>
       <variable name="effective_caller_id_name" value="1001"/>
       <variable name="effective_caller_id_number" value="1001"/>
       <variable name="voicemail_alternate_greet_id" value="1001"/>
     </variables>
  </user>
```

So when a user dials extension number 1001, your dialplan can use the 'user_data' function to look up the ID attribute associated with that number alias. In the default dialplan, the 'Local Extension' section can be made to work with a small change to the 'bridge' line:

```
  <action application="bridge" data="user/${user_data(${destination_number}@${domain_name} attr id)}"/>
```

⚠ Using this `user_data` function in combination with `mod_xml_curl` will generate an additional request each time the `user_data` function is called. Note that it is already called once in the Local Extension section of the default vanilla dialplan to determine the callgroup. **Beware of the performance implications of this with high-volume systems**.

More Complex Examples

Each <user> can also have it's own variables and gateways with their own special configurations.

User-Specific Gateways

```
<user id="user1">
  <params>
    <param name="password" value="1"/>
  </params>
  <variables>
    <variable name="register-gateway" value="user1out"/>
  </variables>
  <gateways>
    <gateway name="user1out">
      <param name="username" value="4347382173"/>
      <param name="password" value="1"/>
      <param name="proxy" value="sip.example.com"/>
      <param name="register" value="false"/>
    </gateway>
  </gateways>
</user>
```

The <register-gateway> variable can be set to the name of a specific gateway, a comma delimited list of multiple gateways, or "all". Setting it to one or more gateways will register the named gateway(s) when the <user> registers with FreeSWITCH.(whether they're in this user's <gateways> or some other user's <gateways> or anywhere). Setting the variable to "all" will register all of the particular user's gateways.

Group Call with Answer Confirmation

The following example makes the user reachable at multiple phone numbers in SIP and PSTN networks. A call to a mobile phone may usually be answered with a message that a mobile user is unreachable, or end up in a voicemail. In order to avoid such situations, group_confirm_key is used to request the user to press "1" for call confirmation.

Loopback endpoint duplicates all channel variables, so if group_confirm_key is set globally, the confirmation key would be asked twice. So it needs to be applied with [] to limit to one leg only. The loopback endpoint is used to route the call to a PSTN number, assuming that users in the default context are allowed to dial international numbers and they are routed to an appropriate gateway.

```
<!-- dialplan/default.xml -->
   <extension name="global" continue="true">
     <condition>
       <action application="set" data="group_confirm_file=phrase:press_one_to_answer"/>
       <action application="set" data="group_confirm_read_timeout=1000"/>
     </condition>
   </extension>
   <extension name="dvop_groupcall">
     <condition field="destination_number" expression="^(71[1-9]0)$">
       <action application="answer"/>
       <action application="set" data="ringback=$${hold_music}"/>
       <action application="set" data="call_timeout=60"/>
       <action application="set" data="hangup_after_bridge=true"/>
       <action application="bridge" data="${group_call(hunt_$1@${domain_name}+A)}"/>
     </condition>
   </extension>

 <!-- directory/default.xml -->
    <users>
      <user id="7012">
        <params>
          <param name="a1-hash" value="538db5a1dcf95cd9df62bf2ff0466c4b"/>
        </params>
        <variables>
          <variable name="user_context" value="default"/>
        </variables>
      </user>
      <user id="7017">
        <params>
          <param name="dial-string" value="[group_confirm_key=1]loopback/00491637743380/default"/>
        </params>
      </user>
    </users>
    <groups>
      <group name="hunt_7190">
        <users>
          <user id="7012" type="pointer"/>
          <user id="7017" type="pointer"/>
        </users>
      </group>
    </groups>
```

Domains & Users Parameters

<params> and <variables> tags are valid inside <user>, <group> and <domain> tags.

Parameters and variables set on the domain will apply to all users in the domain, and parameters and variables set in a group will apply to all users in the group.

Priority will be given to identical parameters and variables in the following order: users, groups, domain.

Password

```
<param name="password" value="123456"/>
```

Do Not Allow Empty Passwords

If you do not include the above 'password' parameter, anybody can register as the user without using a password. It is always wise to include this parameter in the directory section in case a user does not set their password.

```
<param name="allow-empty-password" value="false"/>
```

Prevent SIP registration

You can prevent a particular user from registering over SIP with the sip-forbid-register parameter:

```
<param name="sip-forbid-register" value="true"/>
```

Dial String

The dial string MUST be defined and will control the behavior of the call when a user is dialed. The dial-string parameter is used by the user/ endpoint.

Default value goes as follows:

```
<param name="dial-string" value="${sofia_contact(${dialed_user}@${dialed_domain})}"/>
```

Channel Variables Useful for the Dial String

- transfer_fallback_extension
- presence_id

```
<param name="dial-string" value="{transfer_fallback_extension=${dialed_user}}${sofia_contact(${dialed_user}
@${dialed_domain})}"/>
```

Include Pickup Endpoints

```
<param name="dial-string" value="${sofia_contact(${dialed_user}@${dialed_domain})},pickup/${dialed_user}
@${dialed_domain}"/>
```

Advanced Dial String

- Sets presence and creates pickup endpoint

```
<param name="dial-string" value="{sip_invite_domain=${dialed_domain},presence_id=${dialed_user}
@${dialed_domain}}${sofia_contact(${dialed_user}@${dialed_domain})},pickup/${dialed_user}@${dialed_domain}"/>
```

Variables

Any variables defined in the domain or user will be defined as channel variables when there is a call to that user or when there is a call initiated by that user. These variables can be read in the dialplan and can affect the handling of the call.

Forcing a Particular User to a Particular Extension

In a PABX environment, an authenticated user can specify that they are at an arbitrary extension. This behavior can be restricted in two ways.

To force a user to use a specified extension, add

```
<variable name="sip-force-user" value="<extension>"/>
```

to that user's directory entry.

Alternatively, to check that users authenticate with the same username as that in their contact field for an entire profile, add

```
<variable name="inbound-reg-force-matching-username" value="true"/>
```

to that profile's definition.

Additionally look at JIRA ticket FS-5119 - easy way of registration user with different SIP ID (Contact header) and username (Authorization header).

Force Registrations to Expire

To help prevent stale registrations you are able to override the client–specified registration expiry. You do this in the client's directory profile, by simply adding the variable

```
<variable name="sip-force-expires" value="180"/>
```