navneetgupta / **Using_Custom_XMPP_stanza_in_Ejabberd.md**

Last active 2 years ago • Report abuse

Star

Code   Revisions 16

Create Custom Ejabberd XMPP elements as well as tools to work with them.

ejabberd_new_stanza_record.md

Creating new Custom XMPP elements/ Codecs steps:

- Clone the https://github.com/processone/xmpp.

As Per Repository Readme.md

```
All XMPP elements (records) are defined in include/xmpp_codec.hrl
file.
For convenience, every record has the corresponding type spec.
There is also predefined xmpp_element() type which is a container for
all defined record types:
so sometimes we will refer to an arbitrary XMPP element as
xmpp_element() in the rest of this document.
These records are generated automatically by XML generator from
specification file specs/xmpp_codec.spec.
The specification file contains information about XML elements
defined within XMPP related namespace.
```

Adding Custom XMPP elements.

- add the required spec in `xmpp_codec.spec` . **refer other elements in same file for how to write the spec**.
- run `make specs` to generate the related modules and tools for processing the element.

General representation of an XMPP xml Element

```
 <element_name xmlns="namespace" attr1_name="attr1_value"
attr2_name="attr2_value">{...subEls}</element_name>
```

Few important parts of Spec:

```
-xml($name_of_spec,
      #elem{name = $element_name,
            xmlns = $namespace ,
            module = $module_name,
            result = {$recordname, '$xmlns', '$attr1_name', , '$attr2_na
            attrs = [#attr{name = <<"xmlns">>},
                     #attr{name = <<"attr1_name">>, required = true},
                     #attr{name = <<"attr2_name">>}]
      }).
```

Replace

$**name_of_spec** -- should be atom, spec_name can be used in nested spec.
$**element_name** -- should be binary, it would be the name of custom element created.
$**module_name** -- this will be the module where related processing tools for the element will be created.
$**recordname** -- is optional, If exists generally same as $name_of_spec. Record created will be of this name.
$**namespace** -- could be single binary namespace or list of binary namespaces this xmpp element supports.

`$_els` => represent the xmpp element support for child elements and will be represented in record by key `sub_els` of list type. -- optional put this only if xmpp element could have other child elelemnt.

Above spec will generate below items:

1. $module_name.erl file in `src` folder, and contain related tools for processing to/from record/xml.
2. $recordname record and corresponiding related type defination in `xmpp_codec.hrl` .

Example:

```
-xml(message_custom,
      #elem{name = <<"message_c">>,
            xmlns = <<"urn:ns:message_c:0">> ,
            module = message_custom_module,
            result = {message_custom, '$xmlns', '$to', , '$from', '$text
            attrs = [#attr{name = <<"xmlns">>},
                     #attr{name = <<"to">>, required = true},
                     #attr{name = <<"from">>},
                     #attr{name = <<"from">>, required = true}]
      }).
```

Above spec will generate:

-- module `src/message_custom_module.erl`.
-- Below entry in `include/xmpp_codec.erl`.

```
-record(message_custom, {xmlns = <<>> :: binary(),
                         to = <<>> :: binary(),
                         from = <<>> :: binary(),
                         text = <<>> :: binary()}).
-type message_custom() :: #message_custom{}.
```

-- and Will represent xmpp Element

```
<message_c xmlns="urn:ns:message_c:0" to="to@domain"
from="from@domain" text="some text" />
```

TODO:

1. **Above is minimal steps to create a custom xmpp element.**
2. **More attributes validation can be added to attrs section of each attr element. Seed JID attribute validation**

```
-xml(message_custom,
      #elem{name = <<"message_c">>,
            xmlns = <<"urn:ns:message_c:0">> ,
            module = message_custom_module,
            result = {message_custom, '$xmlns', '$to', , '$from',
'$text'},
            attrs = [#attr{name = <<"xmlns">>},
                     #attr{name = <<"to">>,
                           required = true,
                           dec = {jid, decode, []},
                           enc = {jid, encode, []}},
                     #attr{name = <<"text">>, required = true},
                     #attr{name = <<"from">>,
                           required = true,
                           dec = {jid, decode, []},
                           enc = {jid, encode, []}}]
      }).
```

3. **Form-Fields xdata can be customized uisng corresponding files `$form_name.cfg` and `$form_name.xdata` in `specs` folder.**

4. **More Complex nested structure can be create. Follow `pubsub` (search `-xml(pubsub, )` spec for more complex example in the `xmpp_codec.spec` file.**

Using_Custom_XMPP_stanza_in_Ejabberd.md

For Using Above created Custom xmpp element :

Two ways:

1. Modify Dependencies of ejabberd to use your cloned xmpp repositories instead of provided one.
2. Let's say u want to use those Custom xmpp Element in your module named my_custom_module.
   a. Add the generated files/modules `(src/message_custom_module.erl)` from above example to your ejabberd src directory.
   b. copy the `type and record defination` generated in `xmpp_codec.hrl` (use git diff to get the highlighted changes.) and put it in your `include/my_custom_module.hrl` (naming as per convention, could be anything else also, but as per best procatices.) in the ejabberd repository.
   c. In Your custom Module (my_custom_module.erl) include the `-include("my_custom_module.hrl")`.
   d. Register the codecs module generated `src/message_custom_module.erl` in the module.
   i. To Register , In `start(Host,Opts)` of module add `xmpp:register_codec(message_custom_module)`
   ii. To unregister, In `stop(Host)` of module add `xmpp:unregister_codec(message_custom_module)`.

Example:

1. Add `message_custom_module.erl` file generated in `xmpp/src` directory to `ejabberd/src`.

2. *ejabberd/include/my_custom_module.hrl*

```
%% paste type and record definitions from xmpp_codec.hrl

-record(message_custom, {xmlns = <<>> :: binary(),
                         to = <<>> :: binary(),
                         from = <<>> :: binary(),
                         text = <<>> :: binary()}).
-type message_custom() :: #message_custom{}.
```

3. *ejabberd/src/my_custom_module.erl*

```
-module(my_custom_module).
-behaviour(gen_mod).

-include("my_custom_module.hrl").

-export([start/2, stop/1, reload/3, depends/2, mod_options/1]).

start(_Host, _Opts) ->
  xmpp:register_codec(message_custom_module),
  %% Other Configuration as per your module requirement
  ok.
stop(_Host) ->
  xmpp:register_codec(message_custom_module),
  %% Other Configuration as per your module requirement
  ok.

reload(Host, NewOpts, OldOpts) -> ok.

depends(_Host, _Opts) ->
  [].

mod_options(_Host) -> [].

 %% Add Other handlers, hooks , Api Calls, Callbacks etc.
```