usrloc Module

Jan Janak
FhG FOKUS

Edited by

Jan Janak

usrloc Module

Edited by and Jan Janakand Jan Janak

Revision History

Revision $Revision: 1.1.2.1 $ $Date: 2003/08/05 21:48:08 $

Table of Contents

List of Examples

Chapter 1. User’s Guide

1.1. Overview

User location module. The module keeps a user location table and provides access to the table to other modules. The module exports no functions that could be used directly from scripts.

1.2. Dependencies

1.2.1. SER Modules

The following modules must be loaded before this module:

- *Optionally a database module*.

1.2.2. External Libraries or Applications

The following libraries or applications must be installed before running SER with this module loaded:

- *None*.

1.3. Exported Parameters

1.3.1. `user_column` (string)

Name of column containing usernames.

Default value is “username”.

Example 1-1. Set `user_column` parameter

```
...
modparam("usrloc", "user_column", "username")
...
```

1.3.2. `contact_column` (string)

Name of column containing contacts.

Default value is "contact".

Example 1-2. Set `contact_column` parameter

```
...
modparam("usrloc", "contact_column", "contact")
...
```

1.3.3. `expires_column` (string)

Name of column containing expires value.

Default value is "expires".

Example 1-3. Set `expires_column` parameter

```
...
modparam("usrloc", "expires_column", "expires")
...
```

1.3.4. `q_column` (string)

Name of column containing q values.

Default value is "q".

Example 1-4. Set `q_column` parameter

```
...
modparam("usrloc", "q_column", "q")
...
```

1.3.5. `callid_column` (string)

Name of column containing callid values.

Default value is "callid".

Example 1-5. Set `callid_column` parameter

```
...
modparam("usrloc", "callid_column", "callid")
...
```

1.3.6. `cseq_column` (string)

Name of column containing cseq numbers.

Default value is "cseq".

Example 1-6. Set `cseq_column` parameter

```
...
modparam("usrloc", "cseq_column", "cseq")
...
```

1.3.7. `method_column` (string)

Name of column containing supported methods.

Default value is "method".

Example 1-7. Set `method_column` parameter

```
...
modparam("usrloc", "method_column", "method")
...
```

1.3.8. `db_url` (string)

URL of the database that should be used.

Default value is "sql://ser:heslo@localhost/ser".

Example 1-8. Set `db_url` parameter

```
...
modparam("usrloc", "db_url", "sql://username:password@localhost/ser")
...
```

1.3.9. `timer_interval` (integer)

Number of seconds between two timer runs. The module uses timer to delete expired contacts, synchronize with database and other tasks, that need to be run periodically.

Default value is 60.

Example 1-9. Set `timer_interval` parameter

```
...
modparam("usrloc", "timer_interval", 120)
...
```

1.3.10. `db_mode` (integer)

The usrloc module can utilize database for persistent contact storage. If you use database, your contacts will survive machine restarts or sw crashes. The disadvantage is that accessing database can be very time consuming. Therefore, usrloc module implements three database accessing modes:

- 0 - This disables database completely. Only memory will be used. Contacts will not survive restart. Use this value if you need a really fast usrloc and contact persistence is not necessarry or is provided by other means.
- 1 - Write-Through scheme. All changes to usrloc are immediately reflected in database too. This is very slow, but very reliable. Use this scheme if speed is not your priority but need to make sure that no registered contacts will be lost during crash or reboot.
- 2 - Write-Back scheme. This is a combination of previous two schemes. All changes are made to memory and database synchronization is done in the timer. The timer deletes all expired contacts and flushes all modified or new contacts to database. Use this scheme if you encounter high-load peaks and want them to process as fast as possible. The mode will not help at all if the load is high all the time. Also, latency of this mode is much lower than latency of mode 1, but slightly higher than latency of mode 0.

> **Warning**
>
> In case of crash or restart contacts that are in memory only and haven't been flushed yet will get lost. If you want minimize the risk, use shorter timer interval.

Default value is 0.

Example 1-10. Set `db_mode` parameter

```
...
modparam("usrloc", "db_mode", 2)
...
```

1.4. Exported Functions

There are no exported functions that could be used in scripts.

Chapter 2. Developer’s Guide

2.1. Available Functions

2.1.1. `ul_register_domain(name)`

The function registers a new domain. Domain is just another name for table used in registrar. The function is called from fixups in registrar. It gets name of the domain as a parameter and returns pointer to a new domain structure. The fixup than ’fixes’ the parametr in registrar so that it will pass the pointer instead of the name every time save() or lookup() is called. Some usrloc functions get the pointer as parameter when called. For more details see implementation of save function in registrar.

Meaning of the parameters is as follows:

- *const char* name* - Name of the domain (also called table) to be registered.

2.1.2. `ul_insert_urecord(domain, aor, rec)`

The function creates a new record structure and inserts it in the specified domain. The record is structure that contains all the contacts for belonging to the specified username.

Meaning of the parameters is as follows:

- *udomain_t* domain* - Pointer to domain returned by ul_register_udomain.
- *str* aor* - Address of Record (aka username) of the new record (at this time the record will contain no contacts yet).
- *urecord_t** rec* - The newly created record structure.

2.1.3. `ul_delete_urecord(domain, aor)`

The function deletes all the contacts bound with the given Address Of Record.

Meaning of the parameters is as follows:

- *udomain_t* domain* - Pointer to domain returned by ul_register_udomain.
- *str* aor* - Address of record (aka username) of the record, that should be deleted.

2.1.4. ul_get_urecord(domain, aor)

The function returns pointer to record with given Address of Record.

Meaning of the parameters is as follows:

- *udomain_t* domain* - Pointer to domain returned by ul_register_udomain.
- *str* aor* - Address of Record of request record.

2.1.5. ul_lock_udomain(domain)

The function lock the specified domain, it means, that no other processes will be able to access during the time. This prevents race conditions. Scope of the lock is the specified domain, that means, that multiple domain can be accessed simultaneously, they don't block each other.

Meaning of the parameters is as follows:

- *udomain_t* domain* - Domain to be locked.

2.1.6. ul_unlock_udomain(domain)

Unlock the specified domain previously locked by ul_lock_udomain.

Meaning of the parameters is as follows:

- *udomain_t* domain* - Domain to be unlocked.

2.1.7. ul_release_urecord(record)

Do some sanity checks - if all contacts have been removed, delete the entire record structure.

Meaning of the parameters is as follows:

- *urecord_t* record* - Record to be released.

2.1.8. ul_insert_ucontact(record, contact, expires, q, callid, cseq, cont)

The function inserts a new contact in the given record with specified parameters.

Meaning of the parameters is as follows:

- *urecord_t* record* - Record in which the contact should be inserted.
- *str* contact* - Contact URI.
- *time_t expires* - Expires of the contact in absolute value.
- *float q* - q value of the contact.
- *str* callid* - Call-ID of the REGISTER message that contained the contact.
- *int cseq* - CSeq of the REGISTER message that contained the contact.
- *ucontact_t* cont* - Pointer to newly created structure.

2.1.9. `ul_delete_ucontact(record, contact)`

The function deletes given contact from record.

Meaning of the parameters is as follows:

- *urecord_t* record* - Record from which the contact should be removed.
- *ucontact_t* contact* - Contact to be deleted.

2.1.10. `ul_get_ucontact(record, contact)`

The function tries to find contact with given Contact URI and returns pointer to structure representing the contact.

Meaning of the parameters is as follows:

- *urecord_t* record* - Record to be searched for the contact.
- *str_t* contact* - URI of the request contact.

2.1.11. `ul_get_all_ucontacts(buf, len)`

The function retrieves all contacts of all registered users and returns them in the caller-supplied buffer. If the buffer is too small, the function returns positive value indicating how much additional space would be necessary to accomodate all of them. Please note that the positive return value should be used only as a "hint", as there is no gurantee that during the time between two subsequent calls number of registered contacts will remain the same.

Meaning of the parameters is as follows:

- *void* buf* - Buffer for returning contacts.
- *int len* - Length of the buffer.

2.1.12. `ul_update_ucontact(contact, expires, q, callid, cseq)`

The function updates contact with new values.

Meaning of the parameters is as follows:

- *ucontact_t* contact* - Contact URI.
- *time_t expires* - Expires of the contact in absolute value.
- *float q* - q value of the contact.
- *str* callid* - Call-ID of the REGISTER message that contained the contact.
- *int cseq* - CSeq of the REGISTER message that contained the contact.

Chapter 3. Frequently Asked Questions

1. Where can I find more about SER?

Take a look at http://iptel.org/ser.

2. Where can I post a question about this module?

First at all check if your question was already answered on one of our mailing lists:

- http://mail.iptel.org/mailman/listinfo/serusers
- http://mail.iptel.org/mailman/listinfo/serdev

E-mails regarding any stable version should be sent to <`serusers@iptel.org`> and e-mail regarding development versions or CVS snapshots should be send to <`serdev@iptel.org`>.

If you want to keep the mail private, send it to <`serhelp@iptel.org`>.

3. How can I report a bug?

Please follow the guidelines provided at: http://iptel.org/ser/bugs