EUDC Manual

The Emacs Unified Directory Client

by Oscar Figueiredo

`1.30b`

1 Overview

EUDC, the *Emacs Unified Directory Client*, provides a common user interface to access directory servers using different directory protocols.

Currently supported back-ends are:

- LDAP, Lightweight Directory Access Protocol
- CCSO PH/QI
- BBDB, Big Brother's Insiduous Database

The main features of the EUDC interface are:

- Queries using a customizable form
- Inline query expansion (for instance you can expand a name to an email address in a mail message buffer using a server as an address book)
- Multiple servers can be tried in turn until a match is found for an inline query
- Fast minibuffer queries for email addresses and phone numbers
- Interface to BBDB to let you insert server records into your own BBDB database (see section "BBDB" in *BBDB Manual*)

1.1 LDAP

LDAP, *the Lightweight Directory Access Protocol*, is a communication protocol for directory applications defined in RFC 1777.

Quoted from RFC 1777:

> [LDAP] is designed to provide access to the X.500 Directory while not incurring the resource requirements of the Directory Access Protocol (DAP). This protocol is specifically targeted at simple management applications and browser applications that provide simple read/write interactive access to the X.500 Directory, and is intended to be a complement to the DAP itself.

LDAP servers usually store (but are not limited to) information about people such as their name, phone number, email address, office location, etc. . . . More information about LDAP can be found at `http://www.openldap.org/`

EUDC requires external support to access LDAP directory servers (see Section 2.1 [LDAP Requirements], page 3)

1.2 CCSO PH/QI

The Central Computing Services Office (CCSO) of the University of Illinois at Urbana Champaign (UIUC) created and freely distributes a directory system that is currently in use in more than 300 organizations around the world. The system records information about people such as their address, phone number, email, academic information or any other details it was configured to.

The system consists of two parts: a database server traditionally called '`qi`' and a command-line client called '`ph`'. `ftp://uiarchive.cso.uiuc.edu/pub/packages/ph` is the main distribution site. `http://www.uiuc.edu/cgi-bin/ph/lookup?Query=.` provides a listing of the active '`qi`' servers.

The original command-line '`ph`' client that comes with the '`ph/qi`' distribution provides additional features like the possibility to communicate with the server in login-mode which makes it possible to change records in the database. This is not implemented in EUDC.

1.3 BBDB

BBDB is the *Big Brother's Insiduous Database*, a package for Emacs originally written by Jamie Zawinski which provides rolodex-like database functionality featuring tight integration with the Emacs mail and news readers.

It is often used as an enhanced email address book.

EUDC considers BBDB as a directory server back end just like LDAP or PH/QI servers, though BBDB has no client/server protocol and thus always resides locally on your machine. The point in this is not to offer an alternate way to query your BBDB database (BBDB itself provides much more flexible ways to do that), but rather to offer an interface to your local directory that is consistent with the interface to external directories (LDAP, PH/QI). This is particularly interesting when performing queries on multiple servers.

EUDC also offers a means to insert results from directory queries into your own local BBDB (see Section 3.7 [Creating BBDB Records], page 10)

2 Installation

Add the following to your '.emacs' init file:

```
(require 'eudc)
```

This will install EUDC at startup.

After installing EUDC you will find (the next time you launch Emacs) a new `Directory Search` submenu in the '`Tools`' menu that will give you access to EUDC.

You may also find it useful to add the following to your '`.emacs`' initialization file to add a shortcut for email address expansion in email composition buffers (see Section 3.4 [Inline Query Expansion], page 7)

```
(eval-after-load
 "message"
 '(define-key message-mode-map [(control ?c) (tab)] 'eudc-expand-inline))
(eval-after-load
 "sendmail"
 '(define-key mail-mode-map [(control ?c) (tab)] 'eudc-expand-inline))
```

2.1 LDAP Requirements

LDAP support is added by means of '`ldap.el`' which is part of Emacs. '`ldap.el`' needs an external command line utility named '`ldapsearch`' which is available as part of LDAP toolkits:

- Open LDAP Libraries (`http://www.openldap.org/`)
- University of Michigan's LDAP Client software (`http://www.umich.edu/~dirsvcs/ldap/`)

3 Usage

This chapter describes the usage of EUDC. Most functions and customization options are available through the '`Directory Search`' submenu of the '`Tools`' submenu.

3.1 Querying Servers

EUDC's basic functionality is to let you query a directory server and return the results back to you. There are several things you may want to customize in this process.

3.1.1 Selecting a Server

Before doing any query you will need to set the directory server. You need to specify the name of the host machine running the server software and the protocol to use. If you do not set the server in any fashion, EUDC will ask you for one when you make your first query.

You can set the server by selecting one from your hotlist of servers (see Section 3.5 [The Server Hotlist], page 8) available in the '`Server`' submenu or by selecting '`New Server`' in that same menu.

LDAP servers generally require some configuration before you can perform queries on them. In particular, the *search base* must be configured. If the server you select has no configured search base then EUDC will propose you to configure it at this point. A customization buffer will be displayed where you can edit the search base and other parameters for the server.

eudc-server Variable

The name or IP address of the remote directory server. A TCP port number may be specified by appending a colon and a number to the name of the server. You will not need this unless your server runs on a port other than the default (which depends on the protocol). If the directory server resides on your own computer (which is the case if you use the BBDB back end) then 'localhost' is a reasonable value but it will be ignored anyway.

eudc-protocol Variable

The directory protocol to use to query the server. Currently supported protocols in this version of EUDC are `ph`, `ldap` and `bbdb`.

eudc-set-server Command

This command accessible from '`New Server`' submenu lets you specify a new directory server and protocol.

3.1.2 Return Attributes

Directory servers may be configured to return a default set of attributes for each record matching a query if the query specifies none. The variable `eudc-default-return-attributes` controls the return attributes you want to see, if different from the server defaults.

eudc-default-return-attributes Variable

A list of the default attributes to extract from directory entries. If set to the symbol `all` then all available attributes are returned. A value of `nil`, the default, means to return the default attributes as configured in the server.

The server may return several matching records to a query. Some of the records may however not contain all the attributes you requested. You can discard those records.

eudc-strict-return-matches User Option

If non-`nil`, entries that do not contain all the requested return attributes are ignored. Default is `t`.

3.1.3 Duplicate Attributes

Directory standards may authorize different instances of the same attribute in a record. For instance the record of a person may contain several email fields containing different email addresses. When using a QI directory server this is difficult to distinguish from attributes having multi-line values such as the postal address that may contain a line for the street and another one for the zip code and city name. In both cases, EUDC will consider the attribute duplicated.

EUDC has several methods to deal with duplicated attributes. The available methods are:

`list` Makes a list with the different values of the duplicate attribute. The record is returned with only one instance of the attribute with a list of all the different values as a value. This is the default method that is used to handle duplicate fields for which no other method has been specified.

`first` Discards all the duplicate values of the field keeping only the first one.

`concat` Concatenates the different values using a newline as a separator. The record keeps only one instance of the field the value of which is a single multi-line string.

`duplicate`

Duplicates the whole record into as many instances as there are different values for the field. This is the default for the email field. Thus a record containing 3 different email addresses is duplicated into three different records each having a single email address. This is particularly useful in combination with `select` as the method to handle multiple matches in inline expansion queries (see Section 3.4 [Inline Query Expansion], page 7) because you are presented with the 3 addresses in a selection buffer

Because a method may not be applicable to all fields, the variable `eudc-duplicate-attribute-handling-method` lets you specify either a default method for all fields or a method for each individual field.

eudc-duplicate-attribute-handling-method Variable

A method to handle entries containing duplicate attributes. This is either an alist of elements `(`*attr* `.` *method*`)`, or a symbol *method*. The alist form of the variable associates a method to an individual attribute name; the second form specifies a method applicable to all attribute names. Available methods are: `list`, `first`, `concat`, and `duplicate` (see above). The default is `list`.

3.2 Query Form

The simplest way to query your directory server is to use the query form. You display the query form with the ‘`Query with Form`’ menu item or by invoking the command *`M-x eudc-query-form`*. The attribute names presented in this form are defined by the `eudc-query-form-attributes` variable (unless a non-`nil` argument is supplied to `eudc-query-form`).

Since the different directory protocols to which EUDC interfaces may use different names for equivalent attributes, EUDC defines its own set of attribute names and a mapping between these names and their protocol-specific equivalent through the variable `eudc-protocol-attributes-translation-alist`. Names currently defined by EUDC are `name`, `firstname`, `email` and `phone`.

eudc-query-form-attributes Variable

A list of attributes presented in the query form. Attribute names in this list should be either EUDC attribute names or valid attribute names. You can get a list of valid attribute names for the current protocol with the '`List Valid Attribute Names`' menu item or the *`M-x eudc-get-attribute-list`* command. Defaults to `name`, `email` and `phone`.

eudc-query-form *get-fields-from-server* Command

Display a form to query the directory server. If given a non-`nil` argument the function first queries the server for the existing fields and displays a corresponding form. Not all protocols may support a non-`nil` argument here.

Since the names of the fields may not be explicit enough or adapted to be directly displayed as prompt strings in the form, the variable `eudc-user-attribute-names-alist` lets you define more explicit names for directory attribute names. This variable is ignored if `eudc-use-raw-directory-names` is non-`nil`.

eudc-user-attribute-names-alist Variable

This is an alist of user-defined names for the directory attributes used in query/response forms. Prompt strings for attributes that are not in this alist are derived by splitting the attribute name at underscores and capitalizing the individual words.

eudc-use-raw-directory-names Variable

If non-`nil`, use attributes names as defined in the directory. Otherwise, directory query/response forms display the user attribute names defined in `eudc-user-attribute-names-alist`.

3.3 Display of Query Results

Upon successful completion of a form query, EUDC will display a buffer containing the results of the query.

The fields that are returned for each record are controlled by `eudc-default-return-attributes` (see Section 3.1.2 [Return Attributes], page 4).

The display of each individual field can be performed by an arbitrary function which allows specific processing for binary values, such as images or audio samples, as well as values with semantics, such as URLs.

eudc-attribute-display-method-alist Variable

An alist specifying methods to display attribute values. Each member of the list is of the form `(`*name*` . `*func*`)` where *name* is a lowercased string naming a directory attribute (translated according to `eudc-user-attribute-names-alist` if `eudc-use-raw-directory-names` is non-nil) and *func* a function that will be passed the corresponding attribute values for display.

This variable has protocol-local definitions (see see Section 3.8 [Server/Protocol Locals], page 11). For instance, it is defined as follows for LDAP:

```
(eudc-protocol-set 'eudc-attribute-display-method-alist
                   '(("jpegphoto" . eudc-display-jpeg-inline)
                     ("labeledurl" . eudc-display-url)
                     ("audio" . eudc-display-sound)
                     ("labeledurl" . eudc-display-url)
                     ("url" . eudc-display-url))
                   'ldap)
```

EUDC provides a set of built-in functions to display binary value types:

eudc-display-generic-binary *data* Function
: Display a button for unidentified binary *data*.

eudc-display-url *url* Function
: Display URL and make it clickable.

eudc-display-sound *data* Function
: Display a button to play the sound *data*.

eudc-display-jpeg-inline *data* Function
: Display the JPEG *data* inline at point if possible.

eudc-display-jpeg-as-button *data* Function
: Display a button for the JPEG *data*.

Right-clicking on a binary value button pops up a contextual menu with options to process the value. Among these are saving the attribute value to a file or sending it to an external viewer command. External viewers should expect the value on their standard input and should display it or perform arbitrary processing on it. Messages sent to standard output are discarded. External viewers are listed in the variable `eudc-external-viewers` which you can customize.

eudc-external-viewers Variable
: This is a list of viewer program specifications. Each specification is a list whose first element is a string naming the viewer for unique identification, the second element is the executable program which should be invoked and the following elements are arguments that should be passed to the program.

3.4 Inline Query Expansion

Inline query expansion is a powerful method to get completion from your directory server. The most common usage is for expanding names to email addresses in mail message buffers. The expansion is performed by the command *`M-x eudc-expand-inline`* which is available from the '`Expand Inline Query`' menu item but can also be conveniently bound to a key shortcut (see Chapter 2 [Installation], page 3). The operation is controlled by the variables `eudc-inline-expansion-format`, `eudc-inline-query-format`, `eudc-expanding-overwrites-query` and `eudc-multiple-match-handling-method`.

If the query fails for a server, other servers may be tried successively until one of them finds a match (see Section 3.6 [Multi-server Queries], page 9).

eudc-expand-inline *replace-p* Command
: Query the server and expand the query string before point. The query string consists of the buffer substring from the point back to the preceding comma, colon or beginning of line. `eudc-inline-query-format` controls how individual words are mapped onto directory attribute names. After querying the server for the given string, the expansion specified by `eudc-inline-expansion-format` is inserted in the buffer at point. If *replace-p* is `t` then this expansion replaces the query string in the buffer. If `eudc-expanding-overwrites-query` is non-`nil` then the meaning of *replace-p* is negated.

eudc-inline-query-format Variable
: Format of an inline expansion query. This is actually a list of *format*s. A *format* is a list of one or more EUDC attribute names. A *format* applies if it contains as many attributes as individual words in the inline query string. If several *format*s apply then they are tried

in order until a match is found. If `nil` all the words will be mapped onto the default server/protocol attribute name (generally `name`).

For instance, use the following

```
(setq eudc-inline-query-format '((name)
                                 (firstname)
                                 (firstname name)))
```

to indicate that single word expansion queries are to be considered as surnames and if no match is found then they should be tried as first names. Inline queries consisting of two words are considered as consisting of a first name followed by a surname. If the query consists of more than two words, then the first one is considered as the first name and the remaining words are all considered as surname constituents.

*format*s are in fact not limited to EUDC attribute names, you can use server or protocol specific names in them. It may be safer if you do so, to set the variable `eudc-inline-query-format` in a protocol or server local fashion (see see Section 3.8 [Server/Protocol Locals], page 11).

For instance you could use the following to match up to three words against the `cn` attribute of LDAP servers:

```
(eudc-protocol-set 'eudc-inline-query-format
                   '((cn)
                     (cn cn)
                     (cn cn cn))
                   'ldap)
```

eudc-inline-expansion-format Variable

This variable lets you control exactly what is inserted into the buffer upon an inline expansion request. It is a list whose first element is a string passed to `format`. Remaining elements are symbols corresponding to directory attribute names. The corresponding attribute values are passed as additional arguments to `format`. Default is `("%s" email)` but you may want to consider a value like `("%s <%s>" name email)`

eudc-multiple-match-handling-method Variable

This variable controls what to do when multiple entries match a query for an inline expansion. Possible values are:

`first`	The first match is considered as being the only one, the others are discarded.
`select`	A selection buffer pops up where you can choose a particular match. This is the default value of the variable.
`all`	The expansion uses all records successively
`abort`	An error is signaled. The expansion aborts.

Default is `select`

3.5 The Server Hotlist

EUDC lets you maintain a list of frequently used servers so that you can easily switch from one to another. This hotlist appears in the '`Server`' submenu. You select a server in this list by clicking on its name. You can add the current server to the list with the command *`M-x eudc-bookmark-current-server`*. The list is contained in the variable `eudc-server-hotlist` which is stored in and retrieved from the file designated by `eudc-options-file`. EUDC also provides a facility to edit the hotlist interactively (see Section 3.5.1 [The Hotlist Edit Buffer], page 9).

The hotlist is also used to make queries on multiple servers successively (see Section 3.6 [Multi-server Queries], page 9). The order in which the servers are tried is the order they appear in the hotlist, therefore it is important to sort the hotlist appropriately.

eudc-bookmark-server *server* Command

Add *server* to the hotlist of servers

eudc-bookmark-current-server Command

Add the current server to the hotlist of servers

eudc-options-file Variable

The name of a file where EUDC stores its internal variables (the hotlist and the current server). EUDC will try to load that file upon initialization so, if you choose a file name different from the defaults '`~/.eudc-options`', be sure to set this variable to the appropriate value *before* EUDC is itself loaded.

3.5.1 The Hotlist Edit Buffer

The hotlist edit buffer offers a means to manage a list of frequently used servers. Commands are available in the context pop-up menu generally bound to the right mouse button. Those commands also have equivalent key bindings.

eudc-hotlist-add-server Command

Bound to `a`. Add a new server to the hotlist on the line after point

eudc-hotlist-delete-server Command

Bound to *`d`*. Delete the server on the line point is on

eudc-hotlist-select-server Command

Bound to *`s`*. Select the server the point is on as the current directory server for the next queries

eudc-hotlist-transpose-servers Command

Bound to *`t`*. Bubble up the server the point is on to the top of the list

eudc-hotlist-quit-edit Command

Bound to *`q`*. Save the changes and quit the hotlist edit buffer. Use *`x`* or *`M-x kill-buffer`* to exit without saving.

3.6 Multi-server Queries

When using inline query expansion (see Section 3.4 [Inline Query Expansion], page 7), EUDC can try to query successively a sequence of directory servers until one of them successfully finds a match for the query.

eudc-inline-expansion-servers Variable

This variable controls which servers are tried and in which order when trying to perform an inline query. Possible values are:

`current-server`
Only the current directory server is tried

`hotlist` The servers in the hotlist are tried in order until one finds a match for the query or 'eudc-max-servers-to-query' is reached

`server-then-hotlist`
The current server then the servers in the hotlist are tried in the order they appear in the hotlist until one of them finds a match or 'eudc-max-servers-to-query' is reached. This is the default.

eudc-max-servers-to-query Variable

This variable indicates the maximum number of servers to query when performing a multi-server query. The default, `nil`, indicates that all available servers should be tried.

3.7 Creating BBDB Records

With EUDC, you can automatically create BBDB records (see section "BBDB" in *BBDB Manual*) from records you get from a directory server. You do this by moving point to the appropriate record in a query result display buffer and invoking the command *`M-x eudc-insert-record-at-point-into-bbdb`* with the keyboard binding *`b`*[1], or with the menu. EUDC cannot update an existing BBDB record and will signal an error if you try to insert a record matching an existing one.

It is also possible to export to BBDB the whole batch of records contained in the directory query result with the command *`M-x eudc-batch-export-records-to-bbdb`*.

Because directory systems may not enforce a strict record format, local server installations may use different attribute names and have different ways to organize the information. Furthermore BBDB has its own record structure. For these reasons converting a record from its external directory format to the BBDB format is a highly customizable process.

eudc-bbdb-conversion-alist Variable

The value of this variable should be a symbol naming an alist defining a mapping between BBDB field names onto directory attribute names records. This is a protocol-local variable and is initialized upon protocol switch (see Section 3.8 [Server/Protocol Locals], page 11). The alist is made of cells of the form `(`*bbdb-field*` . `*spec-or-list*`)`. *bbdb-field* is the name of a field that must be defined in your BBDB environment (standard field names are `name`, `company`, `net`, `phone`, `address` and `notes`). *spec-or-list* is either a single mapping specification or a list of mapping specifications. Lists of mapping specifications are valid for the `phone` and `address` BBDB fields only. *spec*s are actually s-expressions which are evaluated as follows:

a string evaluates to itself

a symbol evaluates to the symbol value. Symbols corresponding to directory attribute names present in the record evaluate to the value of the field in the record

a form is evaluated as a function. The argument list may contain attribute names which evaluate to the corresponding values in the record. The form evaluation should return something appropriate for the particular *bbdb-field* (see `bbdb-create-internal`). `eudc-bbdbify-phone` and `eudc-bbdbify-address` are provided as convenience functions to parse phones and addresses.

The default value of the PH-specific value of that variable is `eudc-ph-bbdb-conversion-alist`:

[1] This key binding does not actually call `eudc-insert-record-at-point-into-bbdb` but uses `eudc-try-bbdb-insert` instead.

```
((name . name)
 (net . email)
 (address . (eudc-bbdbify-address address "Address"))
 (phone . ((eudc-bbdbify-phone phone "Phone")
           (eudc-bbdbify-phone office_phone "Office Phone"))))
```

This means that:

- the `name` field of the BBDB record gets its value from the `name` attribute of the directory record
- the `net` field of the BBDB record gets its value from the `email` attribute of the directory record
- the `address` field of the BBDB record is obtained by parsing the `address` attribute of the directory record with the function `eudc-bbdbify-address`
- two `phone` fields are created (when possible) in the BBDB record. The first one has *Phone* for location and its value is obtained by parsing the `phone` attribute of the PH/QI record with the function `eudc-bbdbify-phone`. The second one has *Office Phone* for location its value is obtained by parsing the `office_phone` attribute of the PH/QI record with the function `eudc-bbdbify-phone`.

eudc-bbdbify-phone *phone location* Function

This is a convenience function provided for use in `eudc-bbdb-conversion-alist`. It parses *phone* into a vector compatible with `bbdb-create-internal`. *phone* is either a string supposedly containing a phone number or a list of such strings which are concatenated. *location* is used as the phone location for BBDB.

eudc-bbdbify-address *addr location* Function

This is a convenience function provided for use in `eudc-bbdb-conversion-alist`. It parses *addr* into a vector compatible with `bbdb-create-internal`. *addr* should be an address string of no more than four lines or a list of lines. The last line is searched for the zip code, city and state name. *location* is used as the phone location for BBDB.

Note that only a subset of the attributes you selected with `eudc-default-return-attributes` and that are actually displayed may actually be inserted as part of the newly created BBDB record.

3.8 Server/Protocol Locals

EUDC can be customized independently for each server or directory protocol. All variables can be given local bindings that are activated when a particular server and/or protocol becomes active. This is much like buffer-local bindings but on a per server or per protocol basis.

3.8.1 Manipulating local bindings

EUDC offers functions that let you set and query variables on a per server or per protocol basis.

The following predicates allow you to test the existence of server/protocol local bindings for a particular variable.

eudc-server-local-variable-p *var* Function

Return non-`nil` if *var* has server-local bindings

eudc-protocol-local-variable-p *var* Function

Return non-`nil` if *var* has protocol-local bindings

The following functions allow you to set the value of a variable with various degrees of locality.

eudc-default-set *var val* Function
: Set the EUDC default value of *var* to *val*. The current binding of *var* (if local to the current server or protocol) is not changed.

eudc-protocol-set *var val* &optional *protocol* Function
: Set the binding of *var* local to *protocol* to *val*. If omitted, *protocol* defaults to the current value of `eudc-protocol`. The current binding of *var* is changed only if *protocol* is omitted.

eudc-server-set *var val* &optional *server* Function
: Set the binding of *var* local to *server* to *val*. If omitted, *server* defaults to the current value of `eudc-server`. The current binding of *var* is changed only if *server* is omitted.

eudc-set *var val* Function
: Set the most local (server, protocol or default) binding of *var* to *val*. The current binding of *var* is also set to *val*.

The following variables allow you to query the various bindings of a variable (local or non-local).

eudc-variable-default-value *var* Function
: Return the default binding of *var* (outside of a particular server or protocol local binding). Return `unbound` if *var* has no EUDC default value.

eudc-variable-protocol-value *var* &optional *protocol* Function
: Return the value of *var* local to *protocol*. Return `unbound` if *var* has no value local to *protocol*. *protocol* defaults to `eudc-protocol`.

eudc-variable-server-value *var* [*server*] Function
: Return the value of *var* local to *server*. Return `unbound` if *var* has no value local to *server*. *server* defaults to `eudc-server`.

Changing a protocol-local or server-local value of a variable has no effect on its current value. The following command is used to synchronize the current values of variables with their local values given the current `eudc-server` and `eudc-protocol`:

eudc-update-local-variables Function
: Update all EUDC variables according to their local settings.

4 Credits

EUDC was written by Oscar Figueiredo based on '`ph.el`' by the same author.

Thanks to Soren Dayton for his suggestions, his enthusiasm and his help in testing and proofreading the code and docs of '`ph.el`'.

Command and Function Index

Variables Index

Table of Contents