NAME

ce_db_build – build a Classing Engine database from an ASCII description of the database or generate an ASCII description file of a CE database

SYNOPSIS

ce_db_build **user** | **system** | **network** **-from_ascii** | **-to_ascii** *filename*
[**-db_file** *db_filename*]

DESCRIPTION

ce_db_build builds a Classing Engine database from an ASCII description file *or* generate an ASCII description file of a CE database. The Classing Engine implements a simple hierarchy of databases. Each database contains tables of namespaces. Each namespace table allows for namespace specific mappings of names to attributes. For example, the Files namespace allows for mappings from file names to file attributes; e.g., file type.

An example of the contents of an ASCII description file is:

```
{
NS_NAME=Types
NS_ATTR= ((NS_MANAGER,string, <$CEPATH/tns_mgr.so>))
NS_ENTRIES= (
  (
    (TYPE_NAME,type-id,<binder-prog>)
    (TYPE_ICON,icon-file,<$OPENWINHOME/include/images/bind.icon>)
    (TYPE_ICON_MASK,icon-file,<$OPENWINHOME/include/images/bind.mask.icon>)
    (TYPE_FGCOLOR,color,<91 229 229>)
    (TYPE_BGCOLOR,color,<91 126 229>)
    (TYPE_PRINT,string,<lp -Plp>)
  )(
    (TYPE_NAME,type-id,<calctool-prog>)
    (TYPE_ICON,icon-file,<$OPENWINHOME/include/images/calctool.icon>)
    (TYPE_ICON_MASK,icon-file,<$OPENWINHOME/include/images/calctool.mask.icon>)
    (TYPE_FGCOLOR,color,<255 255 255>)
    (TYPE_BGCOLOR,color,<229 45 183>)
    (TYPE_PRINT,string,<lp -Plp>)
   )
  )
}

{
NS_NAME=Files
NS_ATTR=((NS_MANAGER,junk,<$CEPATH/fns_mgr.so>))
NS_ENTRIES=(
 (
   (FNS_TYPE,refto-Types,<binder-prog>)
   (FNS_FILENAME,str,<binder>)
 )(
```

```
  (FNS_TYPE,refto-Types,<default-app>)
  (FNS_FILENAME,str,<default|app>)
 )
)
}
```

In the above example:

* The ASCII description file defines two namespaces called Types and Files.
* The keyword NS_NAME precedes the name of a namespace.
* The keyword NS_ATTR precedes a list of the attributes of the namespace.

An important namespace attribute is called NS_MANAGER, which identifies the name of the shared library (see **dlopen**(3X)) that implements the namespace manager for the namespace.

NS_ENTRIES precedes a list of entries in the namespace. Each entry consists of a list of attributes.

An attribute has a name, a type and a value.

Attribute names in the example are in upper case e.g. TYPE_NAME, FNS_TYPE. Attribute types are just strings that make sense to the CE client. For example, FNS_TYPE has a type of "refto-Types" - an indication to the CE client that FNS_TYPE values are really TYPE_NAMEs that can be looked up in Types namespace.

A BNF for the ASCII description file is:

```
database          :  name_space | database name_space
name_space        :  { name namespace_attributes entries }
name              :  NS_NAME = variable
ns_attrs          :  NS_ATTR = ( attribute_list )
manager_attribute :  ( NS_MANAGER , attribute_type , attribute_value )
attribute_list    :  attribute
                  |  attribute_list attribute
attribute         :  ( attribute_name , attribute_type , attribute_value )
entries           :  NS_ENTRIES = ( list_of_entries )
list_of_entries   :  an_entry
                  |  list_of_entries an_entry
an_entry          :  ( attribute_list )
attribute_name    :  variable
attribute_type    :  variable
variable          :  Identifier
attribute_value   :  attribute_token
```

The terminals are NS_NAME, NS_ATTR, NS_MANAGER, NS_ENTRIES, "{", "}", "(", ")", "," "=", Identifier and attribute_token. Identifier can have a-z, A-Z, 0-9, _, -.

attribute_token can come in two flavors:

It can begin with a "<" and end with a ">" and can have any ASCII character (except a ">") within it.

It can begin with one or more digits (which represent a number "n") followed by zero or more spaces, followed by a "<", followed by any "n" characters, closed off by a ">". This is the escape mechanism to allow for arbitrary byte string attributes that could have ">" characters within them.

OPTIONS

user | system | network
Indicates whether the user wants to access the user, system or network CE database.

-from_ascii *filename*
Indicates that the user wishes to write the stated CE database from the ASCII file *filename*. The entire CE database will be re-written; i.e. this is an all or nothing update of the CE database.

-to_ascii *filename*
Indicates that the file named *filename* should be written with the ASCII description of the stated CE database. This ASCII description can be modified or added to and supplied as input to an invocation of **ce_db_build** with the -from_ascii argument.

db_file *db_filename*
This form should be used in the case that a particular database is to be read from/written to using *db_filename* as the pathname of the CE database.

ENVIRONMENT

CEPATH
This is a colon separated list of up to three pathnames that the CE uses when looking for a CE database to read or write. It also uses CEPATH as a list of places to look if a particular NS_MANAGER filename has CEPATH prepended to it.

The first pathname is for the "user" database, the second pathname is for the "system" database, and the third pathname is for the "network" database.

If a CEPATH is not defined, or if a particular pathname is not specified in CEPATH, the following database pathnames are used by default:
If the database is:
"user" — ~/.cetables/cetables
"system" — /etc/cetables/cetables
"network" — $OPENWINHOME/lib/cetables/cetables
For example, to set the pathname for the "system" database to /foo/cetables while using the defaults for "user" and "network", set CEPATH to:
:/foo/cetables

FILES

cetables
This is the Classing Engine database file produced in the appropriate directory after **ce_db_build** is complete. A CE database file can be identified by its first

characters which state:
"Classing Engine Data File Version 1.0aaa"

SEE ALSO **ce_db_merge**(1)

BUGS If the CE database file you are building is mounted from a pre-SunOS 4.1 machine, the locking protocol used by **ce_db_build** does not work; i.e., the database does not get locked for writing.

Running **ce_db_build** on an empty ASCII file causes it to hang indefinitely as though it were in an infinite loop.