DNS

0.0.1

Generated by Doxygen 1.8.7

Thu Jan 19 2017 16:30:40

Contents

Chapter 1

Main Page

Implementarea simpla a unui Domain name system (simplu) în C++. Source code here. Documentation here.
Autor: Micu Matei-Marius

Resurse: -RFC 1034 -RFC 1035 -Travis CI -CodeShip -Doxygen

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

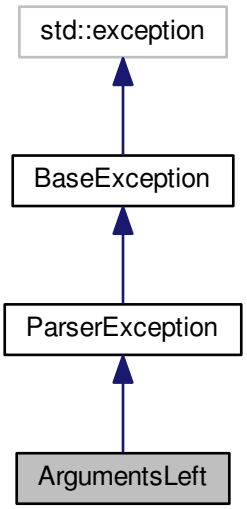
Chapter 5

Class Documentation

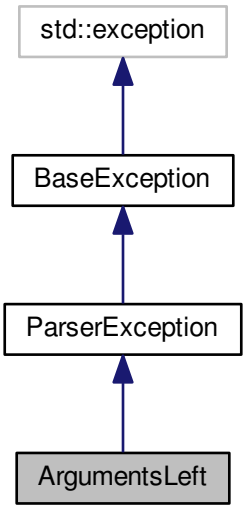
5.1 ArgumentsLeft Class Reference

`#include <exceptions.h>`

Inheritance diagram for ArgumentsLeft:



Collaboration diagram for ArgumentsLeft:



Public Member Functions

- const char * what () const throw ()

5.1.1 Member Function Documentation

5.1.1.1 const char * ArgumentsLeft::what ( ) const throw)

```
122 {
123     return "Au ramas argumente neparsate";
124 }
```

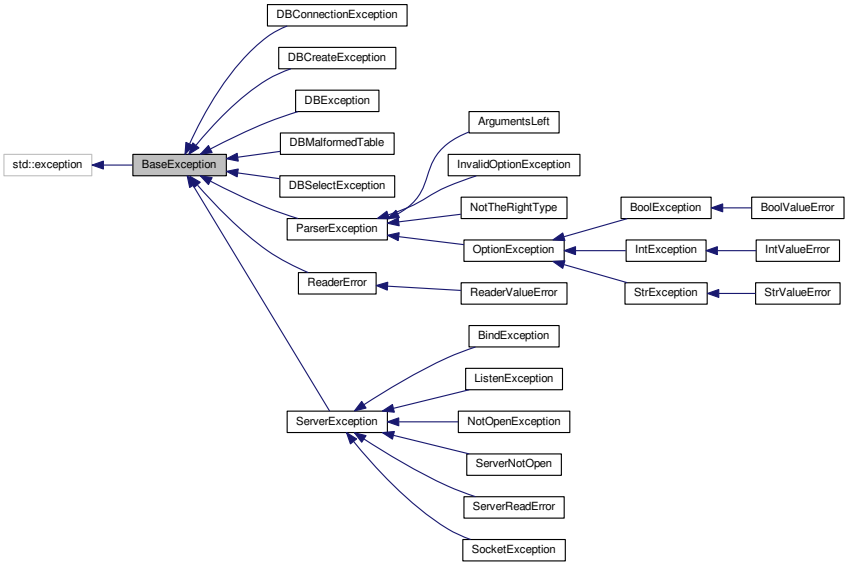
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

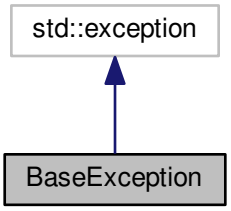
5.2 BaseException Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for BaseException:



Collaboration diagram for BaseException:



Public Member Functions

- const char * what () const throw ()

5.2.1 Member Function Documentation

5.2.1.1 const char * BaseException::what ( ) const throw)

```
17 {
18     return "BaseException: Base exception for DNS project";
19 }
```

Here is the caller graph for this function:

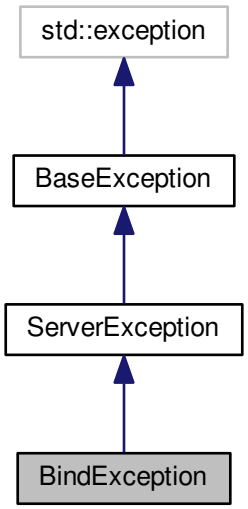
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

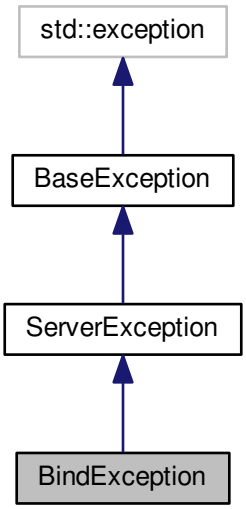
5.3 BindException Class Reference

`#include <exceptions.h>`

Inheritance diagram for BindException:



Collaboration diagram for BindException:



Public Member Functions

- const char * what () const throw ()

5.3.1 Member Function Documentation

5.3.1.1 const char * BindException::what ( ) const throw)

```
142 {
143     return "Eroare la bind !";
144 }
```

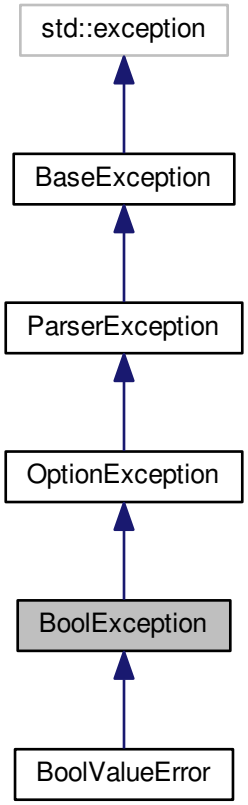
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

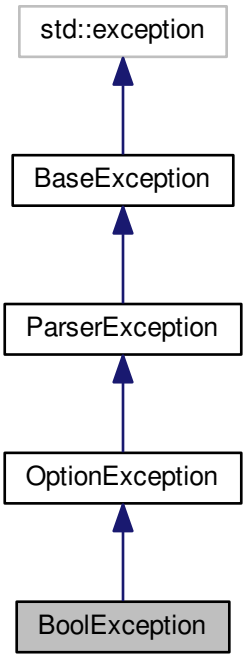
5.4 BoolException Class Reference

`#include <exceptions.h>`

Inheritance diagram for BoolException:



Collaboration diagram for BoolException:



Public Member Functions

- const char * what () throw ()
- BoolException (std::string &primit)

Additional Inherited Members

5.4.1 Constructor & Destructor Documentation

5.4.1.1 BoolException::BoolException (std::string & *primit*)

```
102                                                 : OptionException(
      primit)
103 {
104 }
```

5.4.2 Member Function Documentation

5.4.2.1 const char * BoolException::what ( ) throw)

```
97 {
98     return "BoolException: Base exceptio for BoolOption";
99 }
```

The documentation for this class was generated from the following files:

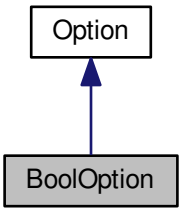
- dns/exceptions.h

- dns/exceptions.cpp

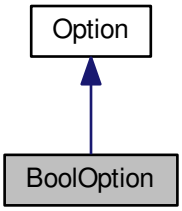
5.5 BoolOption Class Reference

`#include <parser.h>`

Inheritance diagram for BoolOption:



Collaboration diagram for BoolOption:



Public Member Functions

- BoolOption (char short_name, std::string long_name, std::string help_message, bool required)
- void set_default (bool default_value)
- bool is_required ()
- bool get_bool ()
- std::string get_type ()

Protected Member Functions

- void set_value (std::string param_value)

Additional Inherited Members

5.5.1 Constructor & Destructor Documentation

5.5.1.1 BoolOption::BoolOption (char *short_name*, std::string *long_name*, std::string *help_message*, bool *required*)

```
345                                                                 :
346     Option(short_name, long_name, help_message,
      required)
347 {
348     /* Constructor pentru BoolOption
349      *
350      * @param[in] short_name
351      *  Un char care reprezinta varianta prescurtata a parametrului
352      *  Ex: -d
353      *
354      * @param[in] long_name
355      *  Un string care reprezinta varianta lunga aparametrului
356      *  Ex: --delimitator
357      *
358      * @param[in] help_string
359      *  Un string care reprezinta descrierea optiuni
360      *
361      * @param[in] required
362      *  Daca un parametru este necesar sau nu, implicit este setat pe false
363      */
364 }
```

5.5.2 Member Function Documentation

5.5.2.1 bool BoolOption::get_bool ( ) [virtual]

Reimplemented from Option.

```
367 {
368     /* \return{Returneaza valoarea parametrului} */
369     return this->value;
370 }
```

5.5.2.2 std::string BoolOption::get_type ( ) [virtual]

Reimplemented from Option.

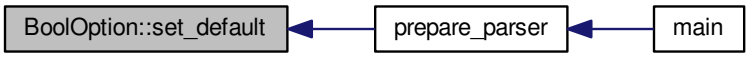
```
414 {
415     /* Returneaza un string care reprezinta tipul optinui */
416     return std::string("bool");
417 }
```

5.5.2.3 bool BoolOption::is_required ( )

5.5.2.4 void BoolOption::set_default (bool *default_value*)

```
373 {
374     /* Seteaza valoarea default a acestei otiuni
375      *
376      * @param[in] defualt_value
377      *  Valoarea default.
378      * */
379     this->default_value = default_value;
380     this->is_set = true; /* Putem sa o consideram setat */
381     this->required = false; /* nu trebuie sa fie parsata de la CLI */
382 }
```

Here is the caller graph for this function:



5.5.2.5 void BoolOption::set_value (std::string *param_value*) [protected], [virtual]

Reimplemented from Option.

```
385 {
386     /* Seteaza valoarea unui parametru convorm unui string
387      *
388      * @param parameter
389      *  Stringul care contine valoarea parametrului
390      */
391     std::string lower_param = parameter;
392     /* Convert to lower case */
393     std::transform(lower_param.begin(), lower_param.end(),
394                    lower_param.begin(), ::tolower);
395
396     if (lower_param == "true" || lower_param == "t")
397     {
398         this->value = true;
399         this->is_set = true;
400         return;
401     }
402
403     if (lower_param == "false" || lower_param == "f")
404     {
405         this->value = false;
406         this->is_set = true;
407         return;
408     }
409
410     throw BoolValueError(parameter);
411 }
```

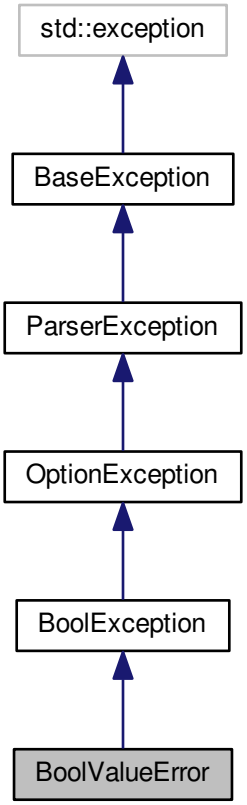
The documentation for this class was generated from the following files:

- dns/parser.h
- dns/parser.cpp

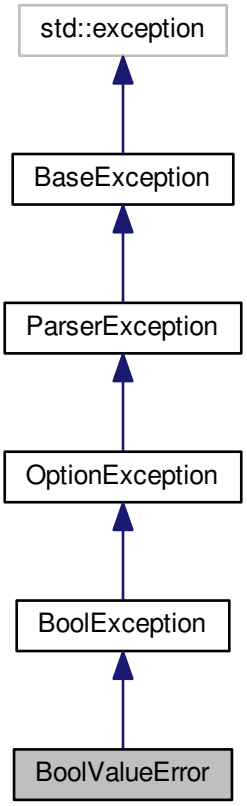
5.6 BoolValueError Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for BoolValueError:



Collaboration diagram for BoolValueError:



Public Member Functions

- const char * what () throw ()
- BoolValueError (std::string &primit)

Additional Inherited Members

5.6.1 Constructor & Destructor Documentation

5.6.1.1 BoolValueError::BoolValueError (std::string & *primit*)

```
112                                                 : BoolException(
      primit)
113 {
114 }
```

5.6.2 Member Function Documentation

5.6.2.1 const char * BoolValueError::what ( ) throw)

```
108 {
```

```
109     return ("Nu putem parsa " + this->primit).c_str();
110 }
```

The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

5.7 DB Class Reference

`#include <db.h>`

Public Member Functions

- DB (char *filename)
- std::string get_ip (char *name, unsigned short name_len)
- ~DB ()

Static Public Attributes

- static unsigned short IP_MAX_SIZE = 3 * 4 + 3

5.7.1 Constructor & Destructor Documentation

5.7.1.1 DB::DB (char * *filename*)

```
63 {
64     /* Initializam conexiunea
65      *
66      *
67      * In cazul in care fisierul nu exista sau nu respecta
68      * structura o vom genera pe loc si o vom popula cu 2 ip-uri
69      * google.com -> 172.217.22.14
70      * example.com -> 1.1.1.1
71      *
72      * @param[in] filename
73      *  Un string cu numele fisierului
74      */
75
76     this->filename = new char[strlen(filename)];
77     bzero(this->filename, strlen(filename));
78     strcpy(this->filename, filename);
79
80     this->_ip = new char[this->IP_MAX_SIZE];
81     bzero(this->_ip, this->IP_MAX_SIZE);
82
83     this->errMsg = NULL;
84
85     int rc = sqlite3_open(this->filename, &this->db);
86
87     if (rc != 0)
88     {
89         throw DBConnectionException();
90     }
91     if (this->_is_prepare() == false)
92     {
93         this->_prepare();
94     }
95 }
```

5.7.1.2 DB::~DB ( )

```
280 {
281     /* Dealocam memoria */
```

```
282     sqlite3_close(this->db);
283     delete this->filename;
284     delete this->_ip;
285 }
```

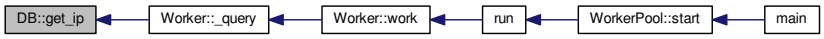
5.7.2 Member Function Documentation

5.7.2.1 std::string DB::get_ip (char * *name*, unsigned short *name_len*)

```
215 {
216     /* Returneaza ip-ul pentru domeniul dat ca parametru.
217      * In cazul in care nu am gasit nici o inregistrare returnam
218      * stringul gol.
219      *
220      * @param[in] name
221      *  Numele domeniului
222      *
223      * @param[in] name_len
224      *  Dimensiunea numelui
225      *
226      */
227     this->lock.lock();
228
229     std::string real_domail;
230     unsigned char lungime = 0;
231     for (unsigned char i = 0; i < name_len-1; ++i)
232     {
233         lungime = name[i];
234         real_domail = real_domail + std::string(".");
235         for (unsigned char j = i+1 ; j <= i+lungime; ++j)
236         {
237             real_domail = real_domail + std::string(1, name[j]);
238         }
239         i = i + lungime;
240     }
241     name = (char*)real_domail.c_str();
242     std::cout << " ------  Search for " << name << std::endl;
243
244     memset(this->_ip, 0, this->IP_MAX_SIZE);
245
246     /* Convertim in string */
247     char c_name[name_len+1];
248     memset(c_name, 0, name_len+1);
249     memcpy(c_name, name, name_len);
250     std::string s_name(c_name), ip("");
251
252     /* NOTE(mmicu): SQL injection DROP TABLE ;) */
253     std::string sql = "SELECT " + std::string(DOMAIN)+", " +std::string(IP) +
254                       " FROM " + std::string(TABLE_NAME) +
255                       " WHERE " + std::string(DOMAIN) +" = '"+ s_name +"';";
256
257     int res = sqlite3_exec(this->db, sql.c_str(), callback, (void*)this->_ip, &this->errMsg);
258
259
260     if (res != 0)
261     {
262         if (this->errMsg != NULL)
263         {
264             /* Eroare de la sqlite */
265             std::cerr << this->errMsg << std::endl;
266             sqlite3_free(this->errMsg);
267         }
268
269     }
270     else
271     {
272         ip = std::string(this->_ip);
273     }
274
275     this->lock.unlock();
276     return ip;
277 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



5.7.3 Member Data Documentation

5.7.3.1 unsigned short DB::IP_MAX_SIZE = 3 * 4 + 3 [static]

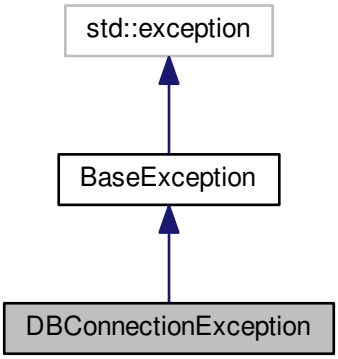
The documentation for this class was generated from the following files:

- dns/db.h
- dns/db.cpp

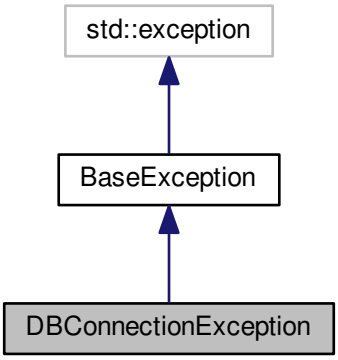
5.8 DBConnectionException Class Reference

`#include <exceptions.h>`

Inheritance diagram for DBConnectionException:



Collaboration diagram for DBConnectionException:



Public Member Functions

- const char * what () const throw ()

5.8.1 Member Function Documentation

5.8.1.1 const char * DBConnectionException::what ( ) const throw)

```
181 {
182     return "Eroare la conectarea cu baza de date.";
183 }
```

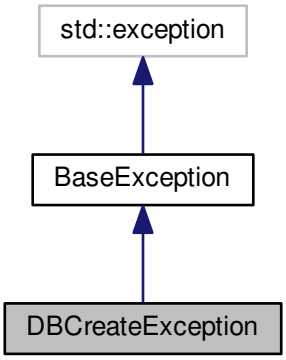
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

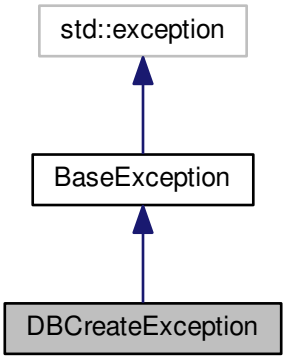
5.9 DBCreateException Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for DBCreateException:



Collaboration diagram for DBCreateException:



Public Member Functions

- const char * what () const throw ()

5.9.1 Member Function Documentation

5.9.1.1 const char * DBCreateException::what ( ) const throw)

```
186 {
187     return "Eroare la crearea bazei de date.";
188 }
```

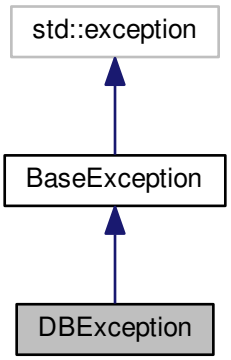
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

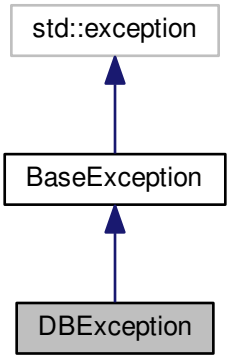
5.10 DBException Class Reference

`#include <exceptions.h>`

Inheritance diagram for DBException:



Collaboration diagram for DBException:



Public Member Functions

- const char * what () const throw ()

5.10.1 Member Function Documentation

5.10.1.1 const char * DBException::what ( ) const throw)

```
176 {
177     return "Eroare la baza de date.";
178 }
```

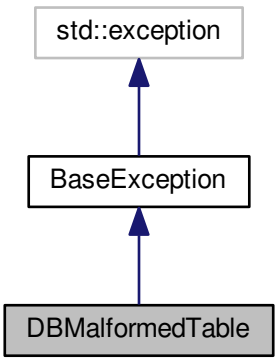
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

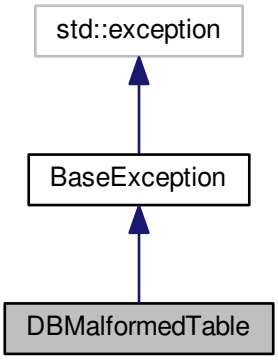
5.11 DBMalformedTable Class Reference

`#include <exceptions.h>`

Inheritance diagram for DBMalformedTable:



Collaboration diagram for DBMalformedTable:



Public Member Functions

- const char * what () const throw ()

5.11.1 Member Function Documentation

5.11.1.1 const char * DBMalformedTable::what ( ) const throw)

```
196 {
197     return "Tabela este malformata!";
198 }
```

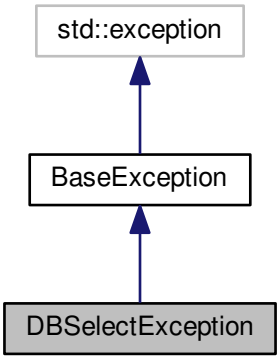
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

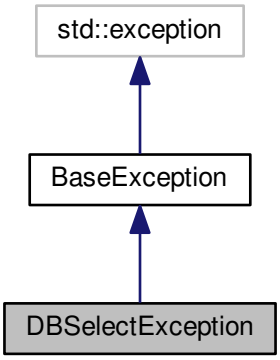
5.12 DBSelectException Class Reference

`#include <exceptions.h>`

Inheritance diagram for DBSelectException:



Collaboration diagram for DBSelectException:



Public Member Functions

- const char * what () const throw ()

5.12.1 Member Function Documentation

5.12.1.1 const char * DBSelectException::what ( ) const throw)

```
191 {
192     return "Eroare la intoregarea bazei de date.";
193 }
```

The documentation for this class was generated from the following files:

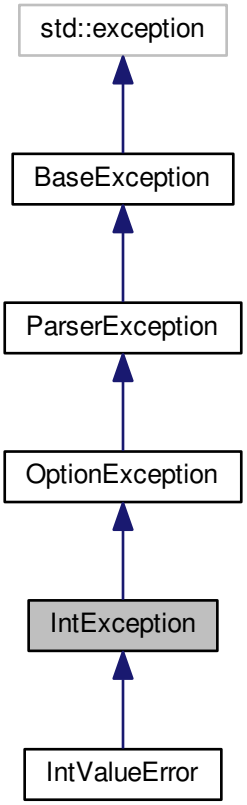
- dns/exceptions.h

- dns/exceptions.cpp

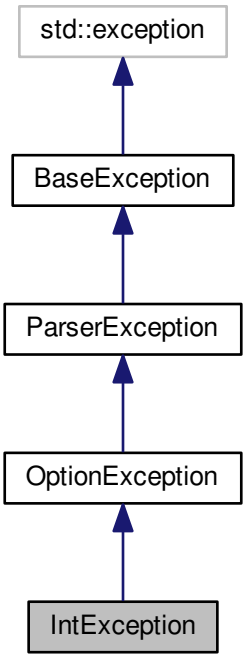
5.13 IntException Class Reference

`#include <exceptions.h>`

Inheritance diagram for IntException:



Collaboration diagram for IntException:



Public Member Functions

- const char * what () throw ()
- IntException (std::string &primit)

Additional Inherited Members

5.13.1 Constructor & Destructor Documentation

5.13.1.1 IntException::IntException (std::string & *primit*)

```
57                                              : OptionException(
      primit)
58 {
59 }
```

5.13.2 Member Function Documentation

5.13.2.1 const char * IntException::what ( ) throw)

```
53 {
54     return "IntException: Base exceptio for IntOption";
55 }
```

The documentation for this class was generated from the following files:

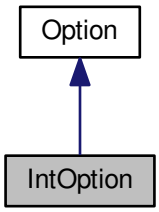
- dns/exceptions.h

- dns/exceptions.cpp

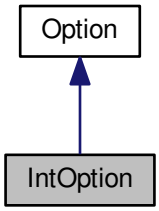
5.14 IntOption Class Reference

`#include <parser.h>`

Inheritance diagram for IntOption:



Collaboration diagram for IntOption:



Public Member Functions

- IntOption (char short_name, std::string long_name, std::string help_message, bool required)
- void set_default (int default_value)
- bool is_required ()
- int get_int ()
- std::string get_type ()

Protected Member Functions

- void set_value (std::string param_value)

Additional Inherited Members

5.14.1 Constructor & Destructor Documentation

5.14.1.1 IntOption::IntOption (char *short_name*, std::string *long_name*, std::string *help_message*, bool *required*)

```
223                                                                 :
224     Option(short_name, long_name, help_message,
      required)
225 {
226     /* Constructor pentru IntOption
227      *
228      * @param[in] short_name
229      *  Un char care reprezinta varianta prescurtata a parametrului
230      *  Ex: -d
231      *
232      * @param[in] long_name
233      *  Un string care reprezinta varianta lunga aparametrului
234      *  Ex: --delimitator
235      *
236      * @param[in] help_string
237      *  Un string care reprezinta descrierea optiuni
238      *
239      * @param[in] required
240      *  Daca un parametru este necesar sau nu, implicit este setat pe false
241      */
242 }
```

5.14.2 Member Function Documentation

5.14.2.1 int IntOption::get_int ( ) [virtual]

Reimplemented from Option.

```
245 {
246     /* \return{Returneaza valoarea parametrului} */
247     return this->value;
248 }
```

5.14.2.2 std::string IntOption::get_type ( ) [virtual]

Reimplemented from Option.

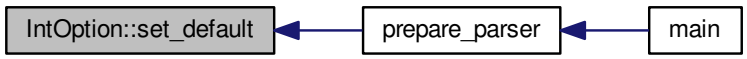
```
279 {
280     /* Returneaza un string care reprezinta tipul optinui */
281     return std::string("int");
282 }
```

5.14.2.3 bool IntOption::is_required ( )

5.14.2.4 void IntOption::set_default (int *default_value*)

```
251 {
252     /* Seteaza valoarea default a acestei otiuni
253      *
254      * @param[in] defualt_value
255      *  Valoarea default.
256      */
257     this->default_value = default_value;
258     this->is_set = true; /* Putem sa o consideram setat */
259     this->required = false; /* nu trebuie sa fie parsata de la CLI */
260 }
```

Here is the caller graph for this function:

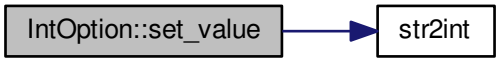


5.14.2.5 void IntOption::set_value (std::string *param_value*) [protected], [virtual]

Reimplemented from Option.

```
263 {
264     /* Seteaza valoarea unui parametru convorm unui string
265      *
266      * @param parameter
267      *  Stringul care contine valoarea parametrului
268      */
269     STR2INT_ERROR out = str2int(this->value, parameter.c_str(), 10);
270 
271     if (out != S2I_SUCCESS)
272     {
273         throw IntValueError(parameter);
274     }
275     this->is_set = true;
276 }
```

Here is the call graph for this function:



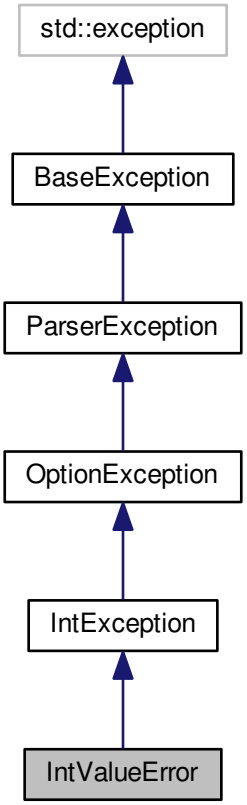
The documentation for this class was generated from the following files:

- dns/parser.h
- dns/parser.cpp

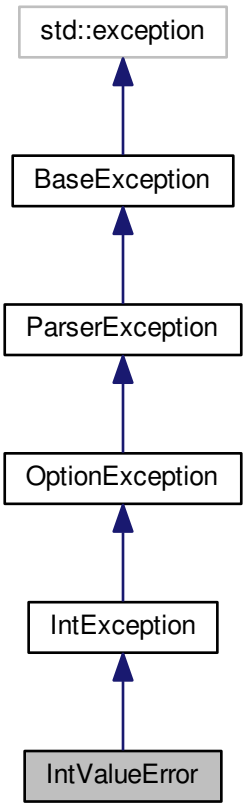
5.15 IntValueError Class Reference

`#include <exceptions.h>`

Inheritance diagram for IntValueError:



Collaboration diagram for IntValueError:



Public Member Functions

- const char * what () const throw ()
- IntValueError (std::string &primit)

Additional Inherited Members

5.15.1 Constructor & Destructor Documentation

5.15.1.1 IntValueError::IntValueError (std::string & *primit*)

```
67                                              : IntException(primit)
68 {
69 }
```

5.15.2 Member Function Documentation

5.15.2.1 const char * IntValueError::what ( ) const throw)

```
63 {
64     return ("Nu putem parsa " + this->primit).c_str();
65 }
```

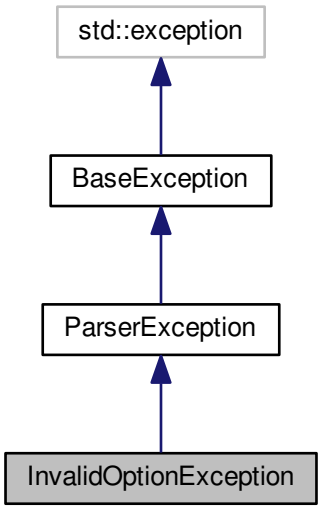
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

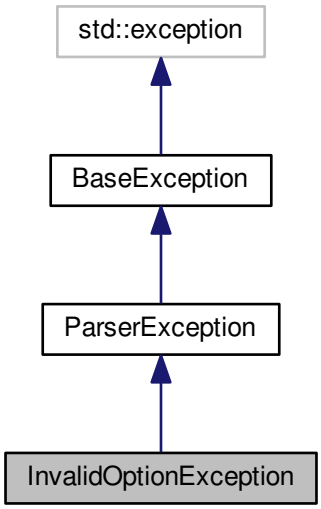
5.16 InvalidOptionException Class Reference

`#include <exceptions.h>`

Inheritance diagram for InvalidOptionException:



Collaboration diagram for InvalidOptionException:



Public Member Functions

- const char * what () const throw ()

5.16.1 Member Function Documentation

5.16.1.1 const char * InvalidOptionException::what ( ) const throw)

```
117 {
118     return "Nu putem adauga optiunea asta";
119 }
```

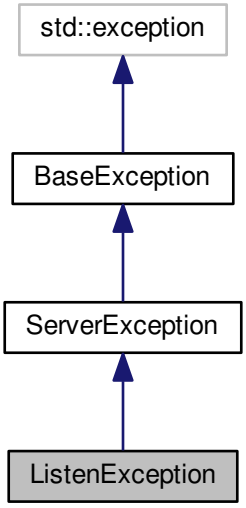
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

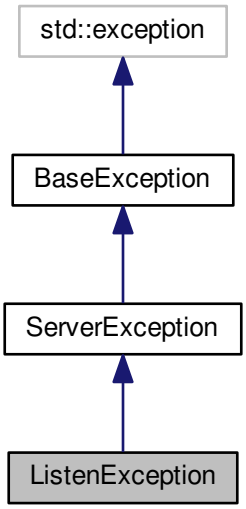
5.17 ListenException Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for ListenException:



Collaboration diagram for ListenException:



Public Member Functions

- const char ∗ what () const throw ()

5.17.1 Member Function Documentation

5.17.1.1 const char * ListenException::what ( ) const throw)

```
147 {
148     return "Eroare la Listen";
149 }
```

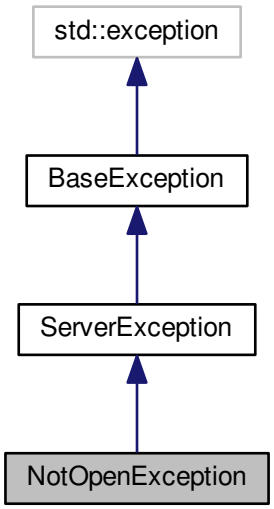
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

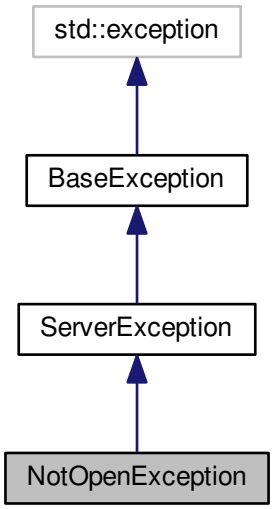
5.18 NotOpenException Class Reference

`#include <exceptions.h>`

Inheritance diagram for NotOpenException:



Collaboration diagram for NotOpenException:



Public Member Functions

- const char * what () const throw ()

5.18.1 Member Function Documentation

5.18.1.1 const char * NotOpenException::what ( ) const throw)

```
151 {
152     return "S-a incercat inchiderea unui server care nu a fost deschis nici o data.";
153 }
```

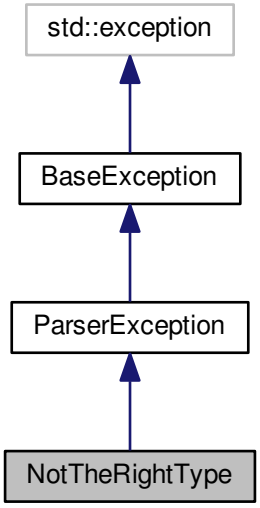
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

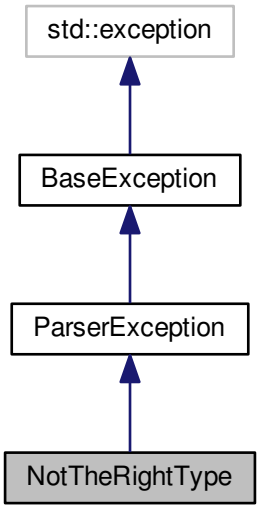
5.19 NotTheRightType Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for NotTheRightType:



Collaboration diagram for NotTheRightType:



Public Member Functions

- const char ∗ what () const throw ()

5.19.1 Member Function Documentation

5.19.1.1 const char * NotTheRightType::what ( ) const throw)

```
127 {
128     return "Argumentul nu are acest tip.";
129 }
```

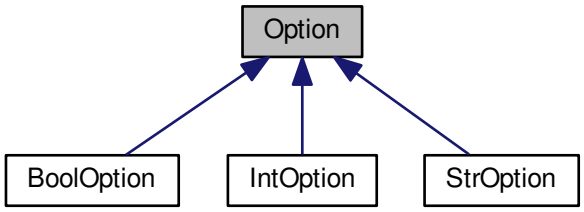
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

5.20 Option Class Reference

`#include <parser.h>`

Inheritance diagram for Option:



Public Member Functions

- Option (char short_name, std::string long_name, std::string help_message, bool required)
- std::string get_help ()
- std::vector< std::string > parse (std::vector< std::string > parameters)
- char get_short_name ()
- std::string get_long_name ()
- bool is_required ()
- virtual int get_int ()
- virtual std::string get_string ()
- virtual bool get_bool ()
- virtual std::string get_type ()

Protected Member Functions

- virtual void set_value (std::string param_value)

Protected Attributes

- char short_name
- std::string long_name
- std::string help_message
- bool default_set
- bool is_set
- bool required

5.20.1 Constructor & Destructor Documentation

5.20.1.1 Option::Option (char *short_name*, std::string *long_name*, std::string *help_message*, bool *required*)

```
95 {
96     /* Constructor pentru Clasa de baza
97      *
98      * @param[in] short_name
99      *  Un char care reprezinta varianta prescurtata a parametrului
100      *  Ex: -d
101      *
102      * @param[in] long_name
103      *  Un string care reprezinta varianta lunga aparametrului
104      *  Ex: --delimitator
105      *
106      * @param[in] help_string
107      *  Un string care reprezinta descrierea optiuni
108      *
109      * @param[in] required
110      *  Daca un parametru este necesar sau nu
111      */
112     this->short_name = short_name;
113     this->long_name = long_name;
114     this->help_message = help_message;
115     this->default_set = false; /* o valoare default este setata dupa crearea obiectului */
116     this->required=required;
117 }
```

5.20.2 Member Function Documentation

5.20.2.1 bool Option::get_bool ( ) [virtual]

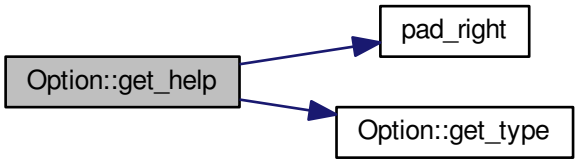
Reimplemented in BoolOption.

```
217 {
218     throw NotTheRightType();
219 }
```

5.20.2.2 std::string Option::get_help ( )

```
120 {
121     /* Returnam mesajul de ajutor complet */
122
123     // TODO(mmicu):
124     // Adauga un format de genu:
125     // (2-4 spaces)short, long(4 - 8 spaces)help_message
126     // poate si un word wrap de ~ 80 caractere
127
128     char ch[2];
129     ch[0] = this->short_name;
130     ch[1] = 0;
131     std::string to_ret = "  -" + std::string(ch) + ", --"+pad_right(this->
    long_name, 12) +
132                          "  "+pad_right("("+this->get_type() +")", 6) + " :" +
133                          this->help_message + "\n";
134     return to_ret;
135 }
```

Here is the call graph for this function:



5.20.2.3 int Option::get_int ( ) [virtual]

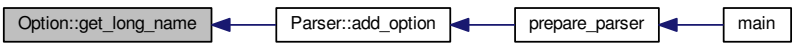
Reimplemented in IntOption.

```
207 {
208     throw NotTheRightType();
209 }
```

5.20.2.4 std::string Option::get_long_name ( )

```
154 {
155     /* Returneaza numele lung a optinui */
156     return this->long_name;
157 }
```

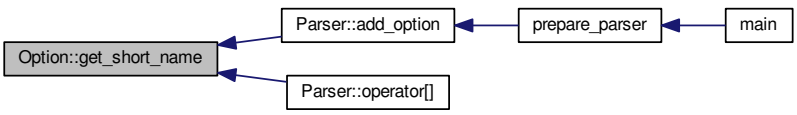
Here is the caller graph for this function:



5.20.2.5 char Option::get_short_name ( )

```
148 {
149     /* Returneaza numele scurt a optinui */
150     return this->short_name;
151 }
```

Here is the caller graph for this function:



5.20.2.6 std::string Option::get_string ( ) [virtual]

Reimplemented in StrOption.

```
212 {
213     throw NotTheRightType();
214 }
```

5.20.2.7 std::string Option::get_type ( ) [virtual]

Reimplemented in BoolOption, StrOption, and IntOption.

```
168 {
169     /* Returneaza un string care reprezinta tipul optinui */
170     return std::string("none");
171 }
```

Here is the caller graph for this function:

5.20.2.8 bool Option::is_required ( )

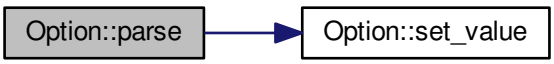
```
160 {
161     /* Returneaza true daca optiunea este obligatori, fals
162      * in caz contrar
163      */
164     return this->required;
165 }
```

5.20.2.9 std::vector< std::string > Option::parse (std::vector< std::string > *parameters*)

```
174 {
175     /* Parseaza lista cu parametri si verifica daca parametrul cautat
176      * se afla in lista.
177      *
178      * @param[in] parameters
179      *  Un vector cu stringuri, fiecare string este un parametru de la CLI.
180      */
181     // for (std::vector<std::string> it = parameters.begin(); it != parameters.end();)
182     for (auto it = parameters.begin(); it != parameters.end();)
183     {
184         char sh[3];
185         sh[0] = '-';
186         sh[1] = this->short_name;
187         sh[2] = 0;
188         if ((std::string)(*it) == std::string(sh) ||
189            (std::string)(*it) == "--"+this->long_name)
190         {
191             parameters.erase(it);
192             this->set_value((std::string)(*it));
193             parameters.erase(it);
194         }
195         else
196         {
197             /* Avanseaza doar daca nu ai eliminat ceva */
198             it++;
```

```
199         }
200
201     }
202
203     return parameters;
204 }
```

Here is the call graph for this function:

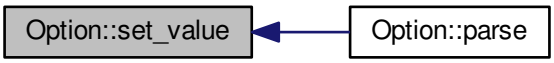


5.20.2.10 void Option::set_value (std::string *param_value*) [protected], [virtual]

Reimplemented in BoolOption, StrOption, and IntOption.

```
138 {
139     /* Seteaza valoarea parametrului
140      *
141      * @param param_value
142      *  Valoarea parametrului
143      */
144     throw ParserException();
145 }
```

Here is the caller graph for this function:



5.20.3 Member Data Documentation

5.20.3.1 bool Option::default_set [protected]

5.20.3.2 std::string Option::help_message [protected]

5.20.3.3 bool Option::is_set [protected]

5.20.3.4 std::string Option::long_name [protected]

5.20.3.5 bool Option::required [protected]

5.20.3.6 char Option::short_name [protected]

The documentation for this class was generated from the following files:

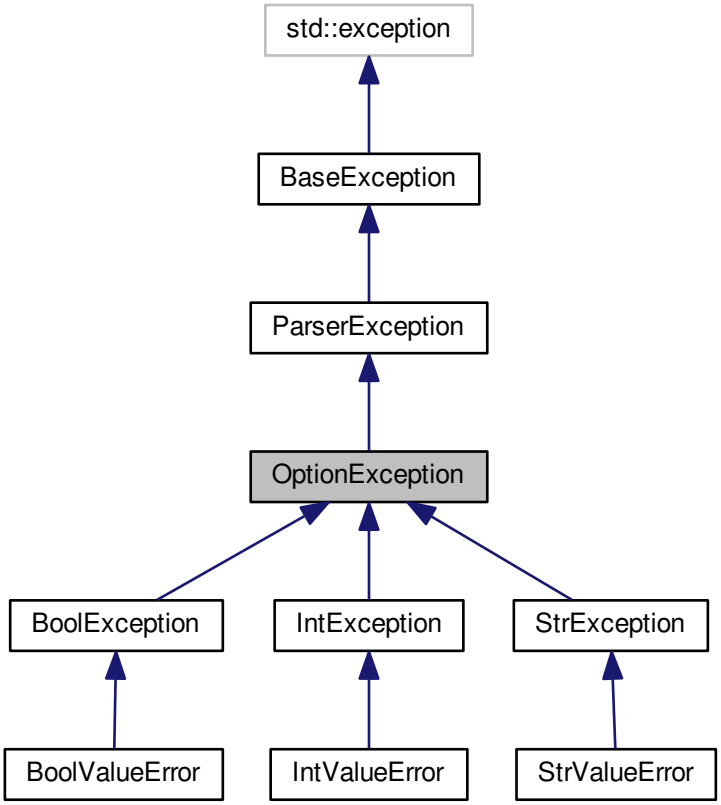
- dns/parser.h

- dns/parser.cpp

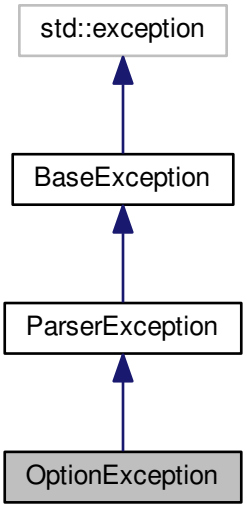
5.21 OptionException Class Reference

`#include <exceptions.h>`

Inheritance diagram for OptionException:



Collaboration diagram for OptionException:



Public Member Functions

- const char * what () throw ()
- OptionException (std::string &primit)
- ~OptionException () throw ()

Public Attributes

- std::string primit

5.21.1 Constructor & Destructor Documentation

5.21.1.1 OptionException::OptionException (std::string & *primit*)

```
36 {
37     /* Contructor pentru OptionException.
38         *
39         * @param[in] primit
40         *  Un string care reprezinta parametrul primit.
41         *  Implicit este stringul gol.
42         */
43     this->primit = primit;
44 }
```

5.21.1.2 OptionException::~OptionException ( ) throw)

```
46 {}
```

5.21.2 Member Function Documentation

5.21.2.1 const char* OptionException::what ( ) throw)

5.21.3 Member Data Documentation

5.21.3.1 std::string OptionException::primit

The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

5.22 Parser Class Reference

```
#include <parser.h>
```

Public Member Functions

- Parser ()
- void add_option (Option *opt)
- void parse (int argc, char *argv[])
- Option * operator[] (std::string name)
- void get_help ()

5.22.1 Constructor & Destructor Documentation

5.22.1.1 Parser::Parser ( )

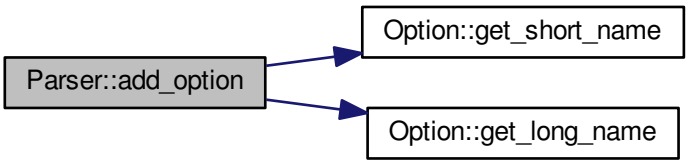
```
422 {
423     /* Initializeaza un parser */
424     this->options.clear();
425     this->args.clear();
426 }
```

5.22.2 Member Function Documentation

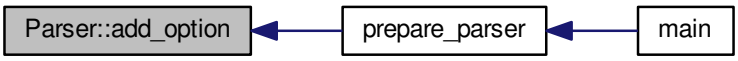
5.22.2.1 void Parser::add_option (Option * *opt*)

```
428 {
429     /* Adauga o noua optiune parserului
430      *
431      * @param opt
432      *  O instanta a unei optiuni
433      */
434     for (std::vector<Option*>::iterator it = this->options.begin();
435         it != this->options.end(); it++)
436     {
437         if ((*it)->get_short_name() == opt->get_short_name() ||
438             (*it)->get_long_name() == opt->get_long_name())
439         {
440             throw InvalidOptionException();
441         }
442     }
443     this->options.push_back(opt);
444 }
```

Here is the call graph for this function:



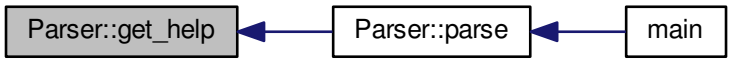
Here is the caller graph for this function:



5.22.2.2 void Parser::get_help ( )

```
526 {
527     /* Print the help for each option */
528     std::cout << "Usage :" << std::endl << "  " << this->exe_name << std::endl;
529 
530 
531     /* Argumente obligatorii */
532     for (std::vector<Option*>::iterator it = this->options.begin();
533         it != this->options.end(); it++)
534     {
535         if ((*it)->is_required())
536         {
537             std::cout << (*it)->get_help() << std::endl;
538         }
539     }
540 
541     std::cout<< "Options :" << std::endl;
542     for (std::vector<Option*>::iterator it = this->options.begin();
543         it != this->options.end(); it++)
544     {
545         if ((*it)->is_required() == false)
546         {
547             std::cout << (*it)->get_help() << std::endl;
548         }
549     }
550 }
```

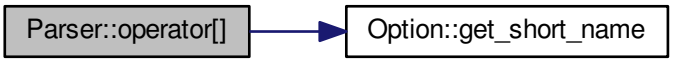
Here is the caller graph for this function:



5.22.2.3 Option * Parser::operator[] (std::string *name*)

```
502 {
503     /* Returneaza optiunea cu acel nume
504      *
505      * @param name
506      *  Numele optiuni, scurt sau lung
507      */
508
509     for (std::vector<Option*>::iterator it = this->options.begin();
510         it != this->options.end(); it++)
511     {
512         char sh[2];
513         sh[0] = (*it)->get_short_name();
514         sh[0] = 0;
515         if ((*it)->get_long_name() == name ||
516             std::string(sh) == name)
517         {
518             return *it;
519         }
520     }
521
522     throw InvalidOptionException();
523 }
```

Here is the call graph for this function:



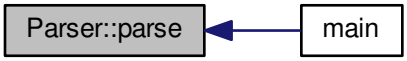
5.22.2.4 void Parser::parse (int *argc*, char * *argv[]*)

```
447 {
448     /* Parse the arguments
449      *
450      * @param argc
451      *  Numarul de argumente primite
452      *
453      * @param argv
454      *  Un vector cu char* care reprezinta argumentele
455      */
456
457     /* Pastram numele executabilului */
458     this->exe_name = std::string(argv[0]);
459
460     /* Transformam char* in vector de std::string */
461
462     /* NOTE(mmicu): Pornim de la 1 ca sa excludem
```

```
463      * numele binarului
464      */
465     for (int i = 1; i < argc; i++)
466     {
467         this->args.push_back(std::string(argv[i]));
468     }
469 
470     /* Verific daca s-a cecur help-ul */
471     for (auto it = this->args.begin(); it != this->args.end(); it++)
472     {
473         if ((*it) == "-h" || (*it) == "--help" )
474         {
475             this->get_help();
476             exit(0);
477         }
478     }
479 
480     /* Daca nu avem nici un argument, afisam help-ul */
481     if (this->args.size() == 0)
482     {
483         /* Daca afisam mesajul de ajutor, nu mai parsam restul argumentelor.
484          * Terminam totui executia programului. */
485         this->get_help();
486         exit(0);
487     }
488 
489     for (std::vector<Option*>::iterator it = this->options.begin();
490         it != this->options.end(); it++)
491     {
492         this->args = (*it)->parse(this->args);
493     }
494 
495     if (this->args.size() > 0 )
496     {
497         throw ArgumentsLeft();
498     }
499 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



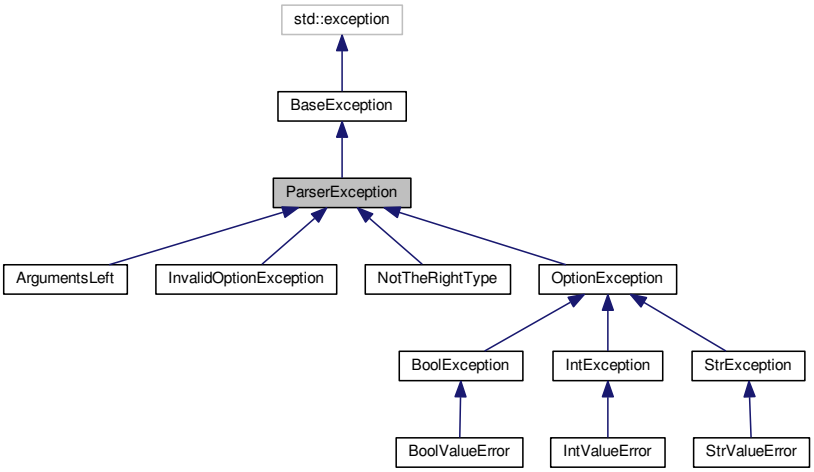
The documentation for this class was generated from the following files:

- dns/parser.h
- dns/parser.cpp

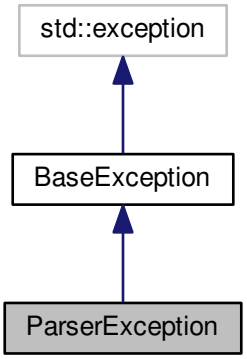
5.23 ParserException Class Reference

`#include <exceptions.h>`

Inheritance diagram for ParserException:



Collaboration diagram for ParserException:



Public Member Functions

- const char * what () const throw ()

5.23.1 Member Function Documentation

5.23.1.1 const char * ParserException::what ( ) const throw)

```
25 {
```

```
26     return "ParserException: Base exception for parser";
27 }
```

The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

5.24 Question Class Reference

`#include <dns.h>`

Public Member Functions

- Question ()
- void set_name (char *name, unsigned short length)
- void set_type (char type[2])
- void set_class (char qclass[2])
- void get_name (char **name, unsigned short &length)
- void get_type (char *type)
- void get_class (char *qclass)
- void print_info ()
- void serialize (char **data, unsigned short &len)
- void serialize_hex ()

5.24.1 Constructor & Destructor Documentation

5.24.1.1 Question::Question ( )

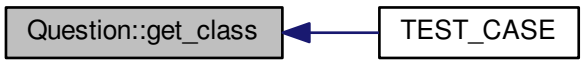
```
69 {
70     /* Initializeaza o unsigned shortrebare */
71     this->qname = NULL;
72 
73     bzero(this->qtype, 2);
74     bzero(this->qclass, 2);
75 
76     this->qname_len = 0;
77 }
```

5.24.2 Member Function Documentation

5.24.2.1 void Question::get_class (char * *qclass*)

```
153 {
154     /* Returneaza clasa unei unsigned shortrebari
155     *
156     * @param[out] cls[2]
157     *  Clasa unsigned shortrebari.
158     */
159 
160     memcpy(cls, this->qclass, 2);
161 }
```

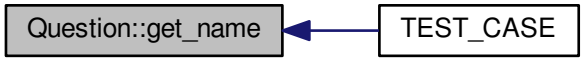
Here is the caller graph for this function:



5.24.2.2 void Question::get_name (char ** *name*, unsigned short & *length*)

```
118 {
119     /* Returneaza numele unsigned shortrebari
120      *
121      * @param[out] *name
122      *  Numele intrebari
123      *
124      * @param[out] lenght
125      *  Lungimea stringului
126      * */
127      if (this->qname_len == 0)
128      {
129          *name = NULL;
130          length = this->qname_len;
131      }
132      else
133      {
134         char *rname = new char[this->qname_len];
135         memcpy(rname, this->qname, this->qname_len);
136         *name = rname;
137         length = this->qname_len;
138      }
139 }
```

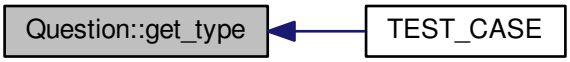
Here is the caller graph for this function:



5.24.2.3 void Question::get_type (char * *type*)

```
142 {
143     /* Returneaza tipul unei unsigned shortrebari
144      *
145      * @param[out] type[2]
146      *  Tipul unsigned shortrebari.
147      */
148
149      memcpy(type, this->qtype, 2);
150 }
```

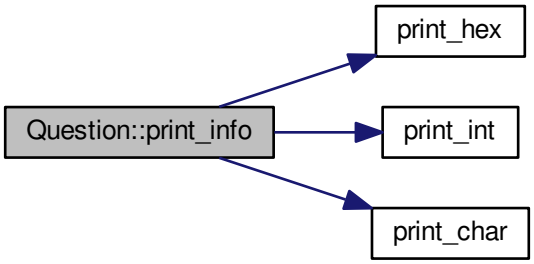
Here is the caller graph for this function:



5.24.2.4 void Question::print_info ( )

```
164 {
165     /* Printeaza informatii despre o intrebare */
166     std::cout <<"  Name :";
167     print_hex(this->qname, this->qname_len, true);
168     std::cout <<"  Name :";
169     print_int(this->qname, this->qname_len, true);
170     std::cout <<"  Name :";
171     print_char(this->qname, this->qname_len, true);
172     std::cout <<"  Lungime :" << (int)this->qname_len << std::endl;
173 
174     std::cout <<" Type: ";
175     print_hex(this->qtype, 2, true);
176 
177     std::cout <<" Class: ";
178     print_hex(this->qclass, 2, true);
179     std::cout << std::endl;
180 }
```

Here is the call graph for this function:

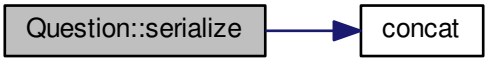


5.24.2.5 void Question::serialize (char ** *data*, unsigned short & *len*)

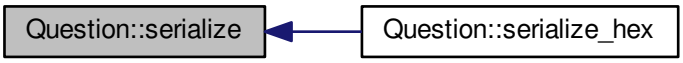
```
183 {
184     /* Serializeaza obiectul curent */
185     char aux[this->qname_len];
186     memset(aux, 0, this->qname_len);
187     memcpy(aux, this->qname, this->qname_len);
188     concat(data, len, aux, (this->qname_len), NULL, 0);
189 
190     char aux_2[2];
191     bzero(aux_2, sizeof(aux_2));
192     memcpy(aux_2, this->qtype, 2);
193     concat(data, len, *data, len, aux_2, 2);
```

```
194
195     bzero(aux_2, sizeof(aux_2));
196     memcpy(aux_2, this->qclass, 2);
197     concat(data, len, *data, len, aux_2, 2);
198 }
```

Here is the call graph for this function:



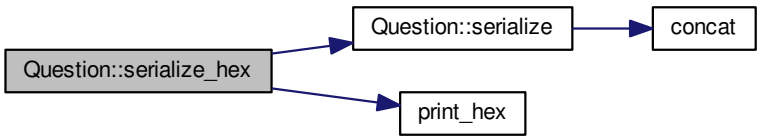
Here is the caller graph for this function:



5.24.2.6 void Question::serialize_hex ( )

```
201 {
202     /* Afiseaza la stdout hexa serializari obiectului curent */
203     char* data;
204     unsigned short len;
205     this->serialize(&data, len);
206     print_hex(data, len, true);
207 }
```

Here is the call graph for this function:

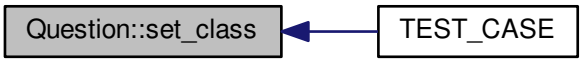


5.24.2.7 void Question::set_class (char *qclass[2]*)

```
107 {
108     /* Seteaza clasa unei intrebari
```

```
109      *
110      * @param[in] cls[2]
111      *  Clasa intrebari.
112      */
113
114      memcpy(this->qclass, cls, 2);
115 }
```

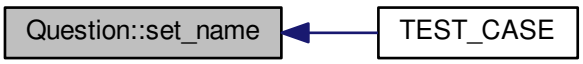
Here is the caller graph for this function:



5.24.2.8 void Question::set_name (char * *name*, unsigned short *length*)

```
80 {
81     /* Seteaza un nume pentru unsigned shortrebare
82      *
83      * @param[in] *name
84      *  Numele intrebari
85      *
86      * @param[in] lenght
87      *  Lungimea stringului
88      * */
89
90      this->qname_len = length;
91
92      this->qname = new char[length];
93      memcpy(this->qname, name, length);
94 }
```

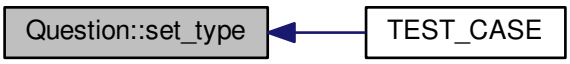
Here is the caller graph for this function:



5.24.2.9 void Question::set_type (char *type[2]*)

```
97 {
98     /* Seteaza tipul unei intrebari
99      *
100      * @param[in] type[2]
101      *  Tipul intrebari.
102      */
103      memcpy(this->qtype, type, 2);
104 }
```

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- dns/dns.h
- dns/dns.cpp

5.25 Reader Class Reference

`#include <reader.h>`

Public Member Functions

- Reader ()
- void set_server (Server *s)
- Server * get_server ()
- Tranzaction * read ()
- ~Reader ()

5.25.1 Constructor & Destructor Documentation

5.25.1.1 Reader::Reader ( )

```
20 {
21     /* Initializam un reader */
22     this->server = NULL;
23     this->buff = (char*)malloc(this->BUFF_SIZE);
24 }
```

5.25.1.2 Reader::~Reader ( )

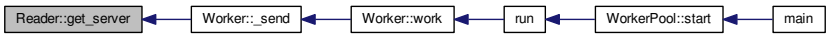
```
27 {
28     /* Initializam un reader */
29     if (this->buff != NULL)
30     {
31         free(this->buff);
32     }
33 }
```

5.25.2 Member Function Documentation

5.25.2.1 Server * Reader::get_server ( )

```
320 {
321     /* Returneaza serverul de dns */
322     return this->server;
323 }
```

Here is the caller graph for this function:

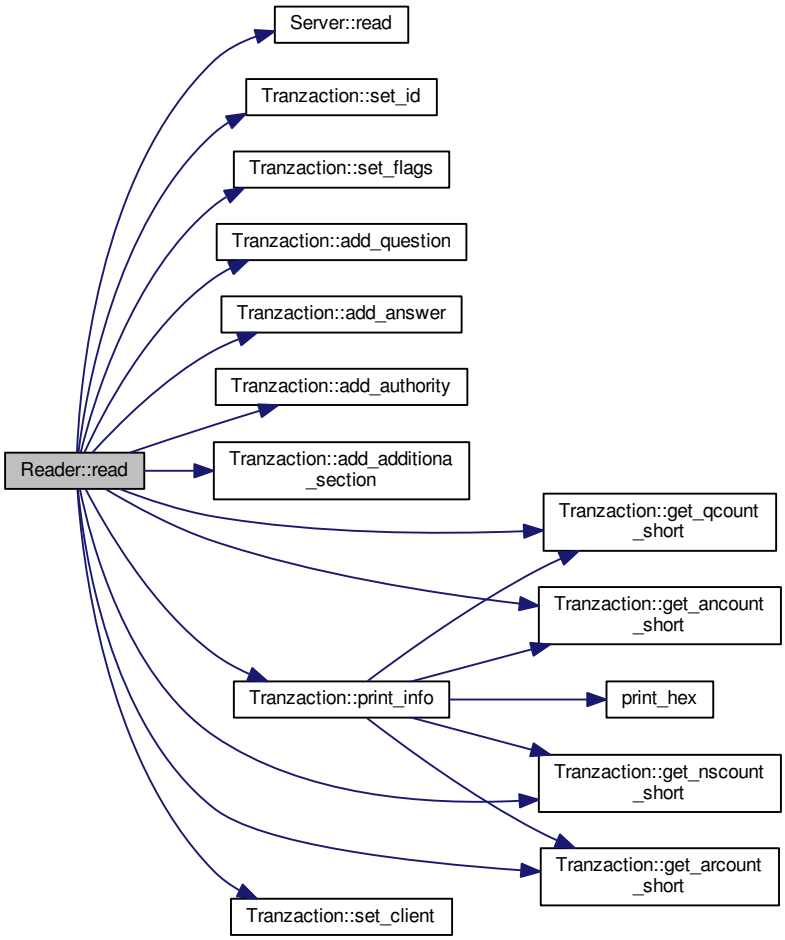


5.25.2.2 Tranzaction * Reader::read ( )

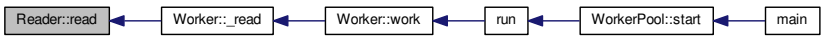
```
175 {
176     Tranzaction* tr = NULL;
177     /* Citim pana cand dam de o tranzactie buna */
178     bool found = false;
179     while (found == false)
180     {
181         if (tr != NULL)
182         {
183             /* Distrugem tranzactia facuta anterior */
184             delete tr;
185         }
186         tr = new Tranzaction();
187         /* Citeste un pachet si returneaza o tranzactie */
188         bzero(this->buff, sizeof(this->buff));
189         int data_len = 0;
190 
191         /* informatii despre client */
192         struct sockaddr client;
193         /* NOTE(mmicu): len_client e parametru de intrare iesier */
194         socklen_t len_client = sizeof(client);
195         bzero(&client, sizeof(client));
196 
197 
198         try
199         {
200             data_len = this->server->read(buff, 512, 0, (struct sockaddr*) &client, &len_client);
201         }
202         catch (ServerReadError& exp)
203         {
204             std::cout << "Malformed request (Eroare la citire) :"<< std::endl;
205             for (int i=0; i<data_len; ++i)
206             {
207                 std::cout << std::hex << (int)buff[i];
208             }
209             std::cout<< std::endl;
210 
211             continue;
212         }
213 
214         if (data_len < 6 * 2)
215         {
216             /* Dimensiunea headerului */
217             std::cout << "Malformed request (Prea scurt) :"<< std::endl;
218             for (int i=0; i<data_len; ++i)
219             {
220                 std::cout << std::hex << (int)buff[i];
221             }
222             std::cout<< std::endl;
223         }
224         /* Setam id-ul tranzactiei */
225         char buff_2[2];
226         memcpy(buff_2, this->buff, 2);
227         tr->set_id(buff_2);
228 
229         /* Setam flags */
230         memcpy(buff_2, &this->buff[2], 2);
231         tr->set_flags(buff_2);
232 
233         unsigned short qcount=0,
234                        ancount=0,
235                        nscount=0,
236                        arcount=0,
237                        short_buff;
238 
239         memcpy(&short_buff, &this->buff[4], 2);
240         qcount = ntohs(short_buff);
241 
242         memcpy(&short_buff, &this->buff[6], 2);
243         ancount = ntohs(short_buff);
244 
```

```
245         memcpy(&short_buff, &this->buff[8], 2);
246         nscount = ntohs(short_buff);
247 
248         memcpy(&short_buff, &this->buff[10], 2);
249         arcount = ntohs(short_buff);
250 
251         unsigned short max_index;
252 
253         /* parsam intrebarile */
254         std::vector<Question> aux = this->parser_question(
255             this->buff, data_len, 12, qcount, max_index);
256 
257         for (std::vector<Question>::iterator it = aux.begin();
258              it != aux.end(); ++it)
259         {
260             tr->add_question((*it));
261         }
262 
263         /* Raspunsuri */
264 
265         std::vector<Resource> aux_res = this->parser_resource(
266             this->buff, data_len, max_index, ancount, max_index);
267 
268         for (std::vector<Resource>::iterator it = aux_res.begin();
269              it != aux_res.end(); ++it)
270         {
271             tr->add_answer((*it));
272         }
273         /* Authority */
274         aux_res = this->parser_resource(
275             this->buff, data_len, max_index, nscount, max_index);
276 
277         for (std::vector<Resource>::iterator it = aux_res.begin();
278              it != aux_res.end(); ++it)
279         {
280             tr->add_authority((*it));
281         }
282 
283         /* Additional */
284         aux_res = this->parser_resource(
285             this->buff, data_len, max_index, arcount, max_index);
286 
287         for (std::vector<Resource>::iterator it = aux_res.begin();
288              it != aux_res.end(); ++it)
289         {
290             tr->add_additiona_section((*it));
291         }
292 
293 
294         /* Verificam daca am consumat tot continutul cum trebuie */
295         if (qcount  != tr->get_qcount_short()   ||
296             ancount != tr->get_ancount_short()  ||
297             nscount != tr->get_nscount_short()  ||
298             arcount != tr->get_arcount_short())
299         {
300             std::cout << "Malformed request (Nu am putut citi cate elemente trebuia) :"<< std::endl;
301             found = false;
302         }
303 
304         if (max_index == data_len)
305         {
306             found = true;
307             tr->set_client(client);
308         }
309         else
310         {
311             std::cout << "Malformed request (Nu am consumat tot raspunsu) :"<< std::endl;
312             found = false;
313         }
314         tr->print_info();
315     } /* end while(found) */
316     return tr;
317 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

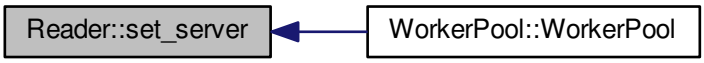


5.25.2.3 void Reader::set_server (Server * s)

```
36 {
37     /* Seteaza un server
38      *
39      * @param[in] s
40      *  Un obiect de tip Server
41      */
42
43     if (s == NULL)
44     {
45         throw ReaderValueError();
46     }
47     this->server = s;
```

```
48 }
```

Here is the caller graph for this function:



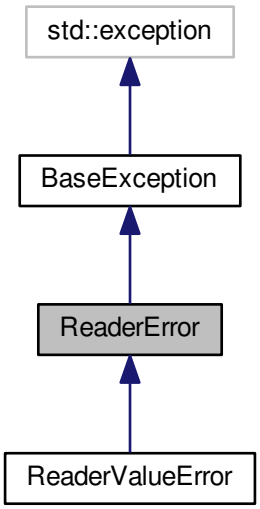
The documentation for this class was generated from the following files:

- dns/reader.h
- dns/reader.cpp

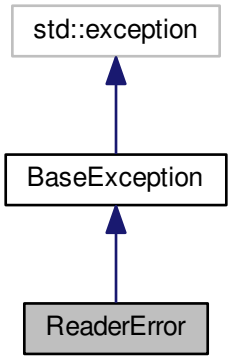
5.26 ReaderError Class Reference

`#include <exceptions.h>`

Inheritance diagram for ReaderError:



Collaboration diagram for ReaderError:



Public Member Functions

- const char * what () const throw ()

5.26.1 Member Function Documentation

5.26.1.1 const char * ReaderError::what ( ) const throw)

```
166 {
167     return "Eroare in Reader";
168 }
```

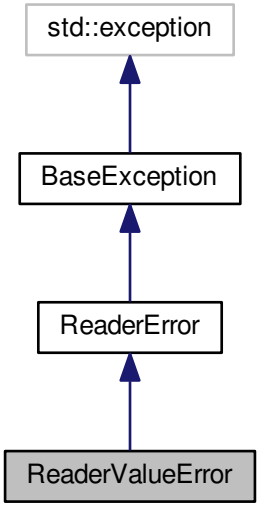
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

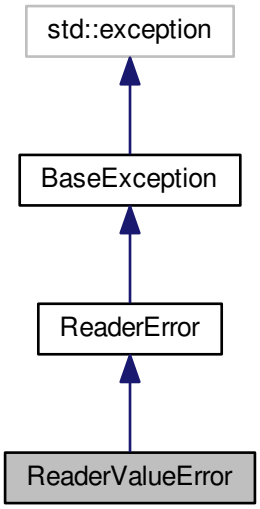
5.27 ReaderValueError Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for ReaderValueError:



Collaboration diagram for ReaderValueError:



Public Member Functions

- const char ∗ what () const throw ()

5.27.1 Member Function Documentation

5.27.1.1 const char * ReaderValueError::what ( ) const throw)

```
171 {
172     return "Valoare gresita pasata unei metode din Reader";
173 }
```

The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

5.28 Resource Class Reference

```
#include <dns.h>
```

Public Member Functions

- Resource ()
- void set_name (char *name, unsigned short length)
- void set_type (char type[2])
- void set_class (char qclass[2])
- void set_ttl (char ttl[4])
- void set_data (char *data, unsigned short length)
- void get_name (char **name, unsigned short &length)
- void get_type (char *type)
- void get_class (char *qclass)
- void get_ttl (char *ttl)
- void get_data (char **data, unsigned short &length)
- void print_info ()
- void serialize (char **data, unsigned short &len)
- void serialize_hex ()

5.28.1 Constructor & Destructor Documentation

5.28.1.1 Resource::Resource ( )

```
210 {
211     /* Initializarea unei resurse */
212 
213     bzero(this->type, 2);
214     bzero(this->cls, 2);
215     bzero(this->rdlength, 2);
216     this->rdata = NULL;
217     this->name = NULL;
218     this->name_len = 0;
219 }
```

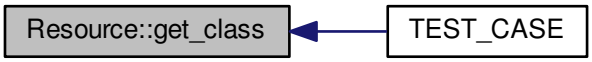
5.28.2 Member Function Documentation

5.28.2.1 void Resource::get_class (char * *qclass*)

```
322 {
323      /* Returneaza clasa unei resurse
324      *
325      * @param[out] cls[2]
326      *  Clasa resursei.
327      */
```

```
328
329      memcpy(cls, this->cls, 2);
330 }
```

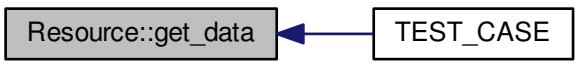
Here is the caller graph for this function:



5.28.2.2 void Resource::get_data (char ** *data*, unsigned short & *length*)

```
344 {
345     /* Setarea unor informatii despre resursa
346      *
347      * @param[out] data
348      *  Informatia propriuzisa
349      *
350      * @param[out] length_data
351      *  Lungimea informatiei
352      */
353     length_data = this->rdlength[0] << 8 | this->rdlength[1];
354     if (length_data == 0)
355     {
356         *data = NULL;
357     }
358     else
359     {
360         *data = new char[length_data];
361         memcpy(*data, this->rdata, length_data);
362     }
363 }
```

Here is the caller graph for this function:

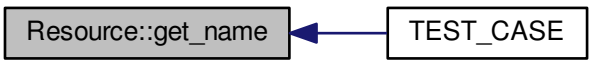


5.28.2.3 void Resource::get_name (char ** *name*, unsigned short & *length*)

```
288 {
289     /* Returneaza numele Resursei
290      *
291      * @param[out] *name
292      *  Numele Resursei
293      *
294      * @param[out] lenght
295      *  Lungimea stringului
296      */
297     if (this->name_len == 0)
298     {
299         *name = NULL;
```

```
300         length = this->name_len;
301     }
302     else
303     {
304         *name = new char[this->name_len];
305         memcpy(*name, this->name, this->name_len);
306         length = this->name_len;
307     }
308 }
```

Here is the caller graph for this function:



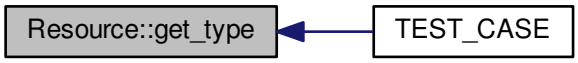
5.28.2.4 void Resource::get_ttl (char * *ttl*)

```
333 {
334      /* Returneaza time tp live pentru o resurse
335      *
336      * @param[out] ttl[4]
337      *  Time to live.
338      */
339
340      memcpy(ttl, this->ttl, 4);
341 }
```

5.28.2.5 void Resource::get_type (char * *type*)

```
311 {
312     /* Returneaza tipul unei resurse
313      *
314      * @param[out] type[2]
315      *  Tipul resursei.
316      */
317
318      memcpy(type, this->type, 2);
319 }
```

Here is the caller graph for this function:

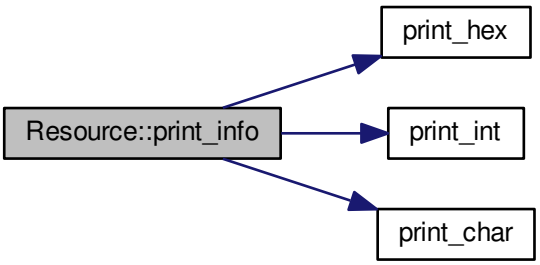


5.28.2.6 void Resource::print_info ( )

```
366 {
```

```
367     /* Printeaza informatii despre resurs */
368     std::cout << "  Name :";
369     print_hex(this->name, this->name_len, true);
370     std::cout << "  Name :";
371     print_int(this->name, this->name_len, true);
372     std::cout << "  Name :";
373     print_char(this->name, this->name_len, true);
374     std::cout <<" Len. Name: " << this->name_len << std::endl;
375 
376     std::cout << "  Type:";
377     print_hex(this->type, 2, true);
378 
379     std::cout << "  Class:";
380     print_hex(this->cls, 2, true);
381 
382     std::cout << "  TTL:";
383     print_hex(this->ttl, 4, true);
384 
385     unsigned short len;
386     memcpy(&len, this->rdlength, 2);
387     len = ntohs(len);
388 
389     std::cout << " Data:";
390     print_hex(this->rdata, len, true);
391 
392     std::cout << " Data:";
393     print_int(this->rdata, len, true);
394 
395     std::cout << " Data:";
396     print_char(this->rdata, len, true);
397 
398     std::cout << "  Len. Data:" << len << std::endl << std::endl;
399 }
```

Here is the call graph for this function:

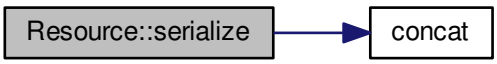


5.28.2.7 void Resource::serialize (char ** *data*, unsigned short & *len*)

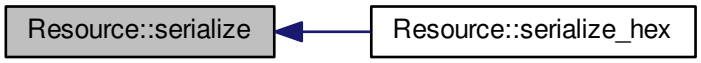
```
402 {
403     /* Serializeaza obiectul curent */
404     char aux[this->name_len];
405     memset(aux, 0, this->name_len);
406     memcpy(aux, this->name, this->name_len);
407     concat(data, data_len, aux, this->name_len, NULL, 0);
408 
409     char aux_2[2];
410     bzero(aux_2, sizeof(aux_2));
411     memcpy(aux_2, this->type, 2);
412     concat(data, data_len, *data, data_len, aux_2, 2);
413 
414     bzero(aux_2, sizeof(aux_2));
415     memcpy(aux_2, this->cls, 2);
416     concat(data, data_len, *data, data_len, aux_2, 2);
417 
418     char aux_4[4];
419     bzero(aux_4, sizeof(aux_4));
420     memcpy(aux_4, this->ttl, 4);
```

```
421     concat(data, data_len, *data, data_len, aux_4, 4);
422
423     /* rdlen */
424     bzero(aux_2, sizeof(aux_2));
425     memcpy(aux_2, this->rdlength, 2);
426     concat(data, data_len, *data, data_len, aux_2, 2);
427
428     /* rdata */
429     unsigned short len;
430     memcpy(&len, this->rdlength, 2);
431     len = ntohs(len);
432
433     char aux_data[len+1];
434     memset(aux_data, 0, len+1);
435     memcpy(aux_data, this->rdata, len);
436     concat(data, data_len, *data, data_len, aux_data, len);
437 }
```

Here is the call graph for this function:



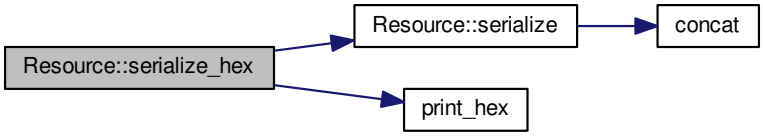
Here is the caller graph for this function:



5.28.2.8 void Resource::serialize_hex ( )

```
440 {
441     /* Afiseaza la stdout hexa serializari obiectului curent */
442     char* data;
443     unsigned short len;
444     this->serialize(&data, len);
445     print_hex(data, len, true);
446 }
```

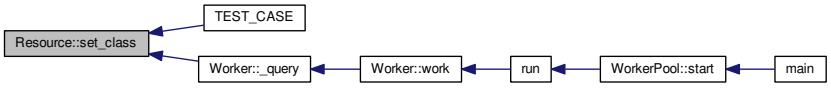
Here is the call graph for this function:



5.28.2.9 void Resource::set_class (char *qclass[2]*)

```
249 {
250     /* Seteaza clasa unei resurse
251      *
252      * @param[in] cls[2]
253      *  Clasa resurseri.
254      */
255
256      memcpy(this->cls, cls, 2);
257
258 }
```

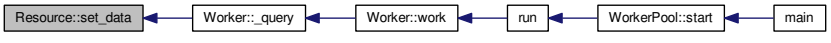
Here is the caller graph for this function:



5.28.2.10 void Resource::set_data (char * *data,* unsigned short *length*)

```
272 {
273     /* Setarea unor informatii despre resursa
274      *
275      * @param[in] data
276      *  Informatia propriuzisa
277      *
278      * @param[in] length_data
279      *  Lungimea informatiei
280      */
281
282     this->rdata = new char[length_data];
283     memcpy(this->rdata, data, length_data);
284     memcpy(this->rdlength, (char*)&length_data, 2);
285 }
```

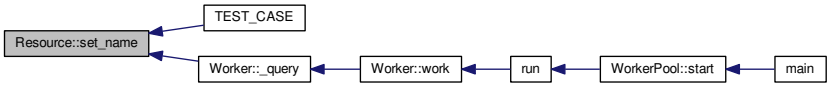
Here is the caller graph for this function:



5.28.2.11 void Resource::set_name (char * *name,* unsigned short *length*)

```
222 {
223     /* Seteaza un nume pentru o resursa
224      *
225      * @param[in] *name
226      *  Numele resursei
227      *
228      * @param[in] lenght
229      *  Lungimea stringului
230      * */
231      this->name_len = length;
232
233      this->name = new char[length];
234      memcpy(this->name, name, length);
235 }
```

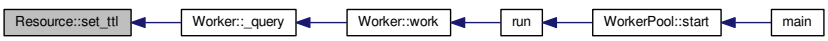
Here is the caller graph for this function:



5.28.2.12 void Resource::set_ttl (char *ttl[4]*)

```
261 {
262     /* Seteaza time to live unei resurse
263      *
264      * @param[in] ttl[4]
265      *  time to live pentru  resurseri.
266      */
267
268      memcpy(this->ttl, ttl, 4);
269 }
```

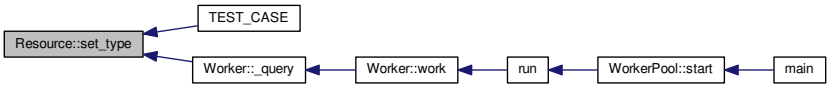
Here is the caller graph for this function:



5.28.2.13 void Resource::set_type (char *type[2]*)

```
238 {
239     /* Seteaza tipul unei resurse
240      *
241      * @param[in] type[2]
242      *  Tipul resurse.
243      */
244
245      memcpy(this->type, type, 2);
246 }
```

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- dns/dns.h
- dns/dns.cpp

5.29 Server Class Reference

`#include <server.h>`

Public Member Functions

- Server (unsigned short port, unsigned int backlog)
- void start ()
- void stop ()
- ssize_t read (void *buf, size_t len, int flags, struct sockaddr *src_addr, socklen_t *addrlen)
- ssize_t send (const void *buf, size_t len, int flags, const struct sockaddr *dest_addr, socklen_t addrlen)
- unsigned short get_port ()

5.29.1 Constructor & Destructor Documentation

5.29.1.1 Server::Server (unsigned short *port*, unsigned int *backlog*)

```
22 {
23     /* Initializeaza un server
24      *
25      * @param[in] port
26      *  Portul pe care o sa deschidem servarul
27      *
28      * @param[in] backlog
29      *  Dimensiunea cozi pentru listen (Folosit doar pentru TCP).
30      * */
31     this->port = port;
32     this->backlog = backlog;
33     this->sock = 0;
34 
35     /* umplem structura folosita de server */
36     bzero(&this->server, sizeof(this->server));
37     this->server.sin_family = AF_INET;
38     server.sin_addr.s_addr = htonl (INADDR_ANY);
39     server.sin_port = htons (this->port);
40 
41 }
```

5.29.2 Member Function Documentation

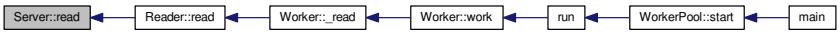
5.29.2.1 unsigned short Server::get_port ( )

5.29.2.2 ssize_t Server::read (void * *buf*, size_t *len*, int *flags*, struct sockaddr * *src_addr*, socklen_t * *addrlen*)

```
79 {
```

```
80      /* Citeste informatii de pe retea.
81       *
82       * @param[out] buf
83       *  Unde o sa salvam datele primite
84       *
85       * @param[in] len
86       *  Lungimea buffarului unde salvam datele
87       *
88       * @param[in] flags
89       *  Flagurile pentru operatia de citire
90       *
91       * @param[out] src_addr
92       *  sockaddr structura umpluta cu informatiile clientului
93       *  de la care am citit date
94       *
95       * @param[in, out] addrlen
96       *  La intreare accepta dimensiune structuri src_addr
97       *  La intreare returneaza dimensiune structuri src_addr completata
98       */
99      if (this->sock == 0)
100     {
101         /* Daca nu am pornit servarul */
102         throw ServerNotOpen();
103     }
104
105     bzero(src_addr, sizeof(src_addr));
106     bzero(buf, len);
107     ssize_t msglen;
108     msglen = recvfrom(this->sock, buf, len, 0, src_addr, addrlen);
109
110     if (msglen < 0)
111     {
112         /* Avem o eroare */
113         perror("Reading from server !\n");
114         throw ServerReadError();
115     }
116
117     return msglen;
118 }
```

Here is the caller graph for this function:

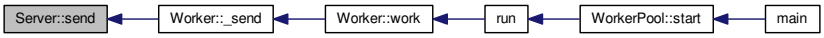


5.29.2.3 ssize_t Server::send (const void * *buf*, size_t *len*, int *flags*, const struct sockaddr * *dest_addr*, socklen_t *addrlen*)

```
122 {
123     /* Trimitem informatii de pe retea.
124      *
125      * @param[in] buf
126      *  Buffer cu infromatiile pe care vrem sa le trimitem
127      *
128      * @param[in] len
129      *  Lungimea buffarului u
130      *
131      * @param[in] flags
132      *  Flagurile pentru operatia de citire
133      *
134      * @param[in] dest_addr
135      *  sockaddr structura umpluta cu informatiile clientului
136      *  (unde vrem sa trimitem)
137      *
138      * @param[in] addrlen
139      *  Lungimea structuri sockaddr
140      */
141     if (this->sock == 0)
142     {
143         /* Daca nu am pornit servarul */
144         throw ServerNotOpen();
145     }
146
147     ssize_t msglen = 1;
148     /* NOTE(mmicu): msglen nu va fi nici o data negativ in
149      * while de asta putem face cast
150      */
```

```
151     while (static_cast<size_t>(msglen) < len)
152     {
153         /* Cat timp nu s-au trimiti toate informatiile */
154         msglen = sendto(this->sock, buf, len, 0,
155                         dest_addr, addrlen);
156 
157         if (msglen < 0)
158         {
159             /* Avem o eroare */
160             perror("Sending from server !\n");
161             throw ServerReadError();
162         }
163     }
164 
165 
166     return msglen;
167 }
```

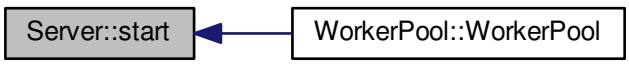
Here is the caller graph for this function:



5.29.2.4 void Server::start ( )

```
44 {
45     /* Porneste un server UDP pe local host */
46     if ((this->sock = socket(this->server.sin_family, SOCK_DGRAM, 0)) == -1)
47     {
48       perror ("[server]Eroare la socket().\n");
49       throw SocketException();
50     }
51 
52     /* Accepta reutilizarea portului */
53     int on=1;
54     setsockopt(this->sock, SOL_SOCKET, SO_REUSEADDR, &on, sizeof(on));
55 
56     /* facem bind la socket cu (adresa, port) */
57     if (bind (this->sock, (struct sockaddr *) &this->server, sizeof (struct sockaddr)) == -1)
58     {
59       perror ("[server]Eroare la bind().\n");
60       throw BindException();
61     }
62 }
```

Here is the caller graph for this function:

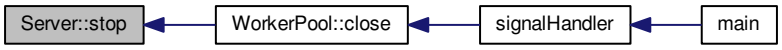


5.29.2.5 void Server::stop ( )

```
65 {
66     /* Opreste servarul */
67     if (this->sock == 0)
68     {
69         /* Nu s-a pornis servarul nici o data */
```

```
70         throw NotOpenException();
71
72     }
73     close(this->sock);
74     this->sock = 0;
75 }
```

Here is the caller graph for this function:



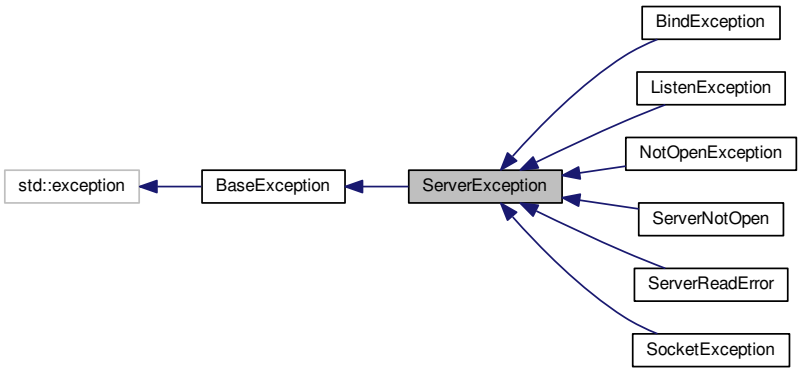
The documentation for this class was generated from the following files:

- dns/server.h
- dns/server.cpp

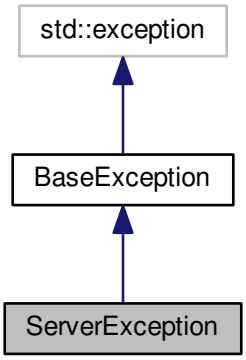
5.30 ServerException Class Reference

`#include <exceptions.h>`

Inheritance diagram for ServerException:



Collaboration diagram for ServerException:



Public Member Functions

- const char * what () const throw ()

5.30.1 Member Function Documentation

5.30.1.1 const char * ServerException::what ( ) const throw)

```
132 {
133     return "Eroare la crearea serverului !";
134 }
```

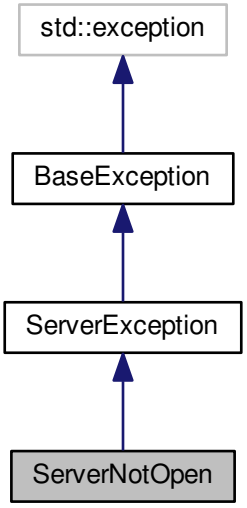
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

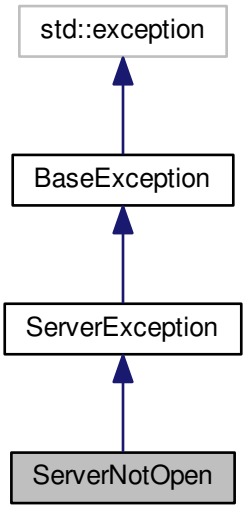
5.31 ServerNotOpen Class Reference

`#include <exceptions.h>`

Inheritance diagram for ServerNotOpen:



Collaboration diagram for ServerNotOpen:



Public Member Functions

- const char ∗ what () const throw ()

5.31.1 Member Function Documentation

5.31.1.1 const char * ServerNotOpen::what ( ) const throw)

```
156 {
157     return "Servarul nu este deschis !";
158 }
```

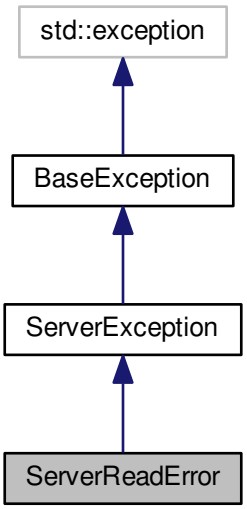
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

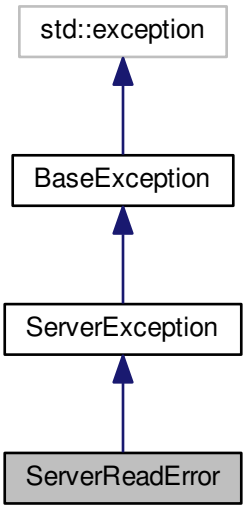
5.32 ServerReadError Class Reference

`#include <exceptions.h>`

Inheritance diagram for ServerReadError:



Collaboration diagram for ServerReadError:



Public Member Functions

- const char * what () const throw ()

5.32.1 Member Function Documentation

5.32.1.1 const char * ServerReadError::what ( ) const throw)

```
161 {
162     return "Eroare la citire !";
163 }
```

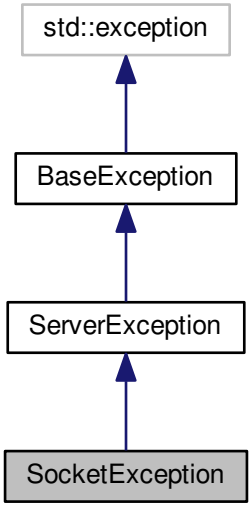
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

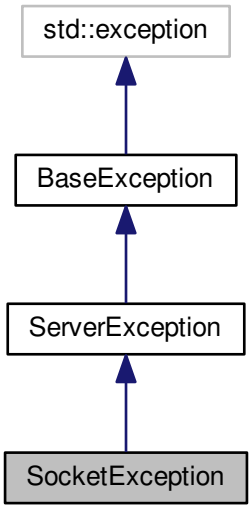
5.33 SocketException Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for SocketException:



Collaboration diagram for SocketException:



Public Member Functions

- const char ∗ what () const throw ()

5.33.1 Member Function Documentation

5.33.1.1 const char * SocketException::what ( ) const throw)

```
137 {
138     return "Eroare la crearea socketului !";
139 }
```

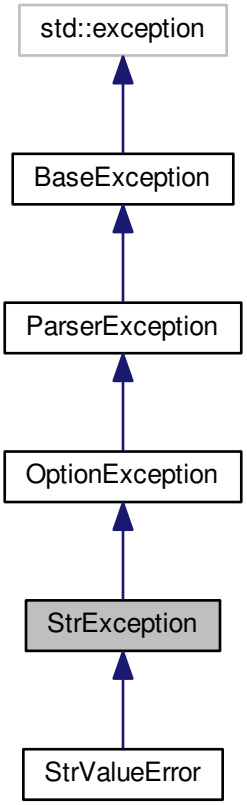
The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

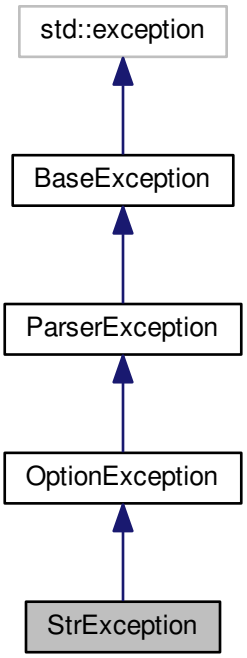
5.34 StrException Class Reference

`#include <exceptions.h>`

Inheritance diagram for StrException:



Collaboration diagram for StrException:



Public Member Functions

- const char * what () throw ()
- StrException (std::string &primit)

Additional Inherited Members

5.34.1 Constructor & Destructor Documentation

5.34.1.1 StrException::StrException (std::string & *primit*)

```
79                                              : OptionException(
      primit)
80 {
81 }
```

5.34.2 Member Function Documentation

5.34.2.1 const char * StrException::what ( ) throw)

```
75 {
76     return "StrException: Base exceptio for StrOption";
77 }
```

The documentation for this class was generated from the following files:

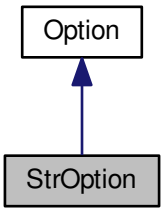
- dns/exceptions.h

- dns/exceptions.cpp

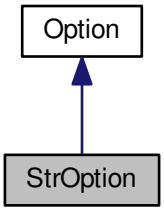
5.35 StrOption Class Reference

`#include <parser.h>`

Inheritance diagram for StrOption:



Collaboration diagram for StrOption:



Public Member Functions

- StrOption (char short_name, std::string long_name, std::string help_message, bool required)
- void set_default (std::string default_value)
- bool is_required ()
- std::string get_string ()
- std::string get_type ()

Protected Member Functions

- void set_value (std::string param_value)

Additional Inherited Members

5.35.1 Constructor & Destructor Documentation

5.35.1.1 StrOption::StrOption (char *short_name*, std::string *long_name*, std::string *help_message*, bool *required*)

```
287                                                                 :
288     Option(short_name, long_name, help_message,
      required)
289 {
290     /* Constructor pentru StrOption
291      *
292      * @param[in] short_name
293      *  Un char care reprezinta varianta prescurtata a parametrului
294      *  Ex: -d
295      *
296      * @param[in] long_name
297      *  Un string care reprezinta varianta lunga aparametrului
298      *  Ex: --delimitator
299      *
300      * @param[in] help_string
301      *  Un string care reprezinta descrierea optiuni
302      *
303      * @param[in] required
304      *  Daca un parametru este necesar sau nu, implicit este setat pe false
305      */
306 }
```

5.35.2 Member Function Documentation

5.35.2.1 std::string StrOption::get_string ( ) [virtual]

Reimplemented from Option.

```
309 {
310     /* \return{Returneaza valoarea parametrului} */
311     return this->value;
312 }
```

5.35.2.2 std::string StrOption::get_type ( ) [virtual]

Reimplemented from Option.

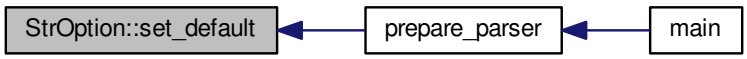
```
338 {
339     /* Returneaza un string care reprezinta tipul optinui */
340     return std::string("str");
341 }
```

5.35.2.3 bool StrOption::is_required ( )

5.35.2.4 void StrOption::set_default (std::string *default_value*)

```
315 {
316     /* Seteaza valoarea default a acestei otiuni
317      *
318      * @param[in] defualt_value
319      *  Valoarea default.
320      * */
321     this->default_value = default_value;
322     this->is_set = true; /* Putem sa o consideram setat */
323     this->required = false; /* nu trebuie sa fie parsata de la CLI */
324 }
```

Here is the caller graph for this function:



5.35.2.5 void StrOption::set_value (std::string *param_value*) [protected], [virtual]

Reimplemented from Option.

```
327 {
328     /* Seteaza valoarea unui parametru convorm unui string
329      *
330      * @param parameter
331      *  Stringul care contine valoarea parametrului
332      */
333     this->value = parameter;
334     this->is_set = true;
335 }
```

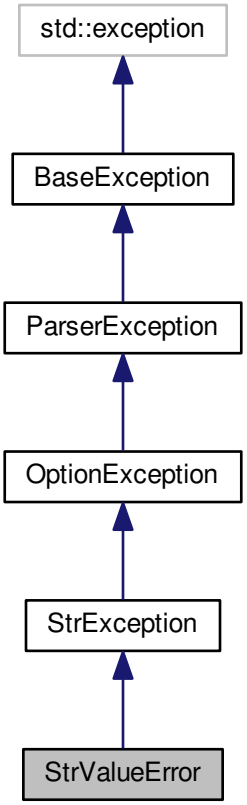
The documentation for this class was generated from the following files:

- dns/parser.h
- dns/parser.cpp

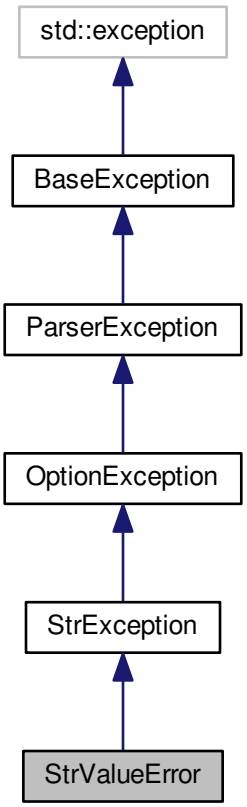
5.36 StrValueError Class Reference

`#include <exceptions.h>`

Inheritance diagram for StrValueError:



Collaboration diagram for StrValueError:



Public Member Functions

- const char * what () throw ()
- StrValueError (std::string &primit)

Additional Inherited Members

5.36.1 Constructor & Destructor Documentation

5.36.1.1 StrValueError::StrValueError (std::string & *primit*)

```
89                                              : StrException(primit)
90 {
91 }
```

5.36.2 Member Function Documentation

5.36.2.1 const char * StrValueError::what ( ) throw)

```
85 {
86     return ("Nu putem parsa " + this->primit).c_str();
87 }
```

The documentation for this class was generated from the following files:

- dns/exceptions.h
- dns/exceptions.cpp

5.37 Tranzaction Class Reference

`#include <dns.h>`

Public Member Functions

- Tranzaction ()
- void set_id (char id[2])
- void set_flags (char flags[2])
- void add_question (Question qt)
- void add_answer (Resource ans)
- void add_authority (Resource aut)
- void add_additiona_section (Resource res)
- void get_id (char id[2])
- void get_flags (char flags[2])
- void get_qcount_char (char qcount[2])
- unsigned short get_qcount_short ()
- void get_ancount_char (char ancount[2])
- unsigned short get_ancount_short ()
- void get_nscount_char (char nscount[2])
- unsigned short get_nscount_short ()
- void get_arcount_char (char arcount[2])
- unsigned short get_arcount_short ()
- void set_client (sockaddr client)
- sockaddr get_client ()
- void set_flag_response ()
- void set_flag_notfound ()
- void print_info ()
- std::vector< Question > get_questions ()
- std::vector< Resource > get_answers ()
- std::vector< Resource > get_authority ()
- std::vector< Resource > get_additional_sections ()
- void serialize (char **data, unsigned short &len)
- void serialize_hex ()

5.37.1 Constructor & Destructor Documentation

5.37.1.1 Tranzaction::Tranzaction ( )

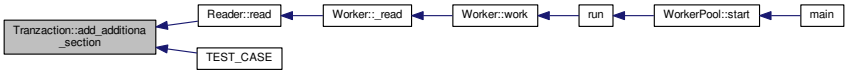
```
449 {
450     /* Initializeaza o tranzactie */
451     bzero(this->id, 2);
452     bzero(this->flags, 2);
453     bzero(this->qcount, 2);
454     bzero(this->ancount, 2);
455     bzero(this->nscount, 2);
456     bzero(this->arcount, 2);
457     this->questions.clear();
458     this->answers.clear();
459     this->authority.clear();
460     this->additional_sections.clear();
461 }
```

5.37.2 Member Function Documentation

5.37.2.1 void Tranzaction::add_additiona_section (Resource *res*)

```
549 {
550     /* Adauga resurse aditionale
551      *
552      * @param res
553      *  Resursa , o instanta a clasei `Resource`
554      */
555
556     this->additional_sections.push_back(res);
557
558     /* Updateaza numarul de autoritati */
559     unsigned short size = this->additional_sections.size();
560     memcpy(this->arcount, (char*)&(size), 2);
561 }
```

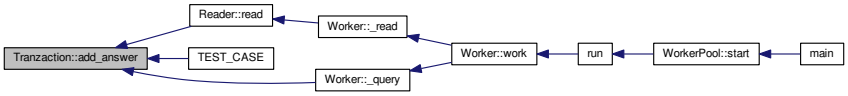
Here is the caller graph for this function:



5.37.2.2 void Tranzaction::add_answer (Resource *ans*)

```
519 {
520     /* Adauga un raspuns
521      *
522      * @param aut
523      *  Raspuns, o instanta a clasei `Resource`
524      */
525
526     this->answers.push_back(ans);
527
528     /* Updateaza numarul de raspunsuri */
529     unsigned short size = this->answers.size();
530     memcpy(this->ancount, (char*)&(size), 2);
531 }
```

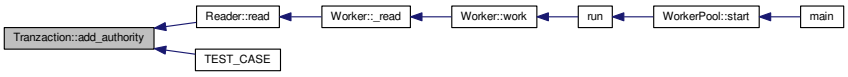
Here is the caller graph for this function:



5.37.2.3 void Tranzaction::add_authority (Resource *aut*)

```
534 {
535     /* Adauga o autoritate tranzactiei
536      *
537      * @param aut
538      *  Autoritatea, o instanta a clasei `Resource`
539      */
540
541     this->authority.push_back(aut);
542
543     /* Updateaza numarul de autoritati */
544     unsigned short size = this->authority.size();
545     memcpy(this->nscount, (char*)&(size), 2);
546 }
```

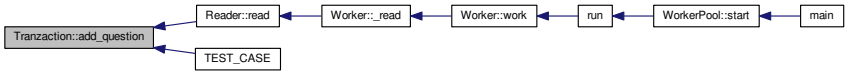
Here is the caller graph for this function:



5.37.2.4 void Tranzaction::add_question (Question *qt*)

```
501 {
502     /* Adauga un query (intrebare) tranzactiei
503      *
504      * @param qt
505      *  Intrebarea, o instanta a clasei `Question`
506      */
507     this->questions.push_back(qt);
508 
509     /* Updateaza numarul de intrebari */
510 
511     /* NOTE(mmicu): Acest lucru ar trebui eliminat, putem avea doar un getter
512      * peste `qcount` care apeleaza `.length` pe `this->questions`
513      */
514     unsigned short size = this->questions.size();
515     memcpy(this->qcount, (char*)&(size), 2);
516 }
```

Here is the caller graph for this function:



5.37.2.5 std::vector< Resource > Tranzaction::get_additional_sections ( )

```
685 {
686     /* Returneaza o lista cu toate resursele aditionale */
687     return this->additional_sections;
688 }
```

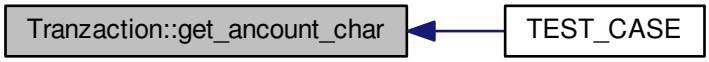
Here is the caller graph for this function:

5.37.2.6 void Tranzaction::get_ancount_char (char *ancount[2]*)

```
605 {
```

```
606     /* Returneza numarul de raspunsuri
607      * sub forma unui char[2]
608      *
609      * @param r_ancount
610      *  Numarul de raspunsuri
611      */
612     memcpy(r_ancount, this->ancount, 2);
613 }
```

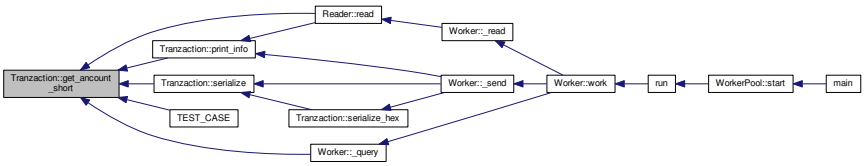
Here is the caller graph for this function:



5.37.2.7 unsigned short Tranzaction::get_ancount_short ( )

```
616 {
617     /* Returneza numarul de raspunsuri
618      * sub forma unui unsigned short
619      */
620     unsigned short count = this->_get_short_from_char(this->ancount);
621     return count;
622 }
```

Here is the caller graph for this function:



5.37.2.8 std::vector< Resource > Tranzaction::get_answers ( )

```
673 {
674     /* Returneaza o lista cu toate raspunsurile */
675     return this->answers;
676 }
```

Here is the caller graph for this function:

5.37.2.9 void Tranzaction::get_arcount_char (char *arcount[2]*)

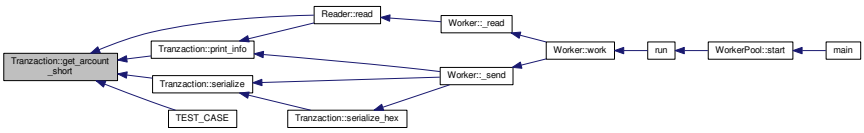
```
646 {
647     /* Returneza numarul de additional records
648      * sub forma unui char[2]
649      *
650      * @param r_arcount
651      *  Numarul de additional records
652      */
653     memcpy(r_arcount, this->nscount, 2);
654 }
```

Here is the caller graph for this function:

5.37.2.10 unsigned short Tranzaction::get_arcount_short ( )

```
657 {
658     /* Returneza numarul de name server authority
659      * sub forma unui unsigned short
660      */
661     unsigned short count = this->_get_short_from_char(this->arcount);
662     return count;
663 }
```

Here is the caller graph for this function:



5.37.2.11 std::vector< Resource > Tranzaction::get_authority ( )

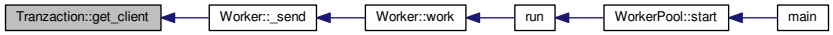
```
679 {
680     /* Returneaza o lista cu toate autoritatile */
681     return this->authority;
682 }
```

Here is the caller graph for this function:

5.37.2.12 sockaddr Tranzaction::get_client ( )

```
701 {
702     /* Returnam clientul */
703     return this->client;
704 }
```

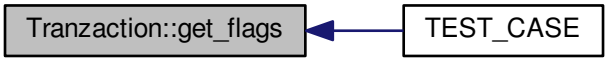
Here is the caller graph for this function:



5.37.2.13 void Tranzaction::get_flags (char *flags[2]*)

```
574 {
575     /* Returneaza flagurile unei tranzactii
576      *
577      * @param[out] flags
578      *  Flagurile tranzactiei
579      */
580     memcpy(flags, this->flags, 2);
581 }
```

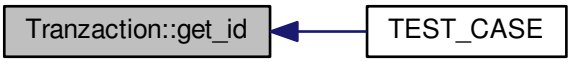
Here is the caller graph for this function:



5.37.2.14 void Tranzaction::get_id (char *id[2]*)

```
564 {
565     /* Returneaza id-ul unei tranzactii
566      *
567      * @param[out] id
568      *  Id-ul tranzactiei
569      */
570     memcpy(id, this->id, 2);
571 }
```

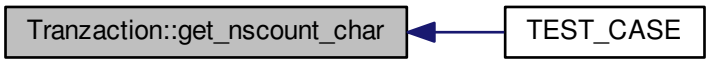
Here is the caller graph for this function:



5.37.2.15 void Tranzaction::get_nscount_char (char *nscount[2]*)

```
626 {
627     /* Returneza numarul de name server authority
628      * sub forma unui char[2]
629      *
630      * @param r_nscount
631      *  Numarul de name server authority
632      */
633     memcpy(r_nscount, this->nscount, 2);
634 }
```

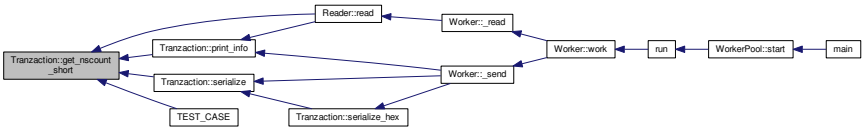
Here is the caller graph for this function:



5.37.2.16 unsigned short Tranzaction::get_nscount_short ( )

```
637 {
638     /* Returneza numarul de name server authority
639      * sub forma unui unsigned short
640      */
641     unsigned short count = this->_get_short_from_char(this->nscount);
642     return count;
643 }
```

Here is the caller graph for this function:



5.37.2.17 void Tranzaction::get_qcount_char (char *qcount[2]*)

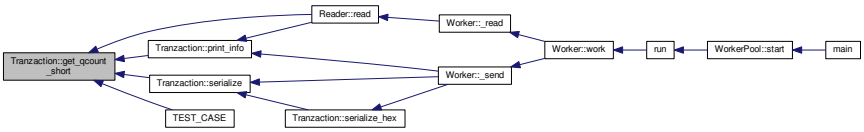
```
584 {
585     /* Returneza numarul de query(intrebari)
586      * sub forma unui char[2]
587      *
588      * @param r_qcount
589      *  Numarul de query
590      */
591     memcpy(r_qcount, this->qcount, 2);
592
593 }
```

Here is the caller graph for this function:

5.37.2.18 unsigned short Tranzaction::get_qcount_short ( )

```
596 {
597     /* Returneza numarul de query(intrebari)
598      * sub forma unui unsigned short
599      */
600     unsigned short count = this->_get_short_from_char(this->qcount);
601     return count;
602 }
```

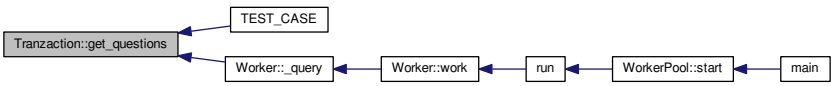
Here is the caller graph for this function:



5.37.2.19 std::vector< Question > Tranzaction::get_questions ( )

```
667 {
668     /* Returneaza o lista cu toate intrebarile */
669     return this->questions;
670 }
```

Here is the caller graph for this function:

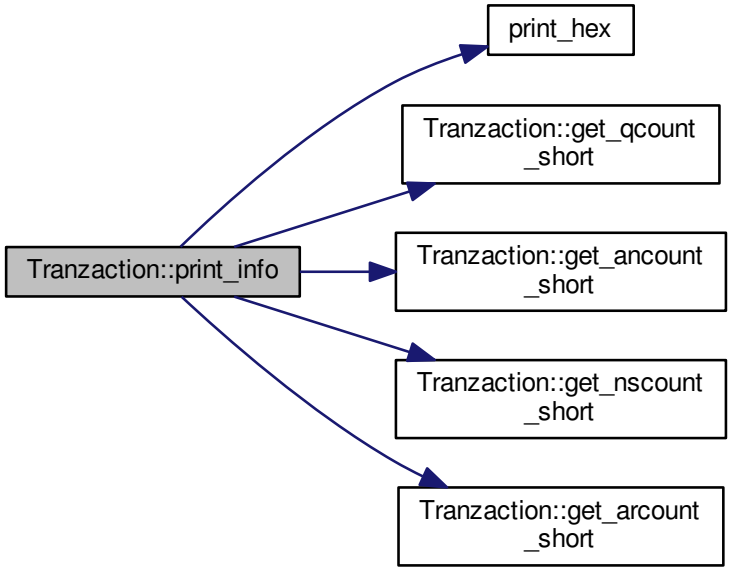


5.37.2.20 void Tranzaction::print_info ( )

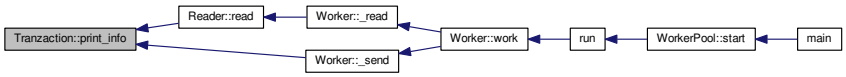
```
720 {
721     /* Printeaza infomatii despre tranzactie */
722     std::cout << " ------------------ " << std::endl;
723     std::cout << "Tranzactie id :";
724     print_hex(this->id, 2, true);
725 
726     std::cout << "Flags :";
727     print_hex(this->flags, 2, true);
```

```
728
729     std::cout << "Qcount :" << this->get_qcount_short() << std::endl;
730     std::cout << "Ancount:" << this->get_ancount_short() << std::endl;
731     std::cout << "Nscount:" << this->get_nscount_short() << std::endl;
732     std::cout << "Arcount :" << this->get_arcount_short() << std::endl;
733
734     std::cout << "Questions :" << std::endl;
735     for (std::vector<Question>::iterator it = this->questions.begin();
736          it != this->questions.end(); ++it)
737     {
738         (*it).print_info();
739     }
740
741     std::cout << "Answers:" << std::endl;
742     for (std::vector<Resource>::iterator it = this->answers.begin();
743          it != this->answers.end(); ++it)
744     {
745         (*it).print_info();
746     }
747
748     std::cout << "Authority:" << std::endl;
749     for (std::vector<Resource>::iterator it = this->authority.begin();
750          it != this->authority.end(); ++it)
751     {
752         (*it).print_info();
753     }
754
755     std::cout << "Additional sections:" << std::endl;
756     for (std::vector<Resource>::iterator it = this->additional_sections.begin();
757          it != this->additional_sections.end(); ++it)
758     {
759         (*it).print_info();
760     }
761
762     std::cout << " ------------------- " << std::endl << std::endl;
763 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

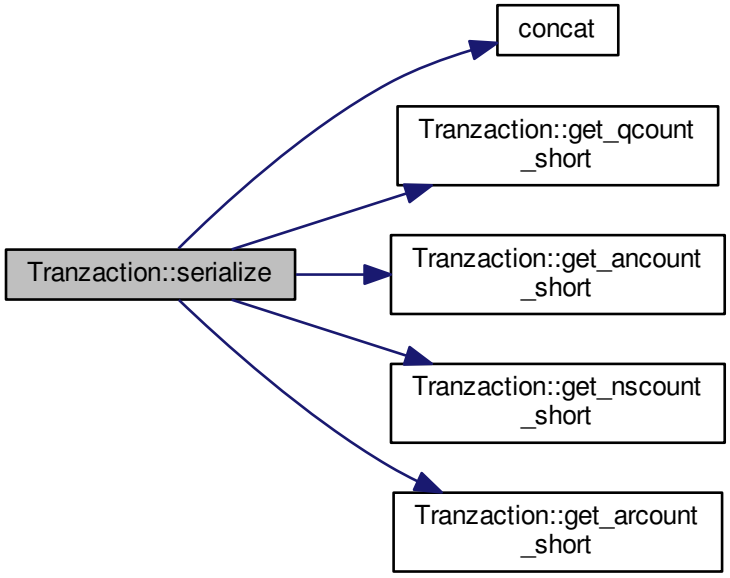


5.37.2.21 void Tranzaction::serialize (char ** *data*, unsigned short & *len*)

```
766 {
767     /* Serializeaza obiectul curent
768      *
769      * @param[out] data
770      * pointer catre array
771      *
772      * @param[out] len
773      * lungimea
774      */
775
776     /* id */
777     char aux_2[3];
778     bzero(aux_2, sizeof(aux_2));
779     memcpy(aux_2, this->id, 2);
780     concat(data, len, aux_2, 2, NULL, 0);
781
782     /* flags */
783     bzero(aux_2, sizeof(aux_2));
784     memcpy(aux_2, this->flags, 2);
785     concat(data, len, *data, len, aux_2, 2);
786
787     unsigned short s = htons(this->get_qcount_short());
788     bzero(aux_2, sizeof(aux_2));
789     memcpy(aux_2, (char*)&s, 2);
790     concat(data, len, *data, len, aux_2, 2);
791
792
793     s = htons(this->get_ancount_short());
794     bzero(aux_2, sizeof(aux_2));
795     memcpy(aux_2, (char*)&s, 2);
796     concat(data, len, *data, len, aux_2, 2);
797
798     s = htons(this->get_nscount_short());
799     bzero(aux_2, sizeof(aux_2));
800     memcpy(aux_2, (char*)&s, 2);
801     concat(data, len, *data, len, aux_2, 2);
802
803     s = htons(this->get_arcount_short());
804     bzero(aux_2, sizeof(aux_2));
805     memcpy(aux_2, (char*)&s, 2);
806     concat(data, len, *data, len, aux_2, 2);
807
808     char *aux_data;
809     unsigned short aux_data_len;
810
811     for (std::vector<Question>::iterator it = this->questions.begin();
812          it != this->questions.end(); ++it)
813     {
814         (*it).serialize(&aux_data, aux_data_len);
815         concat(data, len, *data, len, aux_data, aux_data_len);
816     }
817
818     for (std::vector<Resource>::iterator it = this->answers.begin();
819          it != this->answers.end(); ++it)
820     {
821         (*it).serialize(&aux_data, aux_data_len);
822         concat(data, len, *data, len, aux_data, aux_data_len);
823     }
824
825     for (std::vector<Resource>::iterator it = this->authority.begin();
826          it != this->authority.end(); ++it)
827     {
828         (*it).serialize(&aux_data, aux_data_len);
829         concat(data, len, *data, len, aux_data, aux_data_len);
830     }
831
832     for (std::vector<Resource>::iterator it = this->additional_sections.begin();
833          it != this->additional_sections.end(); ++it)
834     {
```

```
835         (*it).serialize(&aux_data, aux_data_len);
836         concat(data, len, *data, len, aux_data, aux_data_len);
837     }
838 }
```

Here is the call graph for this function:

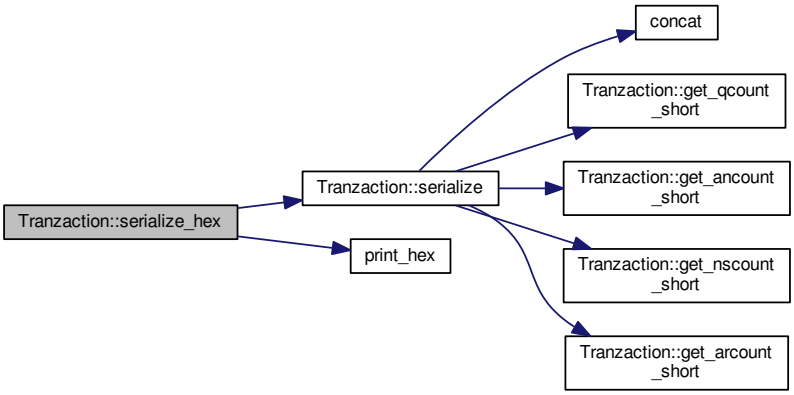


Here is the caller graph for this function:

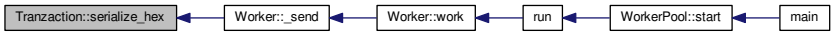
5.37.2.22 void Tranzaction::serialize_hex ( )

```
841 {
842     /* Afiseaza la stdout hexa serializari obiectului curent */
843     char* data;
844     unsigned short len;
845     this->serialize(&data, len);
846     print_hex(data, len, true);
847 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.37.2.23 void Tranzaction::set_client (sockaddr *client*)

```
691 {
692     /* Setam un client pentru tranzactia asta
693      *
694      * @param[in] client
695      *  Clientul pe care dorim sa il setam
696      */
697     this->client = client;
698 }
```

Here is the caller graph for this function:

5.37.2.24 void Tranzaction::set_flag_notfound ( )

```
713 {
714     /* Nu am gasit numele */
715     this->flags[1] |= 1;
716     this->flags[1] |= 2;
717 }
```

Here is the caller graph for this function:

5.37.2.25 void Tranzaction::set_flag_response ( )

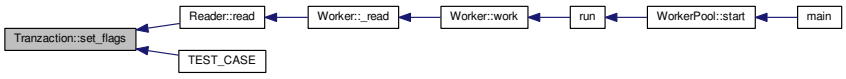
```
707 {
708     /* Setam flagul pentru aceasta tranzactie sa fie un response */
709     this->flags[0] |= 128;
710 }
```

Here is the caller graph for this function:

5.37.2.26 void Tranzaction::set_flags (char *flags[2]*)

```
491 {
492     /* Seteaza flagurile unei tranzactii
493      *
494      * @param[in] flags
495      *  Flagurile tranzactiei
496      * */
497     memcpy(this->flags, flags, 2);
498 }
```

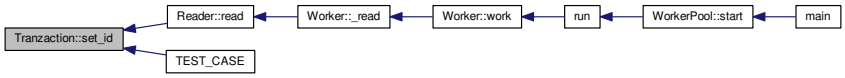
Here is the caller graph for this function:



5.37.2.27 void Tranzaction::set_id (char *id[2]*)

```
481 {
482     /* Seteaza id-ul unei tranzactii
483      *
484      * @param[in] id
485      *  Id-ul tranzactiei
486      */
487     memcpy(this->id, id, 2);
488 }
```

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- dns/dns.h
- dns/dns.cpp

5.38 Worker Class Reference

`#include <worker.h>`

Public Member Functions

- Worker (char *db_name, std::mutex *lock, Reader *rd, Parser *pr, int id)
- ~Worker ()
- Tranzaction * _read ()
- void _query (Tranzaction *tr)
- void _send (Tranzaction *tr)
- void work ()
- void sign_stop ()

5.38.1 Constructor & Destructor Documentation

5.38.1.1 Worker::Worker (char * *db_name*, std::mutex * *lock*, Reader * *rd*, Parser * *pr*, int *id*)

```
33                                                                              :
34     parser(pr)
35 {
36     /* Pregatim un worker
37      *
38      * @param[in] db_name
39      *  Numele bazei de date
40      *
41      * @param[in] *lock
42      *  Lock-ul pentru pool
43      *
44      * @param[in] *rd
45      *  Un obiect de tip reader
46      *
47      * @param[in] *pr
48      *  Un parser sa vedem optiunile de la CLI
49      *
50      * @param[in] id
51      *  Id-ul pentru acest worker
52      */
53     this->id = id;
54     std::cout << "Pornim Worker - "<< this->id << std::endl;
55     this->parser = pr;
56     this->db = new DB(db_name);
57     this->lock = lock;
58
59     this->rd = rd;
60     this->stop = false;
61 }
```

5.38.1.2 Worker::~Worker ( )

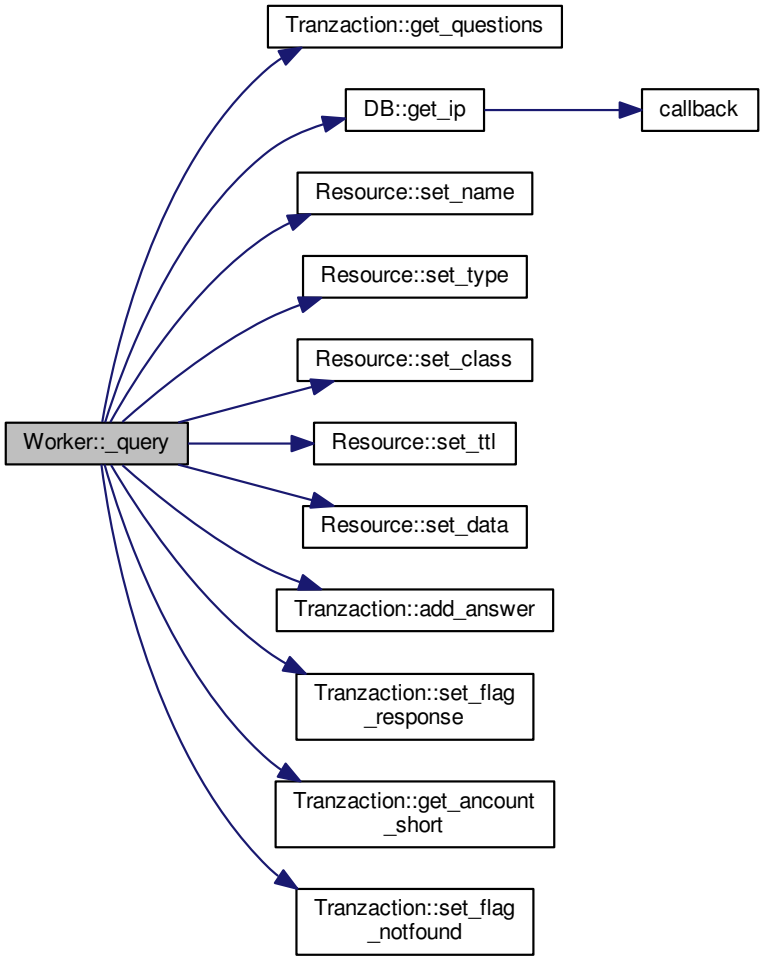
```
64 {
65     /* Deletam tot */
66     delete this->db;
67 }
```

5.38.2 Member Function Documentation

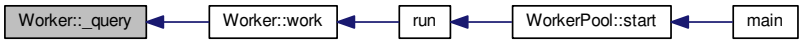
5.38.2.1 void Worker::_query (Tranzaction * *tr*)

```
119 {
120     /* Pregateste raspunsurile
121      *
122      * @param[in, out] tr
123      *  Tranzactia care are intrebarile, la iesire
124      *  va fi populata cu raspunsuri
125      */
126
127     std::vector<Question> aux = tr->get_questions();
128     for (std::vector<Question>::iterator it = aux.begin();
129          it != aux.end(); ++it)
130     {
131         char *domain;
132         unsigned short len;
133         (*it).get_name(&domain, len);
134         std::string ip = this->db->get_ip(domain, len);
135
136
137         if (ip != std::string(""))
138         {
139             /* Daca am gasit un ip */
140             Resource res;
141             res.set_name(domain, len);
142
143             char generic[2]; /* generic 00 01 */
144             generic[0] = 0;
145             generic[1] = 1;
146
147             res.set_type(generic);
148             res.set_class(generic);
149
150             char generic_ttl[4]; /* generic 00 00 00 30 */
151             generic_ttl[0] = 0;
152             generic_ttl[1] = 0;
153             generic_ttl[2] = 0;
154             generic_ttl[3] = 30;
155
156             res.set_ttl(generic_ttl);
157
158             in_addr_t ip_addr = inet_addr(ip.c_str());
159             /* NOTE(mmicu): 4 e hardcodat, desi asta o sa fie dimensiuena unui IPv4
160              * mereu ar trebui sa folosim sizeof */
161
162             std::cout << "ip_addr" << ip_addr << std::endl;
163             unsigned short len = 4;
164             len = htons(len);
165             res.set_data((char*)&ip_addr, len);
166
167             tr->add_answer(res);
168         }
169
170         delete domain;
171     }
172
173     tr->set_flag_response();
174     if (tr->get_ancount_short() == 0)
175     {
176         /* Nu avem nici un raspuns */
177         tr->set_flag_notfound();
178     }
179 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

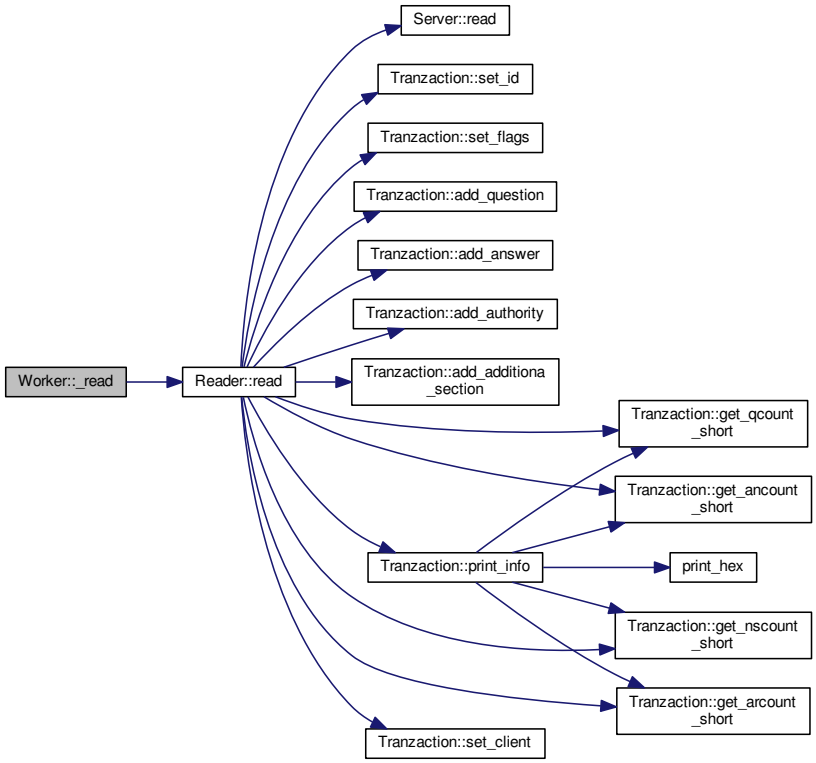


5.38.2.2 Tranzaction * Worker::_read ( )

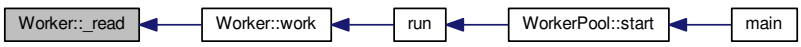
```
102 {
103     /* Citim o noua tranzactie */
104     Tranzaction *tr = NULL;
105     try
106     {
107         tr = this->rd->read();
108     }
109     catch (BaseException* ex)
```

```
110     {
111         std::cout << "[" << this->id << "] Eroare la citire !" << std::endl;
112     }
113 
114 
115     return tr;
116 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

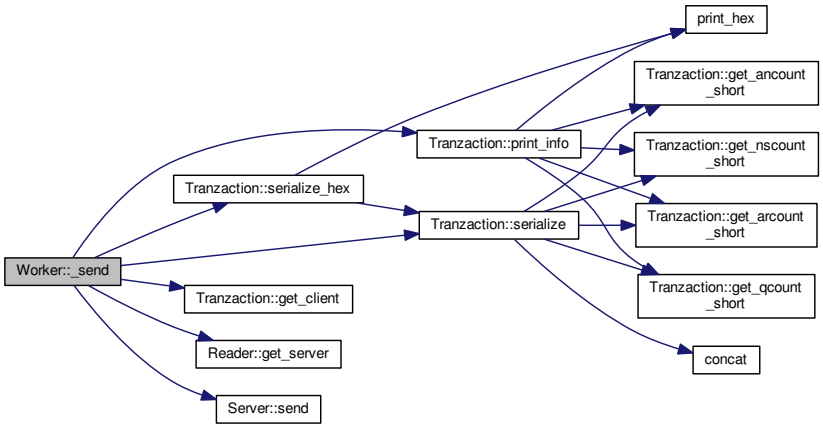


5.38.2.3 void Worker::_send (Tranzaction * *tr*)

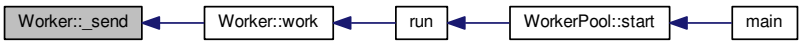
```
182 {
183     /* Serializeaza tranzactia si trimite raspunsul */
184     tr->print_info();
185 
186     char* data;
187     unsigned short data_len;
188     tr->serialize(&data, data_len);
189 
190     std::cout << "Serializare :";
191     tr->serialize_hex();
192 
```

```
193     struct sockaddr client = tr->get_client();
194
195
196     try
197     {
198         this->rd->get_server()->send(data, data_len, 0, (struct sockaddr*)&client, sizeof(
    client));
199     }
200     catch (BaseException* ex)
201     {
202         std::cout << "[" << this->id << "] Trimitere - eroare" << std::endl;
203     }
204
205
206 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.38.2.4 void Worker::sign_stop ( )

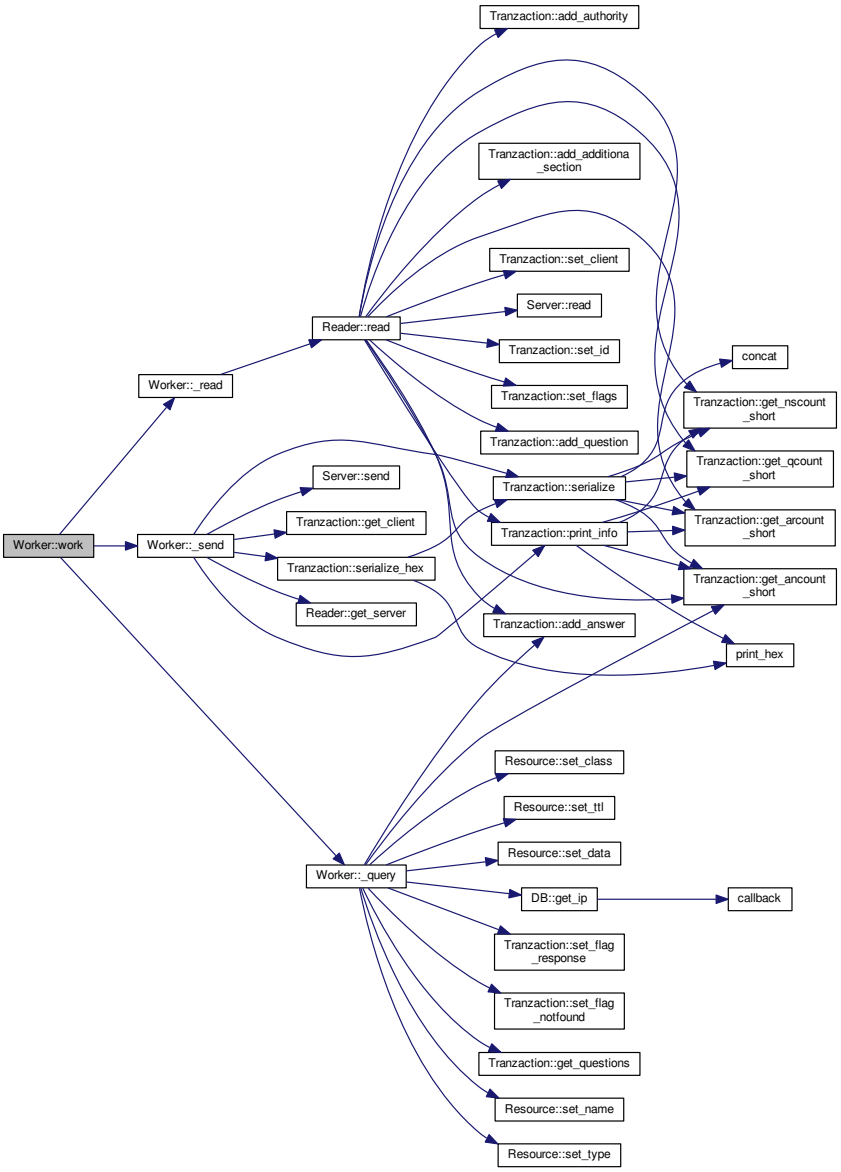
```
70 {
71     /* Trimite semnalul de oprire */
72     std::cout << "Trimitem semnanul de oprire" << std::endl;
73     this->stop = true;
74 }
```

5.38.2.5 void Worker::work ( )

```
77 {
78     /* Realizam un ciclu */
79     std::cout << "[" << this->id << "] Work apel !" << std::endl;
80     while (this->stop == false)
81     {
82         std::cout << "[" << this->id << "] Lock Work !" << std::endl;
83         this->lock->lock();
```

```
84          /* Citim o tranzactie */
85          Tranzaction* tr = this->_read();
86
87          /* Raspundem la tranzactie */
88          this->_query(tr);
89
90          /* Trimitem informatiile serializate */
91          this->_send(tr);
92
93          /* Eliberam memoria */
94          delete tr;
95
96          std::cout << "[" << this->id << "] Unlock Work !" << std::endl;
97          this->lock->unlock();
98      }
99  }
```

Here is the call graph for this function:



Here is the caller graph for this function:

The documentation for this class was generated from the following files:

- dns/worker.h
- dns/worker.cpp

5.39 WorkerPool Class Reference

`#include <worker.h>`

Public Member Functions

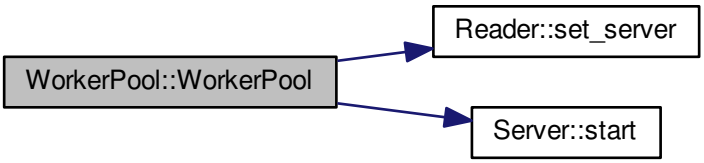
- WorkerPool (Parser *pr)
- void close ()
- void start ()

5.39.1 Constructor & Destructor Documentation

5.39.1.1 WorkerPool::WorkerPool (Parser * *pr*)

```
209 {
210     /* Pregatim Pool-ul de workers
211      *
212      * @param pr
213      * Un obiect de tip parser
214      */
215     std::cout << "Pornim WorkerPool" << std::endl;
216     this->pr = pr;
217 
218     this->server = new Server((*this->pr)["port"]->get_int(),
219                               (*this->pr)["backlog"]->get_int());
220     this->rd = new Reader();
221     this->rd->set_server(this->server);
222 
223     this->server->start();
224 
225 
226     std::cout << "Generam workes  (" <<
227                  (*this->pr)["min_threads"]->get_int() <<
228                  "):" << std::endl;
229     for (int i = 0; i<(*this->pr)["min_threads"]->get_int(); ++i)
230     {
231         /* Generam numarul de workers */
232         std::cout <<" Suntem la " << i << std::endl;
233         this->workers.push_back(
234             new Worker((char*)(*this->pr)["db_name"]->get_string().c_str(),
235                 &(this->lock), this->rd, this->pr, i));
236     }
237 }
```

Here is the call graph for this function:

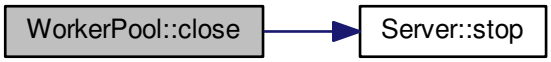


5.39.2 Member Function Documentation

5.39.2.1 void WorkerPool::close ( )

```
261 {
262     std::cout << "Inchide " << std::endl;
263     this->finish = true;
264     /* Trimitem semnalul de oprire */
265     for (std::vector<Worker*>::iterator it = this->workers.begin();
266         it < this->workers.end(); ++it)
267     {
268         (*it)->sign_stop();
269     }
270
271     /* Asteptam dupa fiecare thread */
272     for (std::vector<std::thread*>::iterator it = this->threads.begin();
273         it < this->threads.end(); ++it)
274     {
275         (*it)->join();
276     }
277
278
279     /* Inchidem servarul */
280     this->lock.lock();
281     this->server->stop();
282     this->lock.unlock();
283 }
```

Here is the call graph for this function:

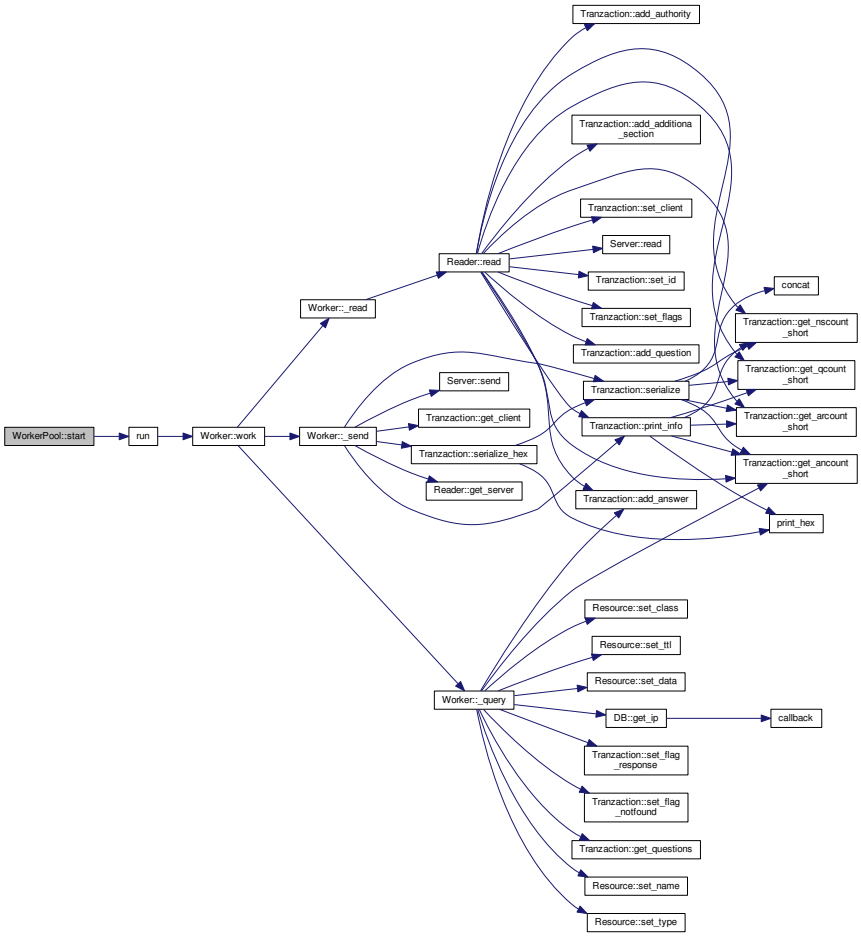


Here is the caller graph for this function:

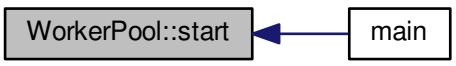
5.39.2.2 void WorkerPool::start ( )

```
240 {
241     /* Pornim Pool-ul */
242     std::cout << "Generam threads :" << std::endl;
243     /* Spawnam threadurile */
244     for (std::vector<Worker*>::iterator it = this->workers.begin();
245         it < this->workers.end(); ++it)
246     {
247         this->threads.push_back(
248             new std::thread(run, (*it)));
249     }
250 
251     this->finish = false;
252 
253     std::cout << "Start !" << std::endl;
254     while (this->finish == false)
255     {
256         /* Astepata */
257     }
258 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- dns/worker.h
- dns/worker.cpp

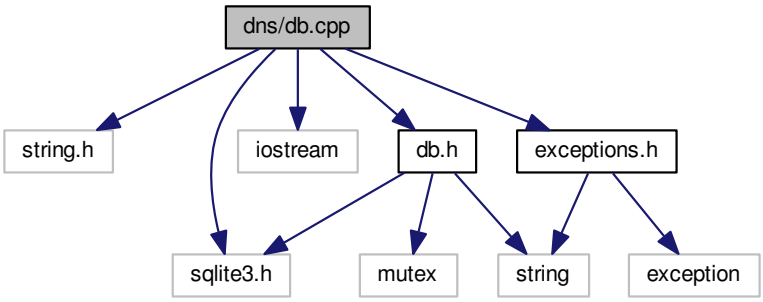
Chapter 6

File Documentation

6.1 dns/db.cpp File Reference

```
#include <string.h>
#include <sqlite3.h>
#include <iostream>
#include "db.h"
#include "exceptions.h"
```

Include dependency graph for db.cpp:



Functions

- static int callback (void *data, int colnum, char **field_data, char **field_name)

Variables

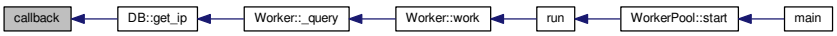
- char TABLE_NAME [] = "my_dns"
- char DOMAIN [] = "domain"
- char IP [] = "ip"

6.1.1 Function Documentation

6.1.1.1 static int callback (void * *data*, int *colnum*, char ** *field_data*, char ** *field_name*) [static]

```
20 {
21     /* Functia de callback pentru SQLite3
22      *
23      * @param[in, out] data
24      *  Parametru setat de noi
25      *
26      * @param[in] rownum
27      *  Numarul de coloane
28      *
29      * @param[in] field_data
30      *  Array cu informatiile din coloane
31      *
32      * @param[in] field_name
33      *  Array cu numele coloanelor
34      */
35     if (colnum == 1)
36     {
37         /* Verificam existenta */
38         if (atoi(field_data[0]) == 0)
39         {
40             memset(data, '0', DB::IP_MAX_SIZE);
41         }
42         else
43         {
44             memset(data, '1', DB::IP_MAX_SIZE);
45         }
46     }
47     else
48     {
49         /* Returnam un IP */
50         if (colnum != 2 || (strcmp(field_name[0], DOMAIN) != 0) ||
51             strcmp(field_name[1], IP) != 0)
52         {
53             /* Nu avem numarul dorit de coloane */
54             return 1;
55         }
56
57         strcpy((char*)data, field_data[1]);
58     }
59     return 0;
60 }
```

Here is the caller graph for this function:



6.1.2 Variable Documentation

6.1.2.1 char DOMAIN[ ] = "domain"

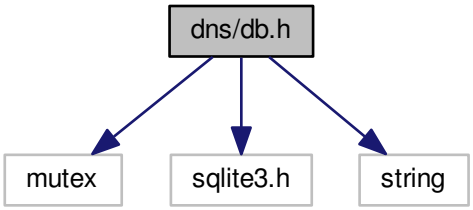
6.1.2.2 char IP[ ] = "ip"

6.1.2.3 char TABLE_NAME[ ] = "my_dns"

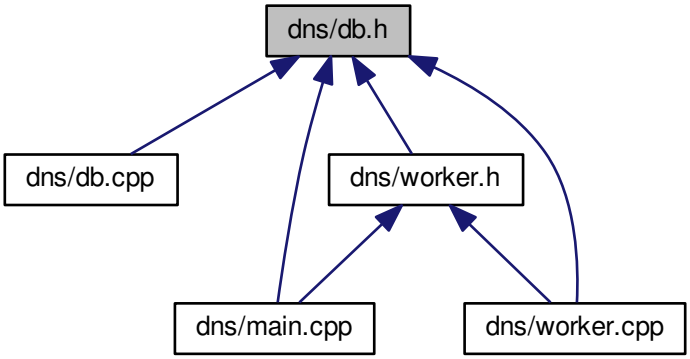
6.2 dns/db.h File Reference

```
#include <mutex>
#include <sqlite3.h>
#include <string>
```

Include dependency graph for db.h:



This graph shows which files directly or indirectly include this file:



Classes

- class DB

Macros

- #define DB_H value

6.2.1 Macro Definition Documentation

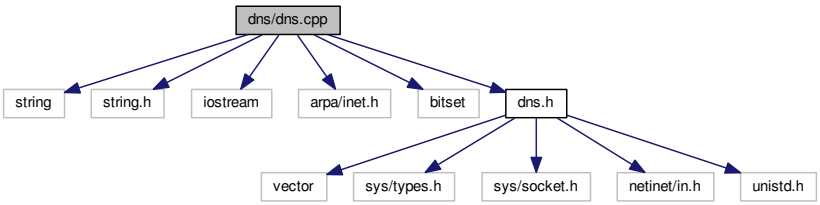
6.2.1.1 #define DB_H value

6.3 dns/dns.cpp File Reference

```
#include <string>
```

```
#include <string.h>
#include <iostream>
#include <arpa/inet.h>
#include <bitset>
#include "dns.h"
```

Include dependency graph for dns.cpp:



Functions

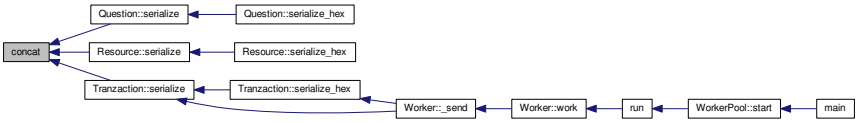
- void print_hex (char *ch, int len, bool endl)
- void concat (char **data, unsigned short &len, char *part1, unsigned short len_1, char *part2, unsigned short len_2)
- void print_char (char *ch, int len, bool endl)
- void print_int (char *ch, int len, bool endl)

6.3.1 Function Documentation

6.3.1.1 void concat (char ** *data*, unsigned short & *len*, char * *part1*, unsigned short *len_1*, char * *part2*, unsigned short *len_2*)

```
30 {
31     /* Concataneaza 2 char* in unul nou */
32     char *data_aux= new char[len_1 + len_2];
33     memcpy(data_aux, part1, len_1);
34     memcpy((data_aux+len_1), part2, len_2);
35 
36     *data = data_aux;
37     len = len_1 + len_2;
38 }
```

Here is the caller graph for this function:

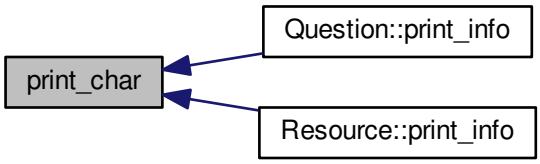


6.3.1.2 void print_char (char * *ch*, int *len*, bool *endl*)

```
41 {
42     /* Afiseaxa caracter cu caracter */
43     for (int i = 0; i < len; ++i)
44     {
45         std::cout << ch[i] << " ";
```

```
46     }
47
48     if (endl == true)
49     {
50         std::cout << std::endl;
51     }
52 }
```

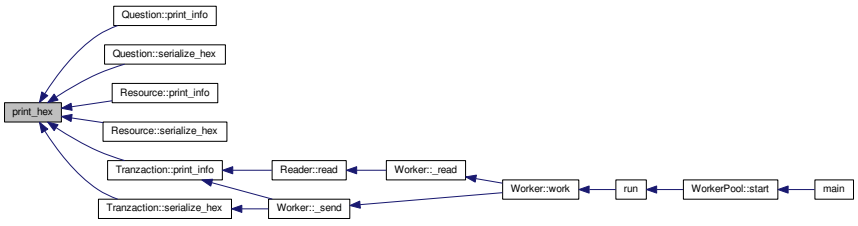
Here is the caller graph for this function:



6.3.1.3 void print_hex (char * *ch*, int *len*, bool *endl*)

```
15 {
16     /* Afiseaxa caracter cu caracter in hexa */
17     for (int i = 0; i < len; ++i)
18     {
19         std::cout << std::hex << (int)ch[i] << " ";
20     }
21
22     if (endl == true)
23     {
24         std::cout << std::endl;
25     }
26 }
```

Here is the caller graph for this function:

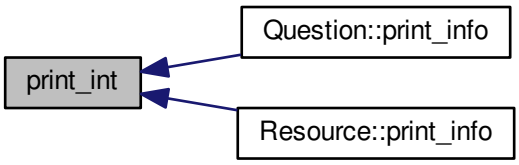


6.3.1.4 void print_int (char * *ch*, int *len*, bool *endl*)

```
55 {
56     /* Afiseaxa caracter cu caracter in int*/
57     for (int i = 0; i < len; ++i)
58     {
59         std::cout << (int)ch[i] << " ";
60     }
61
62     if (endl == true)
63     {
64         std::cout << std::endl;
```

```
65     }
66 }
```

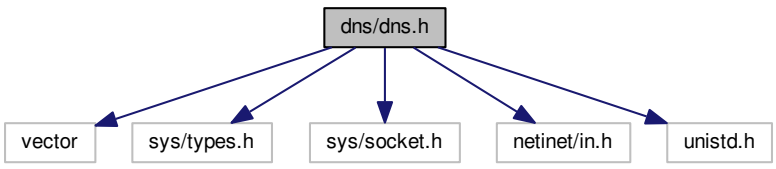
Here is the caller graph for this function:



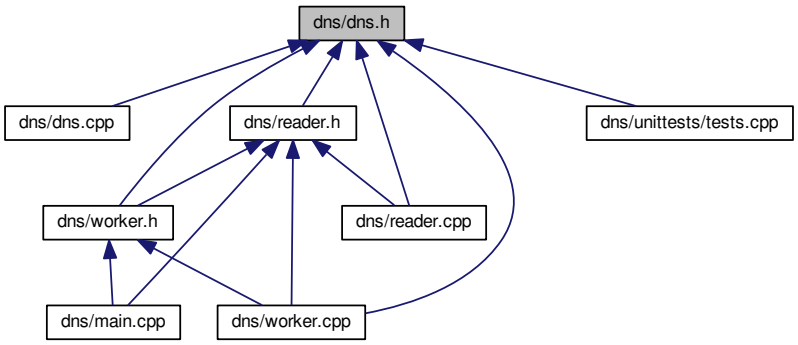
6.4 dns/dns.h File Reference

```
#include <vector>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
```

Include dependency graph for dns.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Question
- class Resource
- class Tranzaction

Macros

- #define DNS_H value

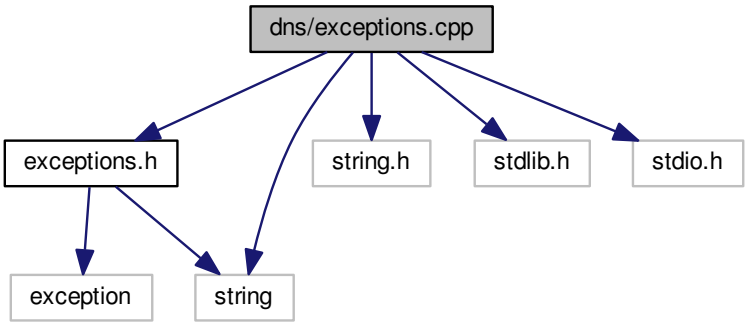
6.4.1 Macro Definition Documentation

6.4.1.1 #define DNS_H value

6.5 dns/exceptions.cpp File Reference

```
#include "exceptions.h"
#include <string>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
```

Include dependency graph for exceptions.cpp:



Functions

- const char * what () throw ()

6.5.1 Function Documentation

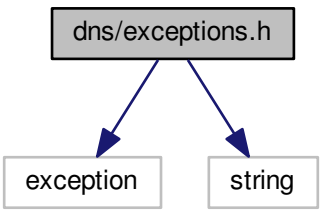
6.5.1.1 const char* what ( ) throw)

```
32 {
33     return "OptionrException: Base exception for option";
34 }
```

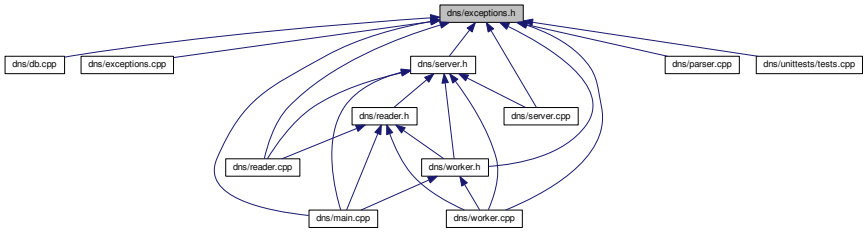
6.6 dns/exceptions.h File Reference

```
#include <exception>
#include <string>
```

Include dependency graph for exceptions.h:



This graph shows which files directly or indirectly include this file:



Classes

- class BaseException
- class ParserException
- class OptionException
- class IntException
- class IntValueError
- class StrException
- class StrValueError
- class BoolException
- class BoolValueError
- class InvalidOptionException
- class ArgumentsLeft
- class NotTheRightType
- class ServerException
- class SocketException
- class BindException
- class ListenException
- class NotOpenException
- class ServerNotOpen
- class ServerReadError
- class ReaderError
- class ReaderValueError
- class DBException
- class DBConnectionException
- class DBCreateException
- class DBSelectException
- class DBMalformedTable

Macros

- #define EXCEPTIONS_H value

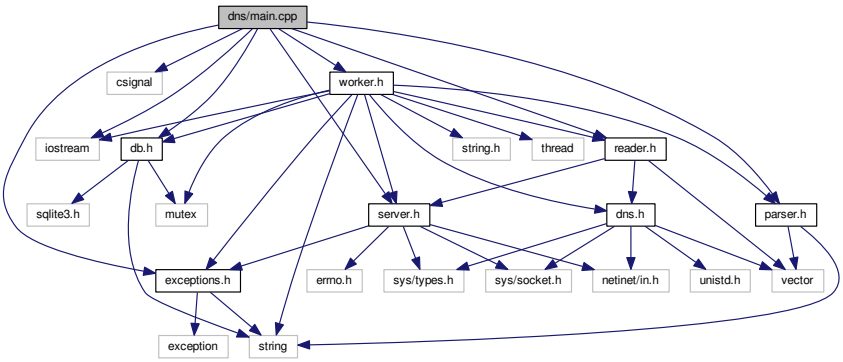
6.6.1 Macro Definition Documentation

6.6.1.1 #define EXCEPTIONS_H value

6.7 dns/main.cpp File Reference

```
#include <iostream>
#include <csignal>
#include "parser.h"
#include "exceptions.h"
#include "server.h"
#include "reader.h"
#include "db.h"
#include "worker.h"
```

Include dependency graph for main.cpp:



Functions

- void signalHandler (int signum)
- Parser prepare_parser ()
- int main (int argc, char ∗argv[])

Variables

- WorkerPool ∗ MAIN = NULL

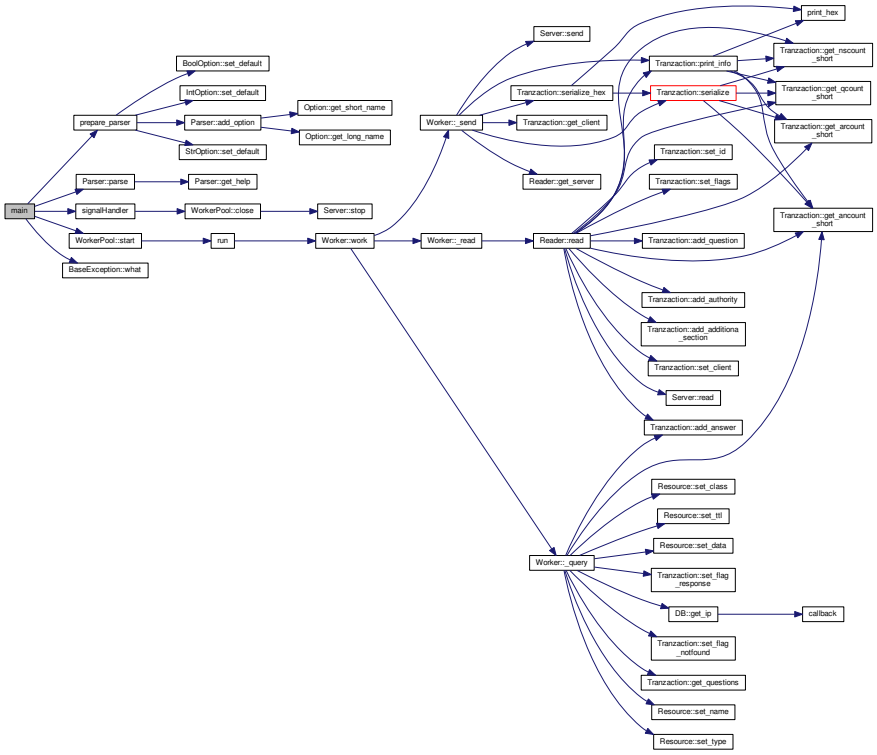
6.7.1 Function Documentation

6.7.1.1 int main (int *argc*, char ∗ *argv[]*)

```
71 {
72     try
73     {
74         Parser p = prepare_parser();
75         p.parse(argc, argv);
76 
77         if (p["soft"]->get_bool() == true)
78         {
79             /* Inregistreaza handler pentru CTRL+C */
80             signal(SIGINT, signalHandler);
81 
82             /* Inregistreaza handler pentru pentru gracefully shutdown */
83             signal(SIGTERM, signalHandler);
84         }
85 
86 
87         WorkerPool pool(&p);
88         MAIN = &pool;
89         pool.start();
90 
```

```
91      }
92      catch (BaseException& ex)
93      {
94          std::cout << "O exceptie nu a fost prinsa !" << std::endl;
95          ex.what();
96      }
97
98      return 0;
99  }
```

Here is the call graph for this function:

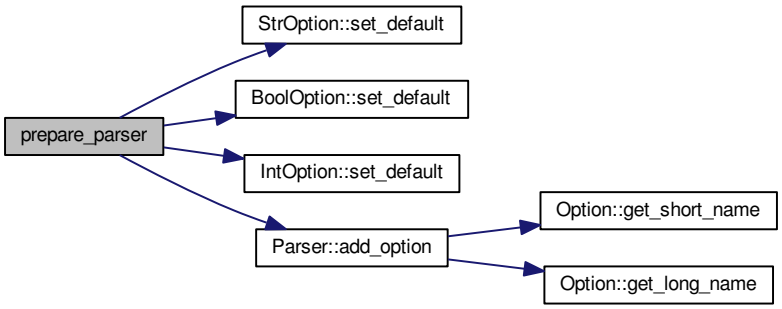


6.7.1.2 Parser prepare_parser ( )

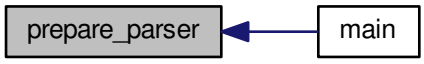
```
37  {
38      /* Pregateste parserul cu toate valorile */
39      Parser parser;
40      StrOption* verbosity = new StrOption('v', "verbose", "Verbosity level(v, vv, vvv).",
        false);
41      verbosity->set_default("v");
42
43      BoolOption* debug = new BoolOption('d', "debug", "Debug mode(default false).",
        false);
44      debug->set_default(false);
45
46      IntOption* port = new IntOption('p', "port", "The port.", false);
47
48      IntOption* min_threads = new IntOption(
49          't', "min_threads", "The min. number of threads, or the base threads.", true);
50
51      IntOption* backlog = new IntOption(
52          'b', "backlog", "The size of the listen queue ( defaut 10).", false);
53      backlog->set_default(10);
54
55
56      StrOption* db_name = new StrOption('f', "db_name", "Fisierul cu baza de date", true);
57      BoolOption* soft = new BoolOption(
58          's', "soft", "Daca dorim sa asteptam fiecare thread inainte de terminare.", true);
59
60      parser.add_option(verbosity);
61      parser.add_option(debug);
```

```
62     parser.add_option(port);
63     parser.add_option(min_threads);
64     parser.add_option(db_name);
65     parser.add_option(soft);
66     parser.add_option(backlog);
67     return parser;
68 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

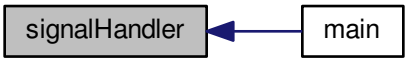


6.7.1.3 void signalHandler (int *signum*)

```
22 {
23     std::cout << "Interrupt signal (" << signum << ") received." << std::endl;
24
25     if (MAIN == NULL)
26     {
27         std::cout << "Inca nu s-a pornit servarul " << std::endl;
28     }
29     else
30     {
31         MAIN->close();
32     }
33 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



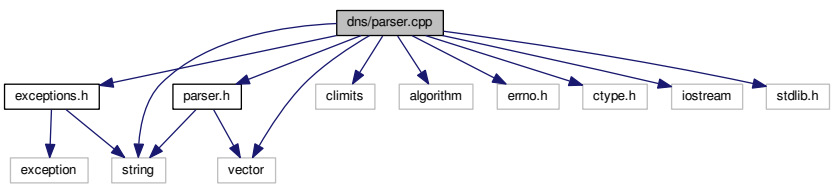
6.7.2 Variable Documentation

6.7.2.1 WorkerPool* MAIN = NULL

6.8 dns/parser.cpp File Reference

```
#include "parser.h"
#include "exceptions.h"
#include <climits>
#include <algorithm>
#include <vector>
#include <string>
#include <errno.h>
#include <ctype.h>
#include <iostream>
#include <stdlib.h>
```

Include dependency graph for parser.cpp:



Enumerations

- enum STR2INT_ERROR { S2I_SUCCESS, S2I_OVERFLOW, S2I_UNDERFLOW, S2I_INCONVERTIBLE }

Functions

- std::string pad_right (std::string const &str, size_t s)
- std::string pad_left (std::string const &str, size_t s)
- STR2INT_ERROR str2int (int &i, char const *s, int base=0)

Variables

- int errno

6.8.1 Enumeration Type Documentation

6.8.1.1 enum STR2INT_ERROR

Enumerator

S2I_SUCCESS

S2I_OVERFLOW

S2I_UNDERFLOW

S2I_INCONVERTIBLE

```
64 {
65     S2I_SUCCESS,
66     S2I_OVERFLOW,
67     S2I_UNDERFLOW,
68     S2I_INCONVERTIBLE
69 };
```

6.8.2 Function Documentation

6.8.2.1 std::string pad_left (std::string const & *str,* size_t *s*)

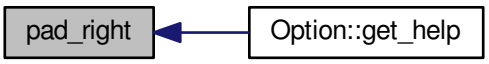
```
43 {
44     /* Adauga padding la stanga un string pana la o dimensiune anume
45      *
46      * @param[in] str
47      *   Stringul caruia vrem sa ii adaugam padding
48      *
49      * @param[in] s
50      *   Cat pading dorim sa ii adaugam
51      */
52     if ( str.length() < s )
53     {
54         return std::string(s-str.length(), ' ') + str;
55     }
56     else
57     {
58         return str;
59     }
60 }
```

6.8.2.2 std::string pad_right (std::string const & *str,* size_t *s*)

```
23 {
24     /* Adauga padding la dreapta un string pana la o dimensiune anume
25      *
26      * @param[in] str
27      *   Stringul caruia vrem sa ii adaugam padding
28      *
29      * @param[in] s
30      *   Cat pading dorim sa ii adaugam
31      */
32     if (str.length() < s)
33     {
34         return str + std::string(s-str.length(), ' ');
35     }
```

```
36     else
37     {
38         return str;
39     }
40 }
```

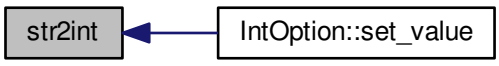
Here is the caller graph for this function:



6.8.2.3 STR2INT_ERROR str2int (int & *i*, char const * *s*, int *base* = 0)

```
72 {
73     char *end;
74     long  val;
75     errno = 0;
76     val = strtol(s, &end, base);
77     if ((errno == ERANGE && val == LONG_MAX) || val > INT_MAX)
78     {
79         return S2I_OVERFLOW;
80     }
81     if ((errno == ERANGE && val == LONG_MIN) || val < INT_MIN)
82     {
83         return S2I_UNDERFLOW;
84     }
85     if (*s == '\0' || *end != '\0')
86     {
87         return S2I_INCONVERTIBLE;
88     }
89     i = val;
90     return S2I_SUCCESS;
91 }
```

Here is the caller graph for this function:



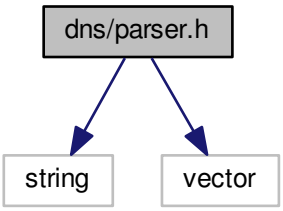
6.8.3 Variable Documentation

6.8.3.1 int errno

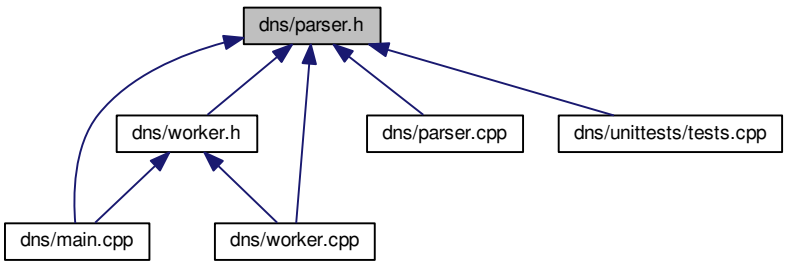
6.9 dns/parser.h File Reference

```
#include <string>
#include <vector>
```

Include dependency graph for parser.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Option
- class IntOption
- class StrOption
- class BoolOption
- class Parser

Macros

- #define PARTSE_H value

6.9.1 Macro Definition Documentation

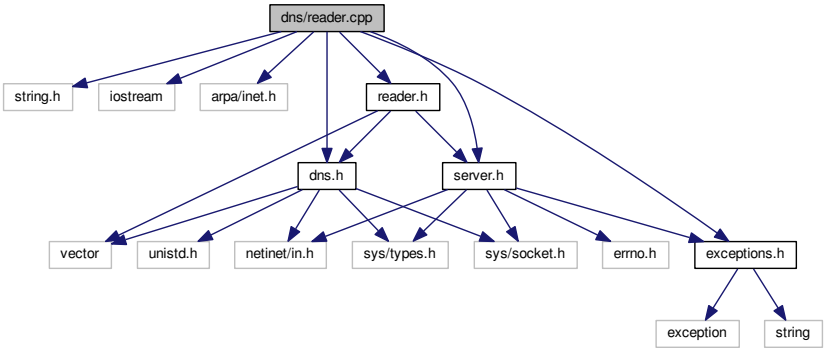
6.9.1.1 #define PARTSE_H value

6.10 dns/reader.cpp File Reference

```
#include <string.h>
```

```
#include <iostream>
#include <arpa/inet.h>
#include "reader.h"
#include "server.h"
#include "dns.h"
#include "exceptions.h"
```

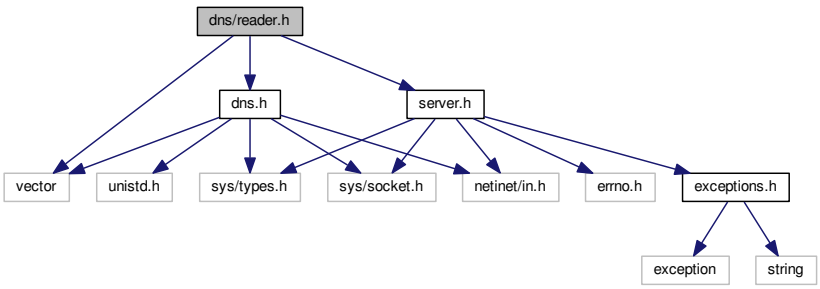
Include dependency graph for reader.cpp:



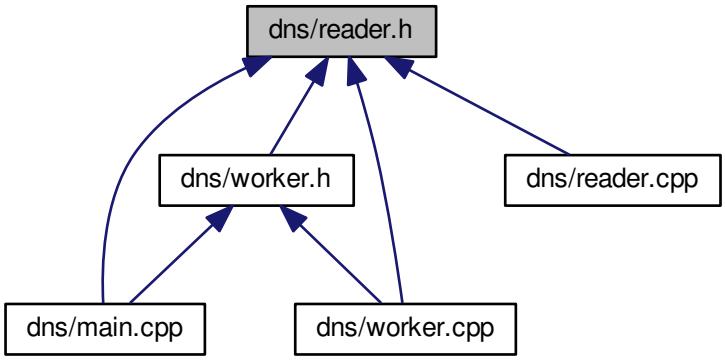
6.11 dns/reader.h File Reference

```
#include <vector>
#include "server.h"
#include "dns.h"
```

Include dependency graph for reader.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Reader

Macros

- #define READER_H_ value

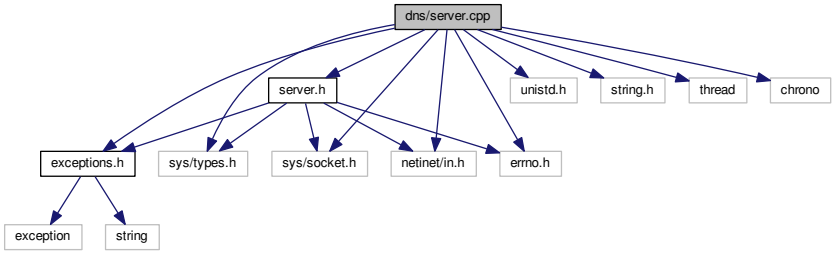
6.11.1 Macro Definition Documentation

6.11.1.1 #define READER_H_ value

6.12 dns/server.cpp File Reference

```
#include "server.h"
#include "exceptions.h"
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <thread>
#include <chrono>
```

Include dependency graph for server.cpp:



Variables

- int errno

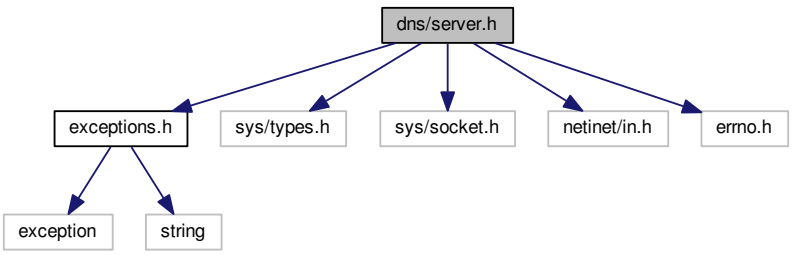
6.12.1 Variable Documentation

6.12.1.1 int errno

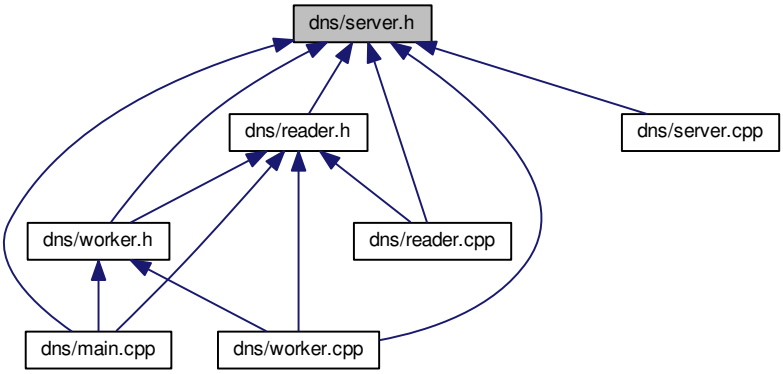
6.13 dns/server.h File Reference

```
#include "exceptions.h"
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <errno.h>
```

Include dependency graph for server.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Server

Macros

- #define SERVER_H value

6.13.1 Macro Definition Documentation

6.13.1.1 #define SERVER_H value

6.14 dns/unittests/tests.cpp File Reference

```
#include <string.h>
#include <iostream>
#include "catch.hpp"
#include "../parser.h"
#include "../exceptions.h"
#include "../dns.h"
```

Include dependency graph for tests.cpp:

Macros

- #define CATCH_CONFIG_MAIN

Functions

- TEST_CASE ("DNS-Question","[DNS-Question]")
- TEST_CASE ("DNS-Resource","[DNS-Resource]")
- TEST_CASE ("DNS-Tranzaction","[DNS-Tranzaction]")

6.14.1 Macro Definition Documentation

6.14.1.1 #define CATCH_CONFIG_MAIN

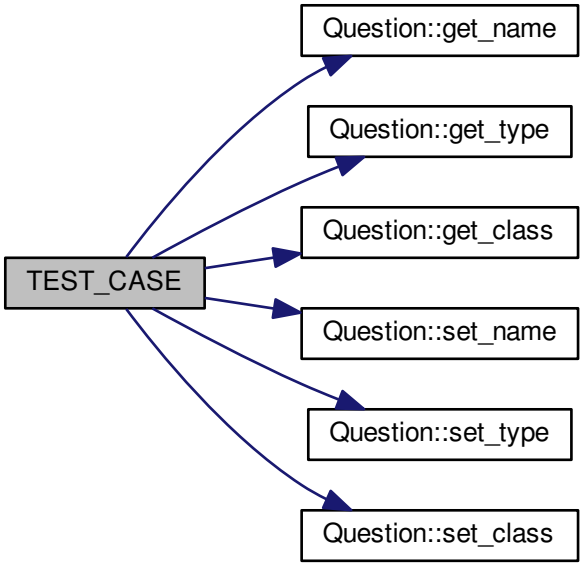
6.14.2 Function Documentation

6.14.2.1 TEST_CASE ("DNS-Question" , "" *[DNS-Question]*)

```
18 {
19     Question q;
20     unsigned short len = 2;
21     int cmp = -100;
22 
23     SECTION("Test default constructor - name")
24     {
25         char* name;
26         name = (char*)1; /* Trebuie sa fie diferit de NULL */
27         unsigned short length;
28 
29         q.get_name(&name, length);
30         REQUIRE(length == 0);
31         REQUIRE(name == NULL);
32     }
33 
34     SECTION("Test default constructor - type")
35     {
36         char type[len];
37         memset(type, 'f',  sizeof(type));
38         q.get_type(type);
39         cmp = strncmp(type, "\0\0", len);
40 
41         REQUIRE(cmp == 0);
42     }
43 
44     SECTION("Test default constructor - class")
45     {
46         char cls[len];
47         memset(cls, 'f',  sizeof(cls));
48         q.get_class(cls);
49         cmp = strncmp(cls, "\0\0", len);
50 
51         REQUIRE(cmp == 0);
52     }
53 
54     SECTION("Test setter - name")
55     {
56 
57         for (unsigned short i = 1; i <= 10; i++)
58         {
59             unsigned short aux_len = 0;
60             char name_set[i];
61             char* name_get = NULL;
62             memset(name_set, 'f',  i);
63 
64             q.set_name(name_set, i);
65             q.get_name(&name_get, aux_len);
66 
67             cmp = strncmp(name_set, name_get, aux_len);
68 
69             REQUIRE(i == aux_len);
70             REQUIRE(cmp == 0);
71         }
72     }
73 
74     SECTION("Test setter - type")
75     {
76         char type_set[len];
77         char type_get[len];
78         memset(type_set, 'f',  sizeof(type_set));
79         memset(type_set, 'x',  sizeof(type_get));
80         q.set_type(type_set);
81         q.get_type(type_get);
82 
```

```
83          cmp = strncmp(type_set, type_get, len);
84
85          REQUIRE(cmp == 0);
86      }
87
88      SECTION("Test setter - class")
89      {
90          char cls_set[len];
91          char cls_get[len];
92          memset(cls_set, 'f',  sizeof(cls_set));
93          memset(cls_set, 'x',  sizeof(cls_get));
94          q.set_class(cls_set);
95          q.get_class(cls_get);
96
97          cmp = strncmp(cls_set, cls_get, len);
98
99          REQUIRE(cmp == 0);
100     }
101
102 }
```

Here is the call graph for this function:



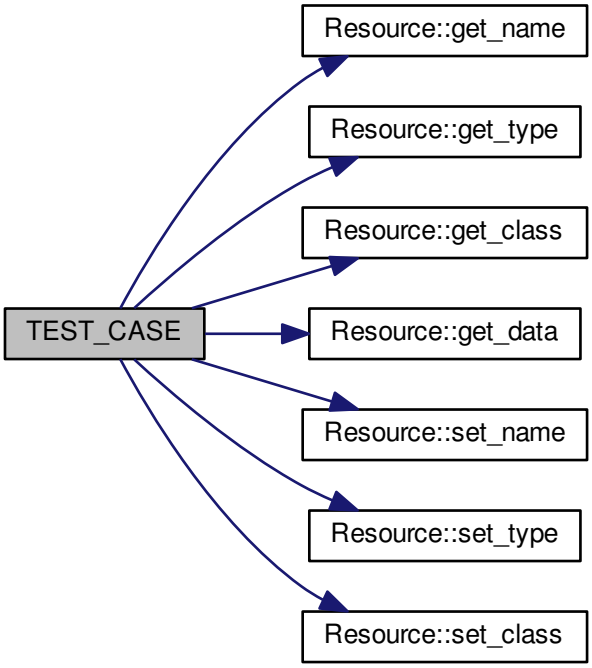
6.14.2.2 TEST_CASE ("DNS-Resource" , "" *[DNS-Resource]*)

```
105 {
106     Resource res;
107     unsigned short len = 2;
108     int cmp = -100;
109
110     SECTION("Test default constructor - name")
111     {
112         char* name;
113         name = (char*)1; /* Trebuie sa fie diferit de NULL */
114         unsigned short length;
115
116         res.get_name(&name, length);
117         REQUIRE(length == 0);
118         REQUIRE(name == NULL);
119     }
120
121     SECTION("Test default constructor - type")
```

```
122     {
123         char type[len];
124         memset(type, 'f',  sizeof(type));
125         res.get_type(type);
126         cmp = strncmp(type, "\0\0", len);
127 
128         REQUIRE(cmp == 0);
129     }
130 
131     SECTION("Test default constructor - class")
132     {
133         char cls[len];
134         memset(cls, 'f',  sizeof(cls));
135         res.get_class(cls);
136         cmp = strncmp(cls, "\0\0", len);
137 
138         REQUIRE(cmp == 0);
139     }
140 
141     SECTION("Test default constructor - data")
142     {
143         char *data = (char*)1;
144         unsigned short len = 1;
145 
146         res.get_data(&data, len);
147         REQUIRE(len == 0);
148         REQUIRE(data == NULL);
149     }
150 
151     SECTION("Test setter - name")
152     {
153 
154         for (unsigned short i = 1; i <= 10; i++)
155         {
156             unsigned short aux_len = 0;
157             char name_set[i];
158             char* name_get = NULL;
159             memset(name_set, 'f',  i);
160 
161             res.set_name(name_set, i);
162             res.get_name(&name_get, aux_len);
163 
164             cmp = strncmp(name_set, name_get, aux_len);
165 
166             REQUIRE(i == aux_len);
167             REQUIRE(cmp == 0);
168         }
169     }
170 
171     SECTION("Test setter - type")
172     {
173         char type_set[len];
174         char type_get[len];
175         memset(type_set, 'f',  sizeof(type_set));
176         memset(type_set, 'x',  sizeof(type_get));
177         res.set_type(type_set);
178         res.get_type(type_get);
179 
180         cmp = strncmp(type_set, type_get, len);
181 
182         REQUIRE(cmp == 0);
183     }
184 
185     SECTION("Test setter - class")
186     {
187         char cls_set[len];
188         char cls_get[len];
189         memset(cls_set, 'f',  sizeof(cls_set));
190         memset(cls_set, 'x',  sizeof(cls_get));
191         res.set_class(cls_set);
192         res.get_class(cls_get);
193 
194         cmp = strncmp(cls_set, cls_get, len);
195 
196         REQUIRE(cmp == 0);
197     }
198 
199     SECTION("Test setter - data")
200 
201     {
202 
203         for (unsigned short i = 1; i <= 10; i++)
204         {
205             unsigned short aux_len = 0;
206             char data_set[i];
207             char* data_get = NULL;
208             memset(data_set, 'f',  i);
```

```
209
210             res.set_name(data_set, i);
211             res.get_name(&data_get, aux_len);
212
213             cmp = strncmp(data_set, data_get, aux_len);
214
215             REQUIRE(i == aux_len);
216             REQUIRE(cmp == 0);
217         }
218     }
219
220 }
```

Here is the call graph for this function:



6.14.2.3 TEST_CASE ("DNS-Tranzaction" , "" *[DNS-Tranzaction]*)

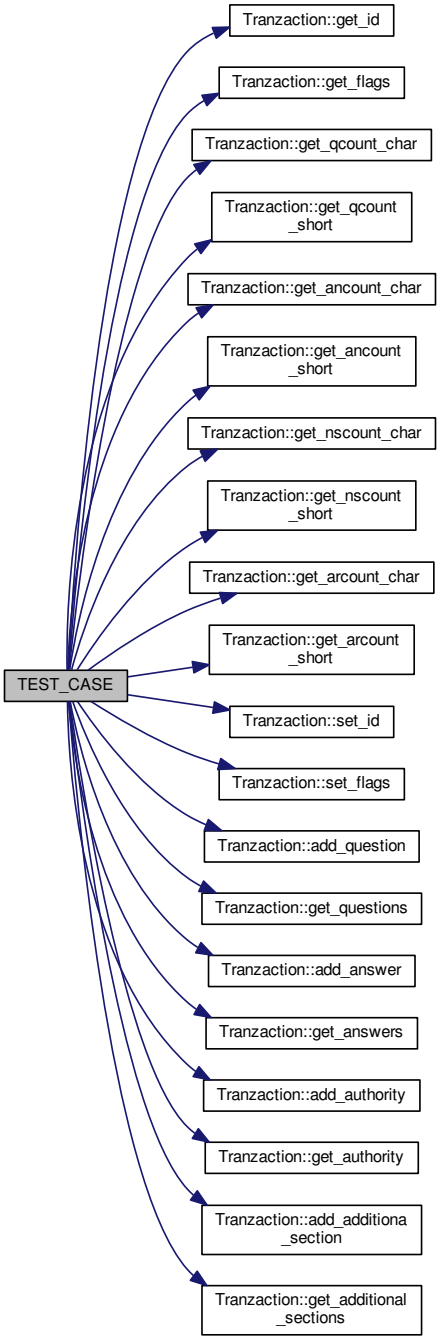
```
223 {
224     Tranzaction tr;
225     unsigned short len = 2;
226     int cmp = -100;
227
228     SECTION("Test default constructor - id")
229     {
230         char id[len];
231         memset(id, 'f',  sizeof(id));
232         tr.get_id(id);
233         cmp = strncmp(id, "\0\0", len);
234
235         REQUIRE(cmp == 0);
236     }
237
238     SECTION("Test default constructor - flags")
239     {
240         char flags[len];
241         memset(flags, 'f',  sizeof(flags));
242         tr.get_flags(flags);
243         cmp = strncmp(flags, "\0\0", len);
```

```
244
245         REQUIRE(cmp == 0);
246     }
247
248     SECTION("Test default constructor - qcount")
249     {
250         char qcount[len];
251         memset(qcount, 'f', len);
252
253         unsigned short rez;
254         tr.get_qcount_char(qcount);
255         rez = tr.get_qcount_short();
256
257         cmp = strncmp(qcount, "\0\0", len);
258         REQUIRE(cmp == 0);
259         REQUIRE(rez == 0);
260     }
261
262     SECTION("Test default constructor - ancount")
263     {
264         char ancount[len];
265         memset(ancount, 'f', len);
266
267         unsigned short rez;
268         tr.get_ancount_char(ancount);
269         rez = tr.get_ancount_short();
270
271         cmp = strncmp(ancount, "\0\0", len);
272         REQUIRE(cmp == 0);
273         REQUIRE(rez == 0);
274     }
275
276     SECTION("Test default constructor - nscount")
277     {
278         char nscount[len];
279         memset(nscount, 'f', len);
280
281         unsigned short rez;
282         tr.get_nscount_char(nscount);
283         rez = tr.get_nscount_short();
284
285         cmp = strncmp(nscount, "\0\0", len);
286         REQUIRE(cmp == 0);
287         REQUIRE(rez == 0);
288     }
289
290     SECTION("Test default constructor - arcount")
291     {
292         char arcount[len];
293         memset(arcount, 'f', len);
294
295         unsigned short rez;
296         tr.get_arcount_char(arcount);
297         rez = tr.get_arcount_short();
298
299         cmp = strncmp(arcount, "\0\0", len);
300         REQUIRE(cmp == 0);
301         REQUIRE(rez == 0);
302     }
303
304     SECTION("Test setter - id")
305     {
306         char id_set[2],
307              id_get[2];
308
309         for (char i ='a'; i <= 'c'; i++)
310         {
311             id_set[0] = i;
312             id_set[1] = i+1;
313
314             tr.set_id(id_set);
315             tr.get_id(id_get);
316
317             cmp = strncmp(id_set, id_get, 2);
318             REQUIRE(cmp == 0);
319         }
320     }
321
322     SECTION("Test setter - flags")
323     {
324         char flags_set[2],
325              flags_get[2];
326
327         for (char i ='a'; i <= 'c'; i++)
328         {
329             flags_set[0] = i;
330             flags_set[1] = i+1;
```

```
331
332             tr.set_flags(flags_set);
333             tr.get_flags(flags_get);
334             cmp = strncmp(flags_set, flags_get, 2);
335             REQUIRE(cmp == 0);
336         }
337     }
338
339
340     SECTION("Test setter - questions")
341     {
342         Question q1, q2;
343
344         REQUIRE(tr.get_qcount_short() == 0);
345
346         /* One question */
347         tr.add_question(q1);
348         std::vector<Question> questions;
349         questions = tr.get_questions();
350
351         REQUIRE(tr.get_qcount_short() == 1);
352         REQUIRE(questions.size() == 1);
353
354         /* Two question */
355         tr.add_question(q2);
356         questions = tr.get_questions();
357
358         REQUIRE(tr.get_qcount_short() == 2);
359         REQUIRE(questions.size() == 2);
360
361     }
362
363     SECTION("Test setter - answers")
364     {
365         Resource r1, r2;
366
367         REQUIRE(tr.get_ancount_short() == 0);
368
369         /* One resources */
370         tr.add_answer(r1);
371         std::vector<Resource> resource;
372         resource = tr.get_answers();
373
374         REQUIRE(tr.get_ancount_short() == 1);
375         REQUIRE(resource.size() == 1);
376
377         /* Two resouces */
378         tr.add_answer(r2);
379         resource = tr.get_answers();
380
381         REQUIRE(tr.get_ancount_short() == 2);
382         REQUIRE(resource.size() == 2);
383     }
384
385     SECTION("Test setter - authority")
386     {
387         Resource r1, r2;
388
389         REQUIRE(tr.get_nscount_short() == 0);
390
391         /* One resources */
392         tr.add_authority(r1);
393         std::vector<Resource> resource;
394         resource = tr.get_authority();
395
396         REQUIRE(tr.get_nscount_short() == 1);
397         REQUIRE(resource.size() == 1);
398
399         /* Two resouces */
400         tr.add_authority(r2);
401         resource = tr.get_authority();
402
403         REQUIRE(tr.get_nscount_short() == 2);
404         REQUIRE(resource.size() == 2);
405     }
406
407     SECTION("Test setter - additional_sections")
408     {
409         Resource r1, r2;
410
411         REQUIRE(tr.get_arcount_short() == 0);
412
413         /* One resources */
414         tr.add_additiona_section(r1);
415         std::vector<Resource> resource;
416         resource = tr.get_additional_sections();
417
```

```
418         REQUIRE(tr.get_arcount_short() == 1);
419         REQUIRE(resource.size() == 1);
420
421         /* Two resouces */
422         tr.add_additiona_section(r2);
423         resource = tr.get_additional_sections();
424
425         REQUIRE(tr.get_arcount_short() == 2);
426         REQUIRE(resource.size() == 2);
427     }
428
429 }
```

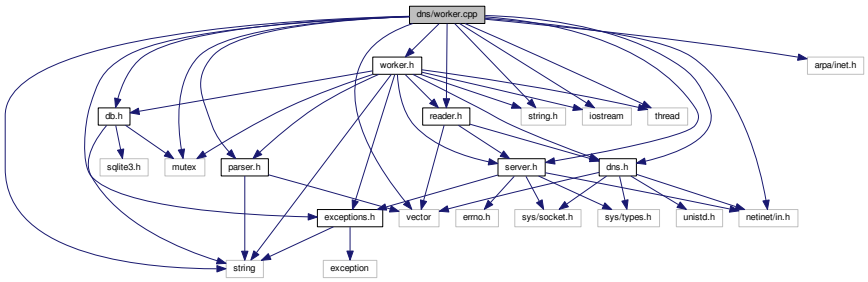
Here is the call graph for this function:



6.15 dns/worker.cpp File Reference

```
#include <string>
#include <string.h>
#include <iostream>
#include <mutex>
#include <vector>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <thread>
#include "server.h"
#include "db.h"
#include "dns.h"
#include "exceptions.h"
#include "parser.h"
#include "reader.h"
#include "worker.h"
```

Include dependency graph for worker.cpp:



Functions

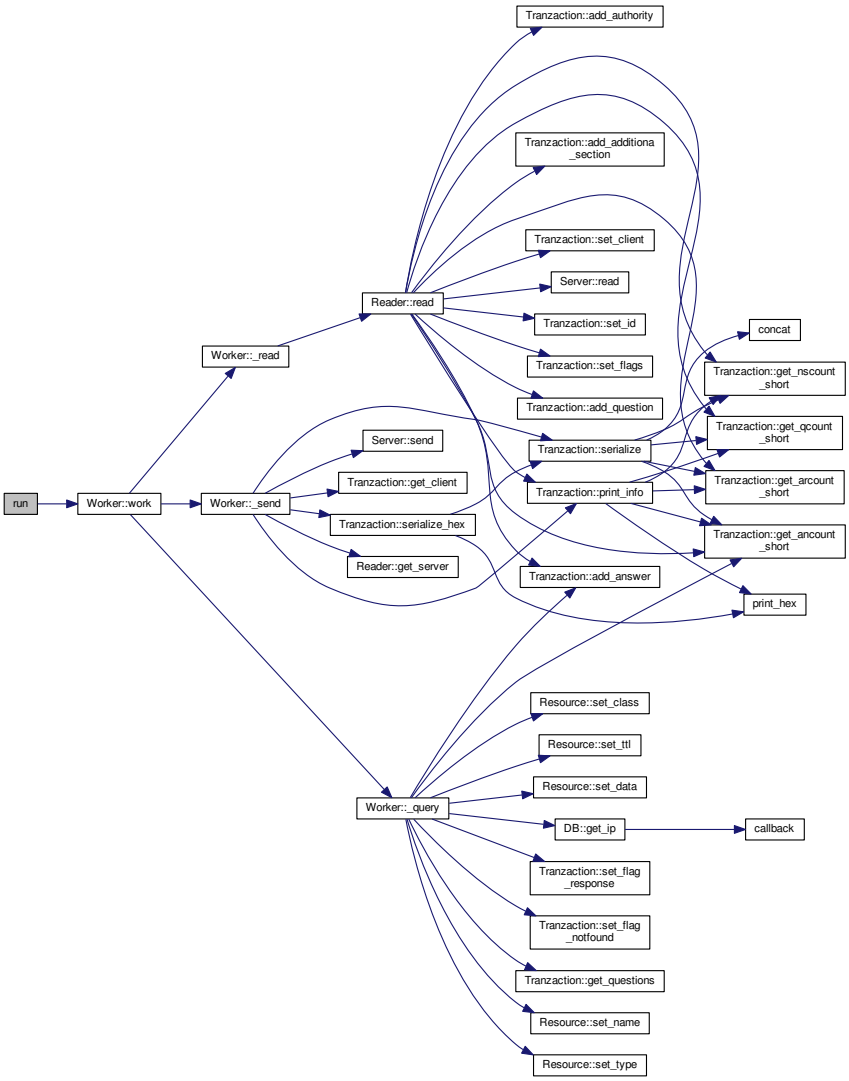
- void run (Worker *w)

6.15.1 Function Documentation

6.15.1.1 void run (Worker * w)

```
23 {
24     /* Pornim workerul
25      *
26      * @param w
27      *  Workerul pe care dorim sa il pornim
28      */
29     std::cout << "Apelam work !" << std::endl;
30     w->work();
31 }
```

Here is the call graph for this function:



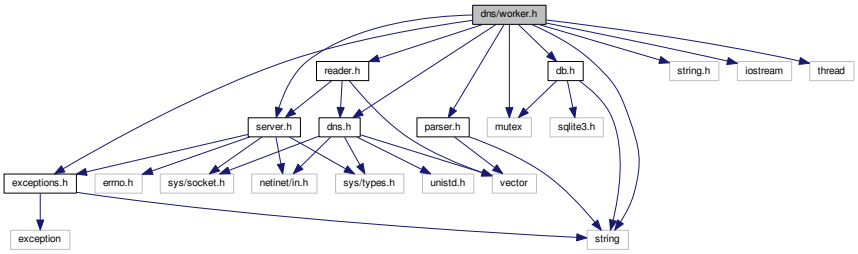
Here is the caller graph for this function:

6.16 dns/worker.h File Reference

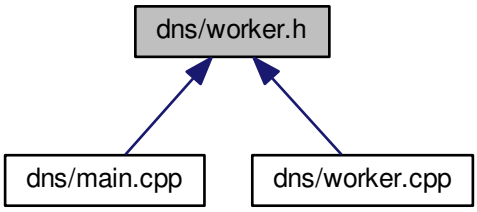
```
#include <string>
```

```
#include <string.h>
#include <iostream>
#include <mutex>
#include <thread>
#include "server.h"
#include "db.h"
#include "dns.h"
#include "parser.h"
#include "reader.h"
#include "exceptions.h"
```

Include dependency graph for worker.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Worker
- class WorkerPool

Macros

- #define WORKER_H value

6.16.1 Macro Definition Documentation

6.16.1.1 #define WORKER_H value

6.17 README.md File Reference