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:

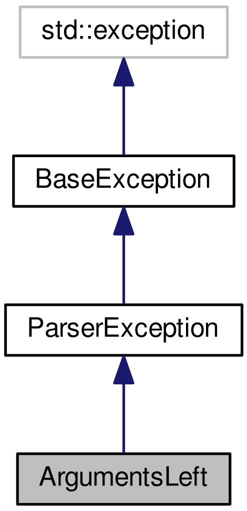
Chapter 5

Class Documentation

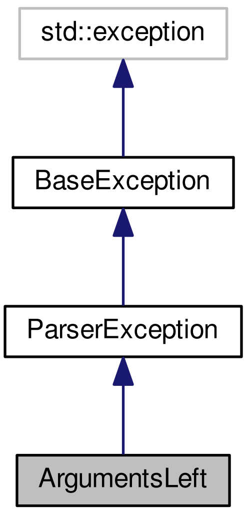
5.1 ArgumentsLeft Class Reference

```
#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)

```
{
    return "Au ramas argumente neparsate";
}
```

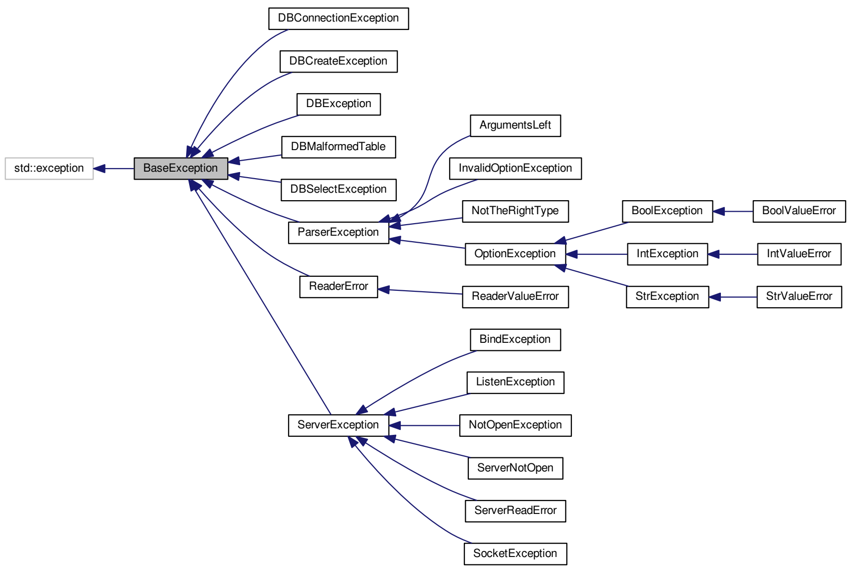
The documentation for this class was generated from the following files:

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

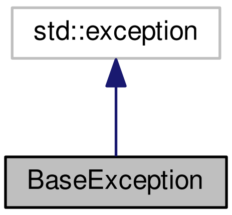
5.2 BaseException Class Reference

`#include <exceptions.h>`

Inheritance diagram for BaseException:



Collaboration diagram for BaseException:



Public Member Functions

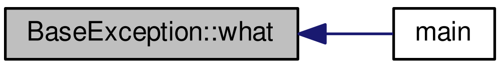
- const char ∗ what () const throw ()

5.2.1 Member Function Documentation

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

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

Here is the caller graph for this function:



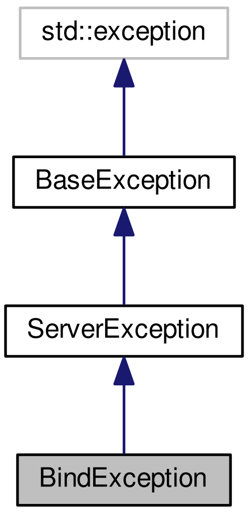
The documentation for this class was generated from the following files:

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

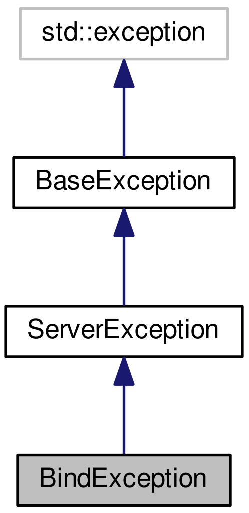
5.3 BindException Class Reference

```
#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)

```
{
    return "Eroare la bind !";
}
```

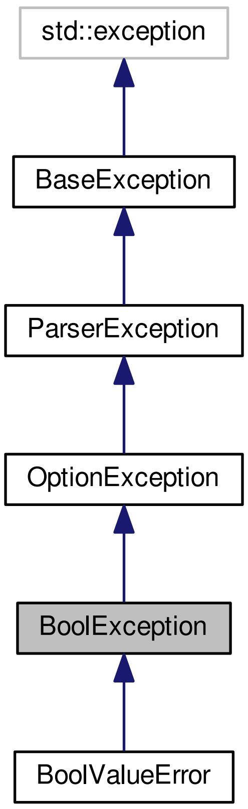
The documentation for this class was generated from the following files:

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

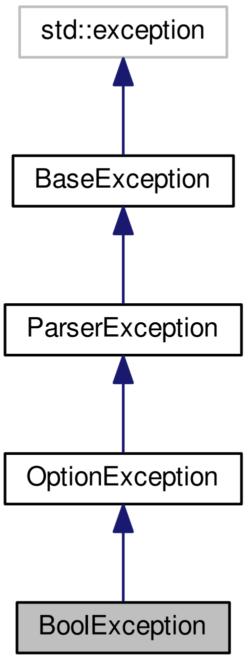
5.4 BoolException Class Reference

`#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*)

```
                                              : OptionException(
      primit)
{
}
```

5.4.2 Member Function Documentation

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

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

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

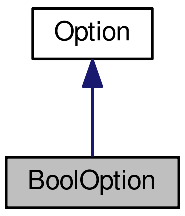
- dns/exceptions.h

- dns/exceptions.cpp

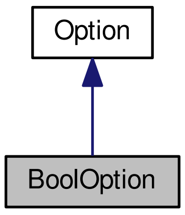
5.5 BoolOption Class Reference

`#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*)

```
                                                              :
   Option(short_name, long_name, help_message,
 required)
{
   /* Constructor pentru BoolOption
    *
    * @param[in] short_name
    *  Un char care reprezinta varianta prescurtata a parametrului
    *  Ex: -d
    *
    * @param[in] long_name
    *  Un string care reprezinta varianta lunga aparametrului
    *  Ex: --delimitator
    *
    * @param[in] help_string
    *  Un string care reprezinta descrierea optiuni
    *
    * @param[in] required
    *  Daca un parametru este necesar sau nu, implicit este setat pe false
    */
}
```

5.5.2 Member Function Documentation

5.5.2.1 bool BoolOption::get_bool ( ) [virtual]

Reimplemented from Option.

```
{
   /* \return{Returneaza valoarea parametrului} */
   return this->value;
}
```

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

Reimplemented from Option.

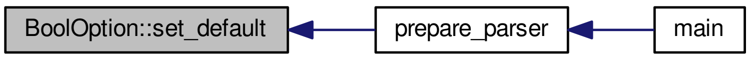
```
{
   /* Returneaza un string care reprezinta tipul optinui */
   return std::string("bool");
}
```

5.5.2.3 bool BoolOption::is_required ( )

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

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

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.

```
{
    /* Seteaza valoarea unui parametru convorm unui string
     *
     * @param parameter
     *  Stringul care contine valoarea parametrului
     */
    std::string lower_param = parameter;
    /* Convert to lower case */
    std::transform(lower_param.begin(), lower_param.end(),
                   lower_param.begin(), ::tolower);

    if (lower_param == "true" || lower_param == "t")
    {
        this->value = true;
        this->is_set = true;
        return;
    }

    if (lower_param == "false" || lower_param == "f")
    {
        this->value = false;
        this->is_set = true;
        return;
    }

    throw BoolValueError(parameter);
}
```

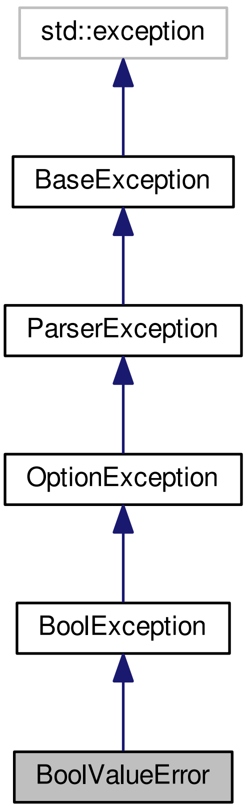
The documentation for this class was generated from the following files:

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

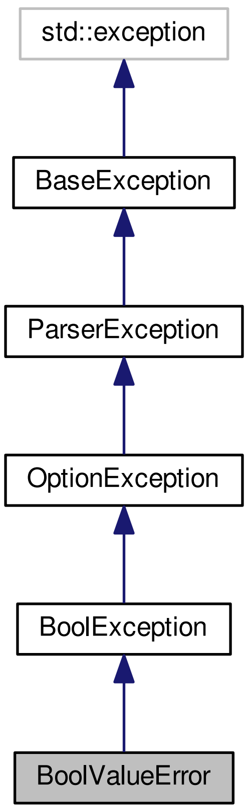
5.6 BoolValueError Class Reference

`#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*)

```
                                                : BoolException(
    primit)
{
}
```

5.6.2 Member Function Documentation

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

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

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*)

```
{
    /* Initializam conexiunea
     *
     *
     * In cazul in care fisierul nu exista sau nu respecta
     * structura o vom genera pe loc si o vom popula cu 2 ip-uri
     * google.com -> 172.217.22.14
     * example.com -> 1.1.1.1
     *
     * @param[in] filename
     *  Un string cu numele fisierului
     */

    this->filename = new char[strlen(filename)];
    bzero(this->filename, strlen(filename));
    strcpy(this->filename, filename);

    this->_ip = new char[this->IP_MAX_SIZE];
    bzero(this->_ip, this->IP_MAX_SIZE);

    this->errMsg = NULL;

    int rc = sqlite3_open(this->filename, &this->db);

    if (rc != 0)
    {
        throw DBConnectionException();
    }
    if (this->_is_prepare() == false)
    {
        this->_prepare();
    }
}
```

5.7.1.2 DB::∼DB ( )

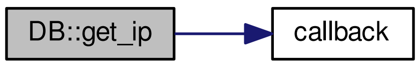
```
{
    /* Dealocam memoria */
    sqlite3_close(this->db);
    delete this->filename;
    delete this->_ip;
}
```

5.7.2 Member Function Documentation

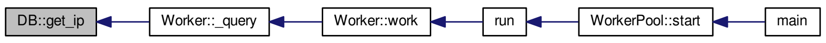
5.7.2.1 std::string DB::get_ip (char ∗ *name,* unsigned short *name_len*)

```
{
    /* Returneaza ip-ul pentru domeniul dat ca parametru.
     * In cazul in care nu am gasit nici o inregistrare returnam
     * stringul gol.
     *
     * @param[in] name
     *  Numele domeniului
     *
     * @param[in] name_len
     *  Dimensiunea numelui
     *
     */
    this->lock.lock();

    std::string real_domail;
    unsigned char lungime = 0;
    for (unsigned char i = 0; i < name_len-1; ++i)
    {
        lungime = name[i];
        real_domail = real_domail + std::string(".");
        for (unsigned char j = i+1 ; j <= i+lungime; ++j)
        {
            real_domail = real_domail + std::string(1, name[j]);
        }
        i = i + lungime;
    }
    name = (char*)real_domail.c_str();
    std::cout << " ------  Search for " << name << std::endl;

    memset(this->_ip, 0, this->IP_MAX_SIZE);

    /* Convertim in string */
    char c_name[name_len+1];
    memset(c_name, 0, name_len+1);
    memcpy(c_name, name, name_len);
    std::string s_name(c_name), ip("");

    /* NOTE(mmicu): SQL injection DROP TABLE ;) */
    std::string sql = "SELECT " + std::string(DOMAIN)+", " +std::string(IP) +
                      " FROM " + std::string(TABLE_NAME) +
                      " WHERE " + std::string(DOMAIN) +" = '"+ s_name +"';";

    int res = sqlite3_exec(this->db, sql.c_str(), callback, (void*)this->_ip, &this->errMsg);


    if (res != 0)
    {
        if (this->errMsg != NULL)
        {
            /* Eroare de la sqlite */
            std::cerr << this->errMsg << std::endl;
            sqlite3_free(this->errMsg);
        }

    }
    else
    {
        ip = std::string(this->_ip);
    }

    this->lock.unlock();
    return ip;
}
```

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]`

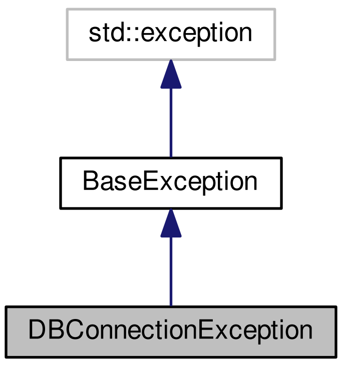
The documentation for this class was generated from the following files:

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

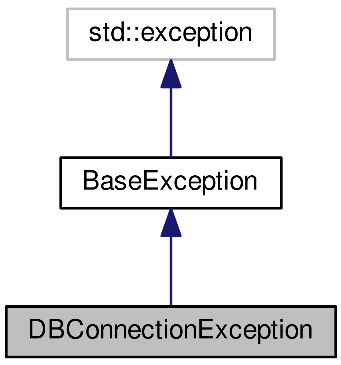
5.8 DBConnectionException Class Reference

`#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)

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

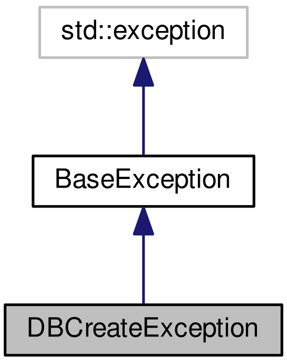
The documentation for this class was generated from the following files:

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

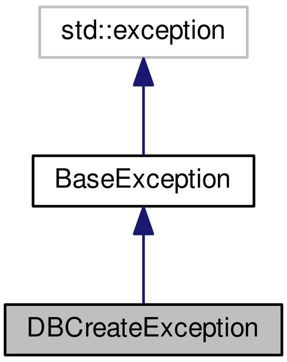
5.9 DBCreateException Class Reference

```
#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)

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

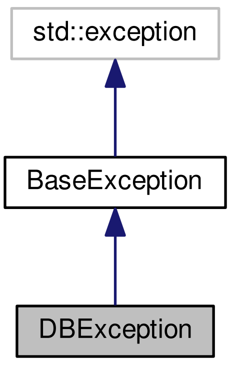
The documentation for this class was generated from the following files:

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

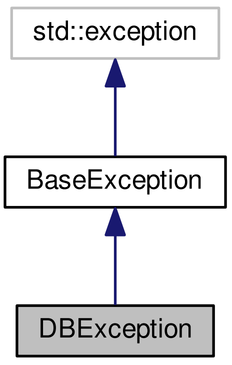
5.10 DBException Class Reference

```
#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)

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

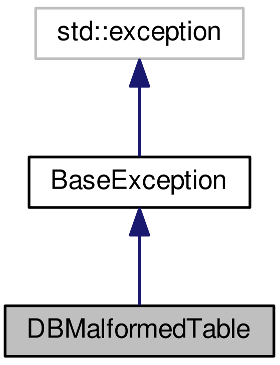
The documentation for this class was generated from the following files:

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

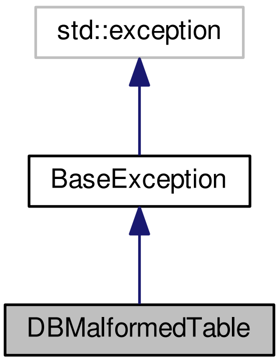
5.11 DBMalformedTable Class Reference

`#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)

```
{
    return "Tabela este malformata!";
}
```

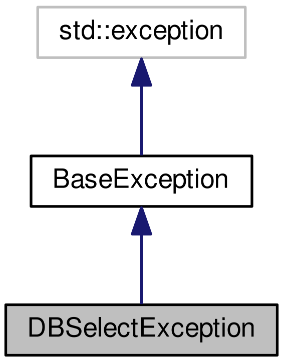
The documentation for this class was generated from the following files:

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

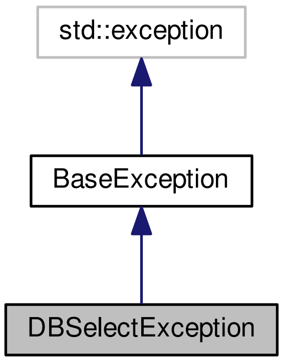
5.12 DBSelectException Class Reference

```
#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)

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

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

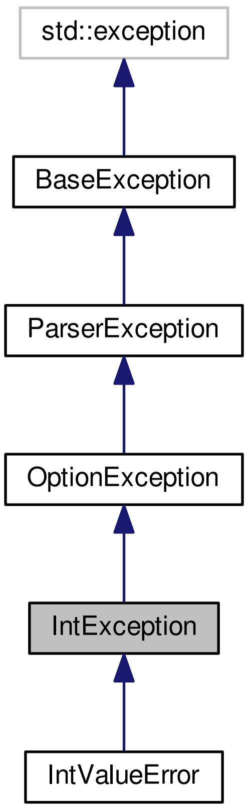
- dns/exceptions.h

- dns/exceptions.cpp

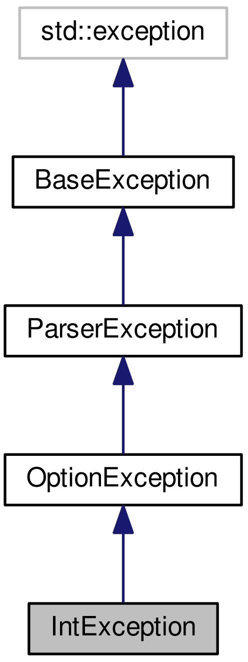
5.13 IntException Class Reference

```
#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*)

```
                                            : OptionException(
      primit)
{
}
```

5.13.2 Member Function Documentation

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

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

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

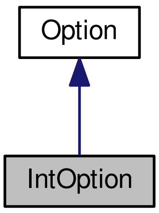
- dns/exceptions.h

- dns/exceptions.cpp

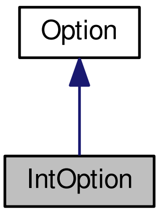
5.14 IntOption Class Reference

```
#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*)

```
                                                          :
  Option(short_name, long_name, help_message,
required)
{
  /* Constructor pentru IntOption
   *
   * @param[in] short_name
   *  Un char care reprezinta varianta prescurtata a parametrului
   *  Ex: -d
   *
   * @param[in] long_name
   *  Un string care reprezinta varianta lunga aparametrului
   *  Ex: --delimitator
   *
   * @param[in] help_string
   *  Un string care reprezinta descrierea optiuni
   *
   * @param[in] required
   *  Daca un parametru este necesar sau nu, implicit este setat pe false
   */
}
```

5.14.2 Member Function Documentation

5.14.2.1 int IntOption::get_int ( ) [virtual]

Reimplemented from Option.

```
{
  /* \return{Returneaza valoarea parametrului} */
  return this->value;
}
```

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

Reimplemented from Option.

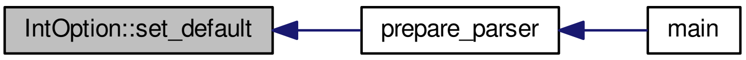
```
{
  /* Returneaza un string care reprezinta tipul optinui */
  return std::string("int");
}
```

5.14.2.3 bool IntOption::is_required ( )

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

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

Here is the caller graph for this function:

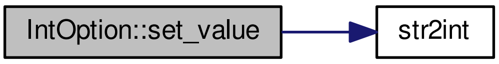


5.14.2.5 void IntOption::set_value (std::string *param_value*) [protected], [virtual]

Reimplemented from Option.

```
{
    /* Seteaza valoarea unui parametru convorm unui string
     *
     * @param parameter
     *  Stringul care contine valoarea parametrului
     */
    STR2INT_ERROR out = str2int(this->value, parameter.c_str(), 10);

    if (out != S2I_SUCCESS)
    {
        throw IntValueError(parameter);
    }
    this->is_set = true;
}
```

Here is the call graph for this function:



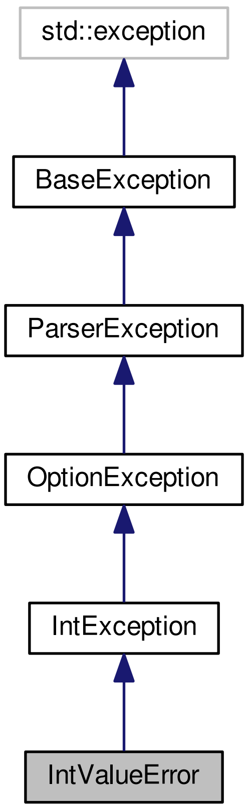
The documentation for this class was generated from the following files:

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

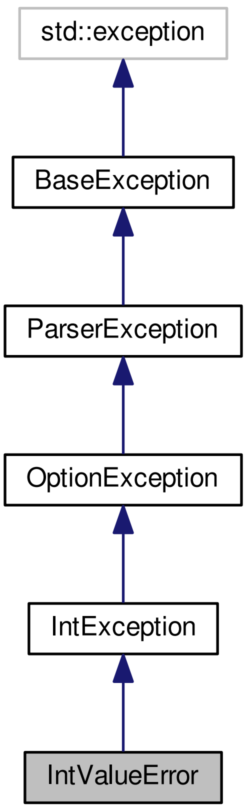
5.15 IntValueError Class Reference

`#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*)

```
                                                : IntException(primit)
{
}
```

5.15.2 Member Function Documentation

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

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

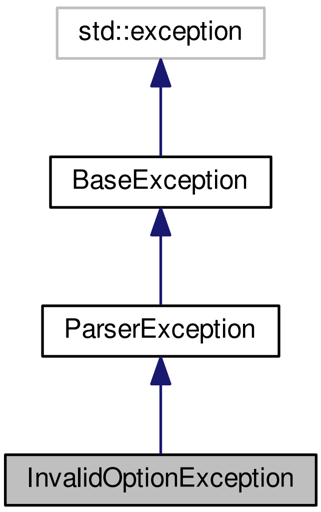
The documentation for this class was generated from the following files:

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

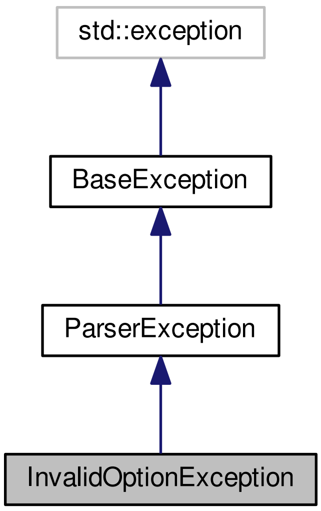
5.16 InvalidOptionException Class Reference

```
#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)

```
{
    return "Nu putem adauga optiunea asta";
}
```

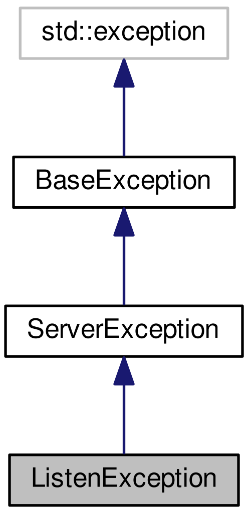
The documentation for this class was generated from the following files:

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

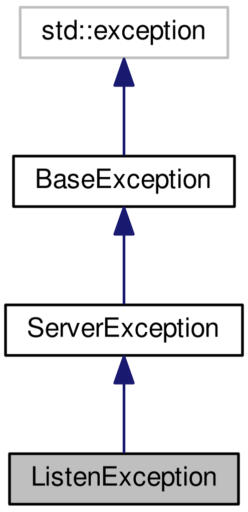
5.17 ListenException Class Reference

```
#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)

```
{
    return "Eroare la Listen";
}
```

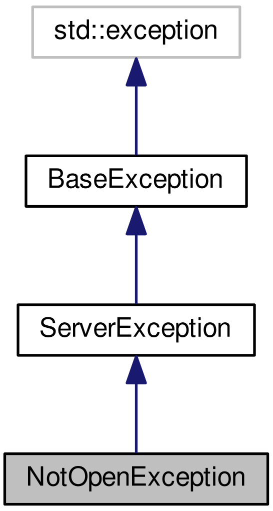
The documentation for this class was generated from the following files:

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

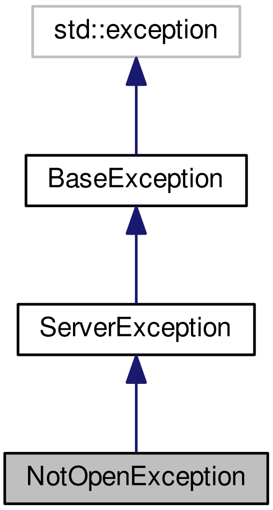
5.18 NotOpenException Class Reference

`#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)

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

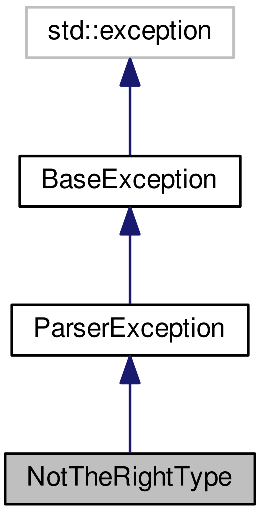
The documentation for this class was generated from the following files:

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

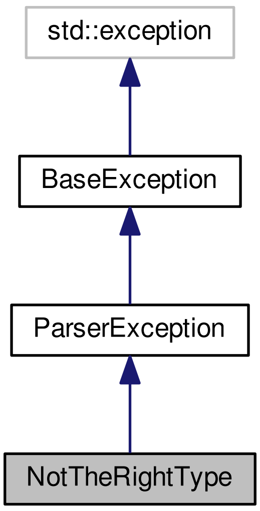
5.19 NotTheRightType Class Reference

`#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)

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

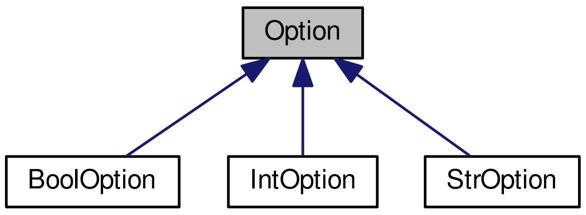
The documentation for this class was generated from the following files:

- 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*)

```
{
    /* Constructor pentru Clasa de baza
     *
     * @param[in] short_name
     *  Un char care reprezinta varianta prescurtata a parametrului
      *  Ex: -d
      *
      * @param[in] long_name
      *  Un string care reprezinta varianta lunga aparametrului
      *  Ex: --delimitator
      *
      * @param[in] help_string
      *  Un string care reprezinta descrierea optiuni
      *
      * @param[in] required
      *  Daca un parametru este necesar sau nu
      */
     this->short_name = short_name;
     this->long_name = long_name;
     this->help_message = help_message;
     this->default_set = false; /* o valoare default este setata dupa crearea obiectului */
     this->required=required;
}
```

5.20.2 Member Function Documentation

5.20.2.1 bool Option::get_bool ( ) [virtual]

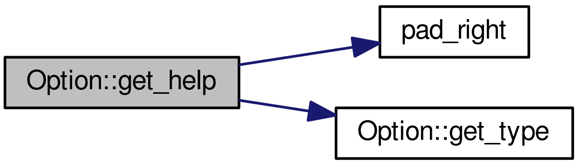
Reimplemented in BoolOption.

```
{
    throw NotTheRightType();
}
```

5.20.2.2 std::string Option::get_help ( )

```
{
    /* Returnam mesajul de ajutor complet */

    // TODO(mmicu):
    // Adauga un format de genu:
    // (2-4 spaces)short, long(4 - 8 spaces)help_message
    // poate si un word wrap de ~ 80 caractere

    char ch[2];
    ch[0] = this->short_name;
    ch[1] = 0;
    std::string to_ret = "  -" + std::string(ch) + ", --"+pad_right(this->
  long_name, 12) +
                         "   "+pad_right("("+this->get_type() +")", 6) + " :" +
                         this->help_message + "\n";
    return to_ret;
}
```

Here is the call graph for this function:



5.20.2.3 int Option::get_int ( ) [virtual]

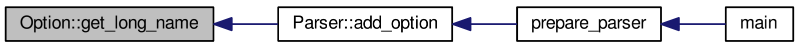
Reimplemented in IntOption.

```
{
    throw NotTheRightType();
}
```

5.20.2.4 std::string Option::get_long_name ( )

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

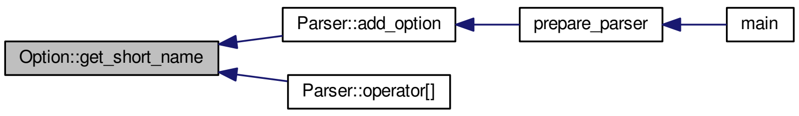
Here is the caller graph for this function:



5.20.2.5 char Option::get_short_name ( )

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

Here is the caller graph for this function:



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

Reimplemented in StrOption.

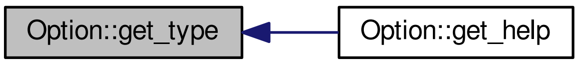
```
{
    throw NotTheRightType();
}
```

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

Reimplemented in BoolOption, StrOption, and IntOption.

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

Here is the caller graph for this function:



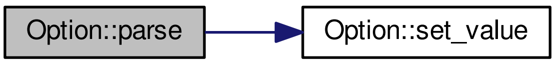
5.20.2.8 bool Option::is_required ( )

```
{
    /* Returneaza true daca optiunea este obligatori, fals
     * in caz contrar
     */
    return this->required;
}
```

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

```
{
    /* Parseaza lista cu parametri si verifica daca parametrul cautat
     * se afla in lista.
     *
     * @param[in] parameters
     *  Un vector cu stringuri, fiecare string este un parametru de la CLI.
     */
    // for (std::vector<std::string> it = parameters.begin(); it != parameters.end();)
    for (auto it = parameters.begin(); it != parameters.end();)
    {
        char sh[3];
        sh[0] = '-';
        sh[1] = this->short_name;
        sh[2] = 0;
        if ((std::string)(*it) == std::string(sh) ||
           (std::string)(*it) == "--"+this->long_name)
        {
            parameters.erase(it);
            this->set_value((std::string)(*it));
            parameters.erase(it);
        }
        else
        {
            /* Avanseaza doar daca nu ai eliminat ceva */
            it++;
        }

    }

    return parameters;
}
```

Here is the call graph for this function:

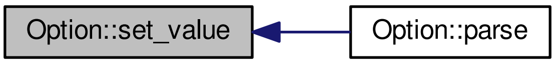


5.20.2.10 void Option::set_value (std::string *param_value*) [protected], [virtual]

Reimplemented in BoolOption, StrOption, and IntOption.

```
{
    /* Seteaza valoarea parametrului
     *
     * @param param_value
     *  Valoarea parametrului
     */
    throw ParserException();
}
```

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:

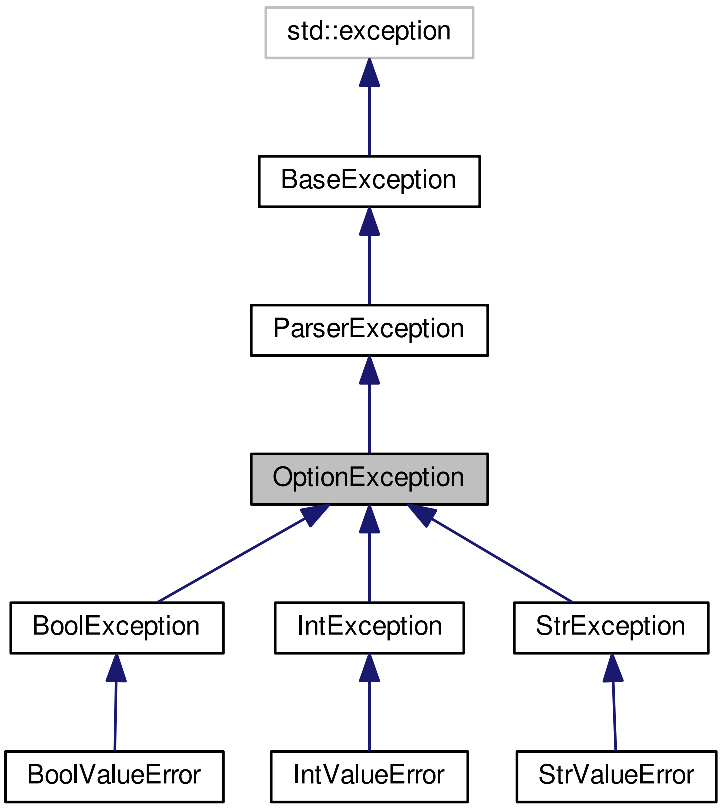
- dns/parser.h

- dns/parser.cpp

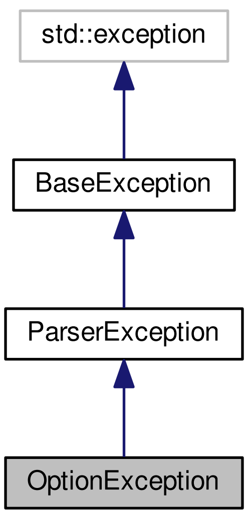
5.21 OptionException Class Reference

`#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*)

```
{
    /* Contructor pentru OptionException.
        *
        * @param[in] primit
        *  Un string care reprezinta parametrul primit.
        *  Implicit este stringul gol.
        */
    this->primit = primit;
}
```

5.21.1.2 OptionException::∼OptionException ( ) throw)

```
{}
```

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 ( )

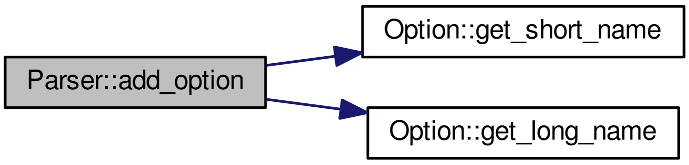
```
{
    /* Initializeaza un parser */
    this->options.clear();
    this->args.clear();
}
```

5.22.2 Member Function Documentation

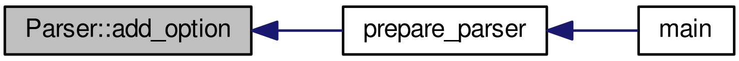
5.22.2.1 void Parser::add_option (Option ∗ *opt*)

```
{
    /* Adauga o noua optiune parserului
     *
     * @param opt
     *  O instanta a unei optiuni
     */
    for (std::vector<Option*>::iterator it = this->options.begin();
        it != this->options.end(); it++)
    {
        if ((*it)->get_short_name() == opt->get_short_name() ||
            (*it)->get_long_name() == opt->get_long_name())
        {
            throw InvalidOptionException();
        }
    }
    this->options.push_back(opt);
}
```

Here is the call graph for this function:



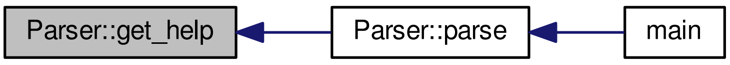
Here is the caller graph for this function:



5.22.2.2 void Parser::get_help ( )

```
{
    /* Print the help for each option */
    std::cout << "Usage :" << std::endl << "  " << this->exe_name << std::endl;


    /* Argumente obligatorii */
    for (std::vector<Option*>::iterator it = this->options.begin();
        it != this->options.end(); it++)
    {
        if ((*it)->is_required())
        {
            std::cout << (*it)->get_help() << std::endl;
        }
    }

    std::cout<< "Options :" << std::endl;
    for (std::vector<Option*>::iterator it = this->options.begin();
        it != this->options.end(); it++)
    {
        if ((*it)->is_required() == false)
        {
            std::cout << (*it)->get_help() << std::endl;
        }
    }
}
```

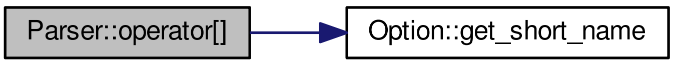
Here is the caller graph for this function:



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

```
{
    /* Returneaza optiunea cu acel nume
     *
     * @param name
     *  Numele optiuni, scurt sau lung
     */

    for (std::vector<Option*>::iterator it = this->options.begin();
        it != this->options.end(); it++)
    {
        char sh[2];
        sh[0] = (*it)->get_short_name();
        sh[0] = 0;
        if ((*it)->get_long_name() == name ||
            std::string(sh) == name)
        {
            return *it;
        }
    }

    throw InvalidOptionException();
}
```

Here is the call graph for this function:

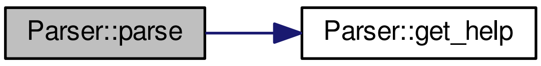


5.22.2.4 void Parser::parse (int *argc,* char ∗ *argv[]*)

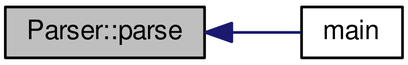
```
{
    /* Parse the arguments
     *
     * @param argc
     *  Numarul de argumente primite
     *
     * @param argv
     *  Un vector cu char* care reprezinta argumentele
     */

    /* Pastram numele executabilului */
    this->exe_name = std::string(argv[0]);

    /* Transformam char* in vector de std::string */

    /* NOTE(mmicu): Pornim de la 1 ca sa excludem
```

```
     * numele binarului
     */
    for (int i = 1; i < argc; i++)
    {
        this->args.push_back(std::string(argv[i]));
    }

    /* Verific daca s-a cecur help-ul */
    for (auto it = this->args.begin(); it != this->args.end(); it++)
    {
        if ((*it) == "-h" || (*it) == "--help" )
        {
            this->get_help();
            exit(0);
        }
    }

    /* Daca nu avem nici un argument, afisam help-ul */
    if (this->args.size() == 0)
    {
        /* Daca afisam mesajul de ajutor, nu mai parsam restul argumentelor.
         * Terminam totui executia programului. */
        this->get_help();
        exit(0);
    }

    for (std::vector<Option*>::iterator it = this->options.begin();
        it != this->options.end(); it++)
    {
        this->args = (*it)->parse(this->args);
    }

    if (this->args.size() > 0 )
    {
        throw ArgumentsLeft();
    }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



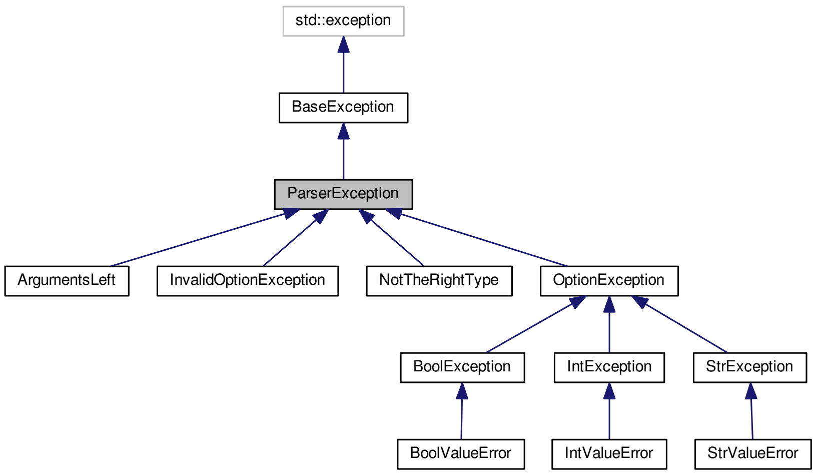
The documentation for this class was generated from the following files:

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

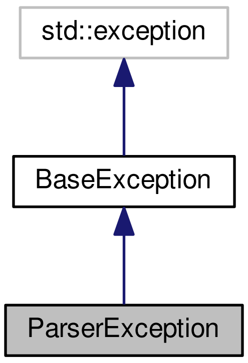
5.23 ParserException Class Reference

```
#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 {
    return "ParserException: Base exception for parser";
}
```

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 ( )

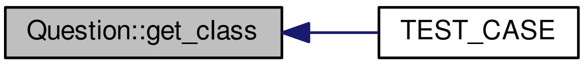
```
{
    /* Initializeaza o unsigned shortrebare */
    this->qname = NULL;

    bzero(this->qtype, 2);
    bzero(this->qclass, 2);

    this->qname_len = 0;
}
```

5.24.2 Member Function Documentation

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

```
{
    /* Returneaza clasa unei unsigned shortrebari
     *
     * @param[out] cls[2]
     *  Clasa unsigned shortrebari.
     */

    memcpy(cls, this->qclass, 2);
}
```

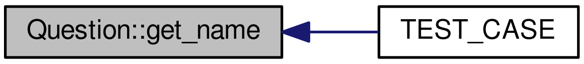
Here is the caller graph for this function:



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

```
{
    /* Returneaza numele unsigned shortrebari
     *
     * @param[out] *name
     *  Numele intrebari
     *
     * @param[out] lenght
     *  Lungimea stringului
     * */
     if (this->qname_len == 0)
     {
         *name = NULL;
         length = this->qname_len;
     }
     else
     {
        char *rname = new char[this->qname_len];
        memcpy(rname, this->qname, this->qname_len);
        *name = rname;
        length = this->qname_len;
     }
}
```

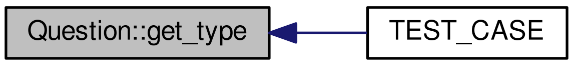
Here is the caller graph for this function:



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

```
{
    /* Returneaza tipul unei unsigned shortrebari
     *
     * @param[out] type[2]
     *  Tipul unsigned shortrebari.
     */

     memcpy(type, this->qtype, 2);
}
```

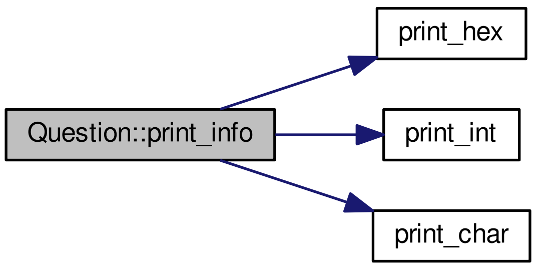
Here is the caller graph for this function:



5.24.2.4 void Question::print_info ( )

```
{
    /* Printeaza informatii despre o intrebare */
    std::cout <<"  Name :";
    print_hex(this->qname, this->qname_len, true);
    std::cout <<"  Name :";
    print_int(this->qname, this->qname_len, true);
    std::cout <<"  Name :";
    print_char(this->qname, this->qname_len, true);
    std::cout <<"  Lungime :" << (int)this->qname_len << std::endl;

    std::cout <<" Type: ";
    print_hex(this->qtype, 2, true);

    std::cout <<" Class: ";
    print_hex(this->qclass, 2, true);
    std::cout << std::endl;
}
```

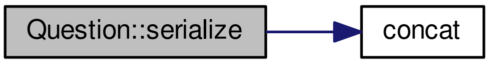
Here is the call graph for this function:



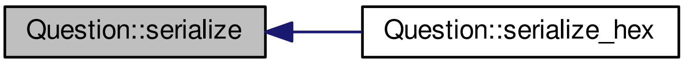
5.24.2.5 void Question::serialize (char ∗∗ *data,* unsigned short & *len*)

```
{
    /* Serializeaza obiectul curent */
    char aux[this->qname_len];
    memset(aux, 0, this->qname_len);
    memcpy(aux, this->qname, this->qname_len);
    concat(data, len, aux, (this->qname_len), NULL, 0);

    char aux_2[2];
    bzero(aux_2, sizeof(aux_2));
    memcpy(aux_2, this->qtype, 2);
    concat(data, len, *data, len, aux_2, 2);

    bzero(aux_2, sizeof(aux_2));
    memcpy(aux_2, this->qclass, 2);
    concat(data, len, *data, len, aux_2, 2);
}
```

Here is the call graph for this function:



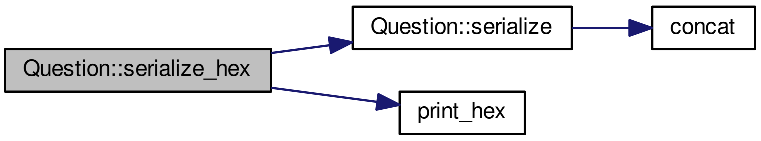
Here is the caller graph for this function:



5.24.2.6 void Question::serialize_hex ( )

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

Here is the call graph for this function:

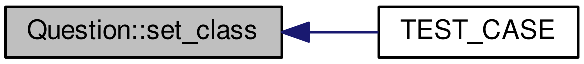


5.24.2.7 void Question::set_class (char *qclass[2]*)

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

```
     *
     * @param[in] cls[2]
     *  Clasa intrebari.
     */

     memcpy(this->qclass, cls, 2);
}
```

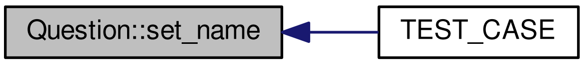
Here is the caller graph for this function:



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

```
{
    /* Seteaza un nume pentru unsigned shortrebare
     *
     * @param[in] *name
     *  Numele intrebari
     *
     * @param[in] lenght
     *  Lungimea stringului
     * */

     this->qname_len = length;

     this->qname = new char[length];
     memcpy(this->qname, name, length);
}
```

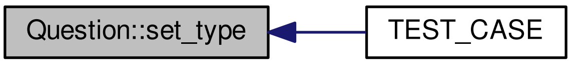
Here is the caller graph for this function:



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

```
{
    /* Seteaza tipul unei intrebari
     *
      * @param[in] type[2]
      *  Tipul intrebari.
      */
      memcpy(this->qtype, type, 2);
}
```

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 ( )

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

5.25.1.2 Reader::∼Reader ( )

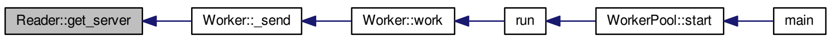
```
{
    /* Initializam un reader */
    if (this->buff != NULL)
    {
        free(this->buff);
    }
}
```

5.25.2 Member Function Documentation

5.25.2.1 Server ∗ Reader::get_server ( )

```
{
    /* Returneaza serverul de dns */
    return this->server;
}
```

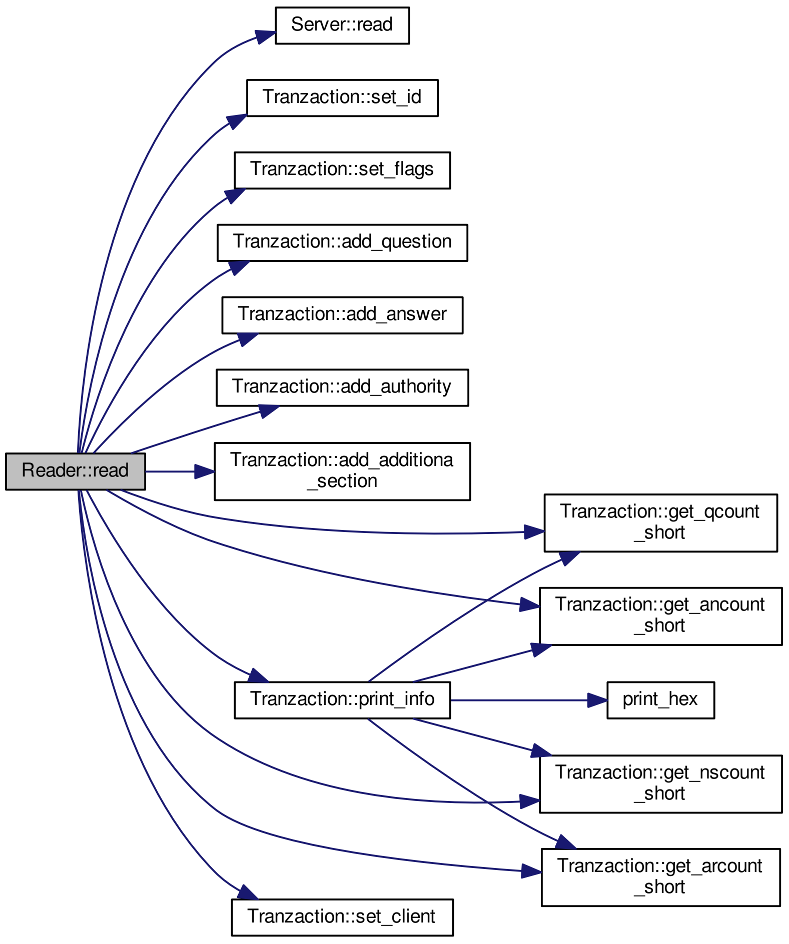
Here is the caller graph for this function:



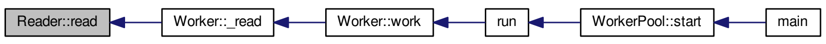
5.25.2.2 Tranzaction ∗ Reader::read ( )

```
{
    Tranzaction* tr = NULL;
    /* Citim pana cand dam de o tranzactie buna */
    bool found = false;
    while (found == false)
    {
        if (tr != NULL)
        {
            /* Distrugem tranzactia facuta anterior */
            delete tr;
        }
        tr = new Tranzaction();
        /* Citeste un pachet si returneaza o tranzactie */
        bzero(this->buff, sizeof(this->buff));
        int data_len = 0;

        /* informatii despre client */
        struct sockaddr client;
        /* NOTE(mmicu): len_client e parametru de intrare iesier */
        socklen_t len_client = sizeof(client);
        bzero(&client, sizeof(client));


        try
        {
            data_len = this->server->read(buff, 512, 0, (struct sockaddr*) &client, &len_client);
        }
        catch (ServerReadError& exp)
        {
            std::cout << "Malformed request (Eroare la citire) :"<< std::endl;
            for (int i=0; i<data_len; ++i)
            {
                std::cout << std::hex << (int)buff[i];
            }
            std::cout<< std::endl;

            continue;
        }

        if (data_len < 6 * 2)
        {
            /* Dimensiunea headerului */
            std::cout << "Malformed request (Prea scurt) :"<< std::endl;
            for (int i=0; i<data_len; ++i)
            {
                std::cout << std::hex << (int)buff[i];
            }
            std::cout<< std::endl;
        }
        /* Setam id-ul tranzactiei */
        char buff_2[2];
        memcpy(buff_2, this->buff, 2);
        tr->set_id(buff_2);

        /* Setam flags */
        memcpy(buff_2, &this->buff[2], 2);
        tr->set_flags(buff_2);

        unsigned short qcount=0,
                       ancount=0,
                       nscount=0,
                       arcount=0,
                       short_buff;

        memcpy(&short_buff, &this->buff[4], 2);
        qcount = ntohs(short_buff);

        memcpy(&short_buff, &this->buff[6], 2);
        ancount = ntohs(short_buff);

        memcpy(&short_buff, &this->buff[8], 2);
        nscount = ntohs(short_buff);

        memcpy(&short_buff, &this->buff[10], 2);
        arcount = ntohs(short_buff);

        unsigned short max_index;

        /* parsam intrebarile */
        std::vector<Question> aux = this->parser_question(
            this->buff, data_len, 12, qcount, max_index);

        for (std::vector<Question>::iterator it = aux.begin();
             it != aux.end(); ++it)
        {
            tr->add_question((*it));
        }

        /* Raspunsuri */

        std::vector<Resource> aux_res = this->parser_resource(
            this->buff, data_len, max_index, ancount, max_index);

        for (std::vector<Resource>::iterator it = aux_res.begin();
             it != aux_res.end(); ++it)
        {
            tr->add_answer((*it));
        }
        /* Authority */
        aux_res = this->parser_resource(
            this->buff, data_len, max_index, nscount, max_index);

        for (std::vector<Resource>::iterator it = aux_res.begin();
             it != aux_res.end(); ++it)
        {
            tr->add_authority((*it));
        }

        /* Additional */
        aux_res = this->parser_resource(
            this->buff, data_len, max_index, arcount, max_index);

        for (std::vector<Resource>::iterator it = aux_res.begin();
             it != aux_res.end(); ++it)
        {
            tr->add_additiona_section((*it));
        }


        /* Verificam daca am consumat tot continutul cum trebuie */
        if (qcount  != tr->get_qcount_short()   ||
            ancount != tr->get_ancount_short()  ||
            nscount != tr->get_nscount_short()  ||
            arcount != tr->get_arcount_short())
        {
            std::cout << "Malformed request (Nu am putut citi cate elemente trebuia) :"<< std::endl;
            found = false;
        }

        if (max_index == data_len)
        {
            found = true;
            tr->set_client(client);
        }
        else
        {
            std::cout << "Malformed request (Nu am consumat tot raspunsu) :"<< std::endl;
            found = false;
        }
        tr->print_info();
    } /* end while(found) */
    return tr;
}
```

Here is the call graph for this function:


Server::read
Tranzaction::set_id
Tranzaction::set_flags
Tranzaction::add_question
Tranzaction::add_answer
Tranzaction::add_authority
Reader::read
Tranzaction::add_additiona _section
Tranzaction::get_qcount _short
Tranzaction::get_ancount _short
Tranzaction::print_info
print_hex
Tranzaction::get_nscount _short
Tranzaction::get_arcount _short
Tranzaction::set_client

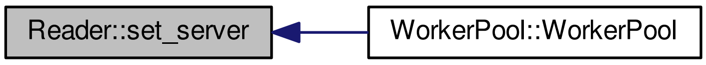
Here is the caller graph for this function:



5.25.2.3 void Reader::set_server (Server ∗ s)

```
{
    /* Seteaza un server
     *
     * @param[in] s
     *  Un obiect de tip Server
     */

    if (s == NULL)
    {
        throw ReaderValueError();
    }
    this->server = s;
48 }
```

Here is the caller graph for this function:



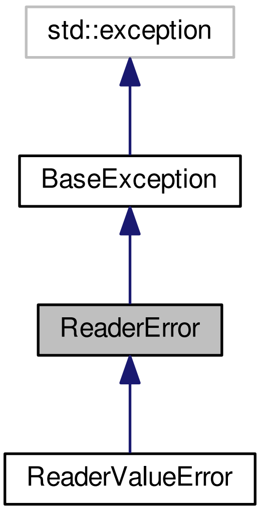
The documentation for this class was generated from the following files:

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

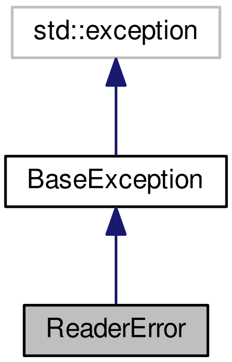
5.26 ReaderError Class Reference

`#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)

```
{
    return "Eroare in Reader";
}
```

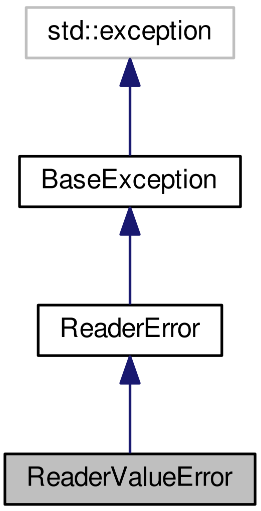
The documentation for this class was generated from the following files:

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

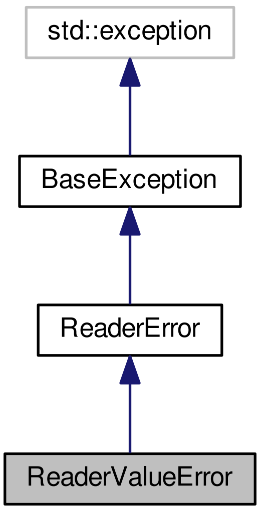
5.27 ReaderValueError Class Reference

```
#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)

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

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 ( )

```
{
    /* Initializarea unei resurse */

    bzero(this->type, 2);
    bzero(this->cls, 2);
    bzero(this->rdlength, 2);
    this->rdata = NULL;
    this->name = NULL;
    this->name_len = 0;
}
```

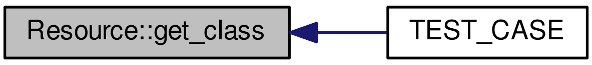
5.28.2 Member Function Documentation

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

```
{
     /* Returneaza clasa unei resurse
     *
     * @param[out] cls[2]
     *  Clasa resursei.
     */
```

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

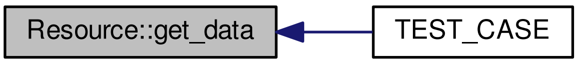
Here is the caller graph for this function:



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

```
{
    /* Setarea unor informatii despre resursa
     *
     * @param[out] data
     *  Informatia propriuzisa
     *
     * @param[out] length_data
     *  Lungimea informatiei
     */
    length_data = this->rdlength[0] << 8 | this->rdlength[1];
    if (length_data == 0)
    {
        *data = NULL;
    }
    else
    {
        *data = new char[length_data];
        memcpy(*data, this->rdata, length_data);
    }
}
```

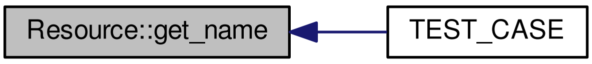
Here is the caller graph for this function:



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

```
{
    /* Returneaza numele Resursei
     *
     * @param[out] *name
     *  Numele Resursei
     *
     * @param[out] lenght
     *  Lungimea stringului
     */
    if (this->name_len == 0)
    {
        *name = NULL;
        length = this->name_len;
    }
    else
    {
        *name = new char[this->name_len];
        memcpy(*name, this->name, this->name_len);
        length = this->name_len;
    }
}
```

Here is the caller graph for this function:



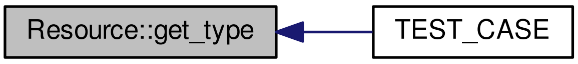
5.28.2.4 void Resource::get_ttl (char * *ttl*)

```
{
     /* Returneaza time tp live pentru o resurse
     *
     * @param[out] ttl[4]
     *  Time to live.
     */

     memcpy(ttl, this->ttl, 4);
}
```

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

```
{
    /* Returneaza tipul unei resurse
     *
     * @param[out] type[2]
     *  Tipul resursei.
     */

     memcpy(type, this->type, 2);
}
```

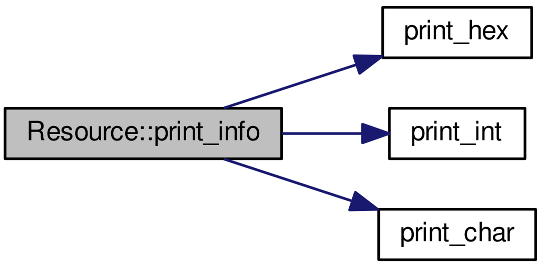
Here is the caller graph for this function:



5.28.2.6 void Resource::print_info ( )

```
{
    /* Printeaza informatii despre resurs */
    std::cout << "  Name :";
    print_hex(this->name, this->name_len, true);
    std::cout << "  Name :";
    print_int(this->name, this->name_len, true);
    std::cout << "  Name :";
    print_char(this->name, this->name_len, true);
    std::cout <<" Len. Name: " << this->name_len << std::endl;

    std::cout << "  Type:";
    print_hex(this->type, 2, true);

    std::cout << "  Class:";
    print_hex(this->cls, 2, true);

    std::cout << "  TTL:";
    print_hex(this->ttl, 4, true);

    unsigned short len;
    memcpy(&len, this->rdlength, 2);
    len = ntohs(len);

    std::cout << " Data:";
    print_hex(this->rdata, len, true);

    std::cout << " Data:";
    print_int(this->rdata, len, true);

    std::cout << " Data:";
    print_char(this->rdata, len, true);

    std::cout << "  Len. Data:" << len << std::endl << std::endl;
}
```

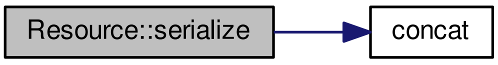
Here is the call graph for this function:



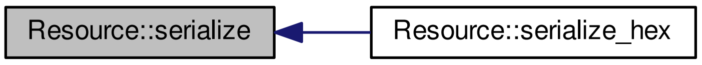
5.28.2.7 void Resource::serialize (char ** *data*, unsigned short & *len*)

```
{
    /* Serializeaza obiectul curent */
    char aux[this->name_len];
    memset(aux, 0, this->name_len);
    memcpy(aux, this->name, this->name_len);
    concat(data, data_len, aux, this->name_len, NULL, 0);

    char aux_2[2];
    bzero(aux_2, sizeof(aux_2));
    memcpy(aux_2, this->type, 2);
    concat(data, data_len, *data, data_len, aux_2, 2);

    bzero(aux_2, sizeof(aux_2));
    memcpy(aux_2, this->cls, 2);
    concat(data, data_len, *data, data_len, aux_2, 2);

    char aux_4[4];
    bzero(aux_4, sizeof(aux_4));
    memcpy(aux_4, this->ttl, 4);
    concat(data, data_len, *data, data_len, aux_4, 4);

    /* rdlen */
    bzero(aux_2, sizeof(aux_2));
    memcpy(aux_2, this->rdlength, 2);
    concat(data, data_len, *data, data_len, aux_2, 2);

    /* rdata */
    unsigned short len;
    memcpy(&len, this->rdlength, 2);
    len = ntohs(len);

    char aux_data[len+1];
    memset(aux_data, 0, len+1);
    memcpy(aux_data, this->rdata, len);
    concat(data, data_len, *data, data_len, aux_data, len);
}
```

Here is the call graph for this function:



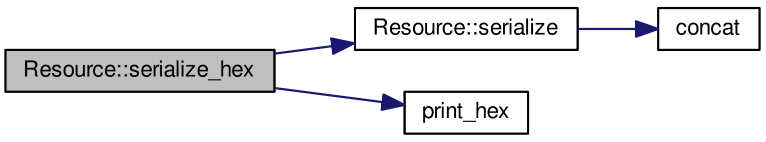
Here is the caller graph for this function:



5.28.2.8 void Resource::serialize_hex ( )

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

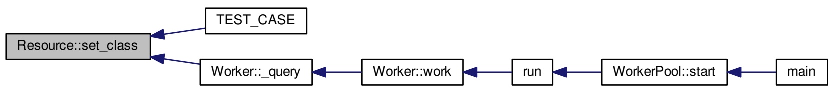
Here is the call graph for this function:



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

```
{
    /* Seteaza clasa unei resurse
     *
     * @param[in] cls[2]
     *  Clasa resurseri.
     */

     memcpy(this->cls, cls, 2);

}
```

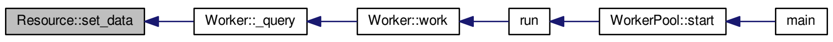
Here is the caller graph for this function:



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

```
{
    /* Setarea unor informatii despre resursa
     *
     * @param[in] data
     *  Informatia propriuzisa
     *
     * @param[in] length_data
     *  Lungimea informatiei
     */

    this->rdata = new char[length_data];
    memcpy(this->rdata, data, length_data);
    memcpy(this->rdlength, (char*)&length_data, 2);
}
```

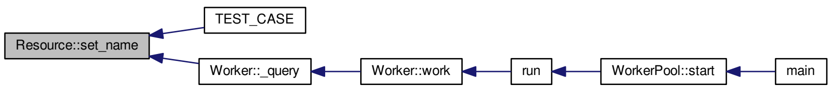
Here is the caller graph for this function:



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

```
{
    /* Seteaza un nume pentru o resursa
     *
     * @param[in] *name
     *  Numele resursei
     *
     * @param[in] lenght
     *  Lungimea stringului
     * */
     this->name_len = length;

     this->name = new char[length];
     memcpy(this->name, name, length);
}
```

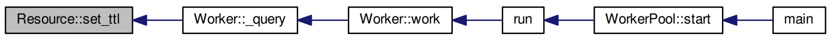
Here is the caller graph for this function:



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

```
{
    /* Seteaza time to live unei resurse
     *
     * @param[in] ttl[4]
     *  time to live pentru  resurseri.
     */

     memcpy(this->ttl, ttl, 4);
}
```

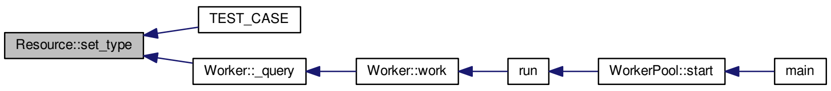
Here is the caller graph for this function:



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

```
{
    /* Seteaza tipul unei resurse
     *
     * @param[in] type[2]
     *  Tipul resurse.
     */

     memcpy(this->type, type, 2);
}
```

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*)

```
{
    /* Initializeaza un server
     *
     * @param[in] port
     *  Portul pe care o sa deschidem servarul
     *
     * @param[in] backlog
     *  Dimensiunea cozi pentru listen (Folosit doar pentru TCP).
     * */
    this->port = port;
    this->backlog = backlog;
    this->sock = 0;

    /* umplem structura folosita de server */
    bzero(&this->server, sizeof(this->server));
    this->server.sin_family = AF_INET;
    server.sin_addr.s_addr = htonl (INADDR_ANY);
    server.sin_port = htons (this->port);

}
```

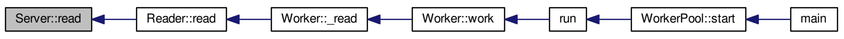
5.29.2 Member Function Documentation

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*)

```
{
    /* Citeste informatii de pe retea.
     *
     * @param[out] buf
     *  Unde o sa salvam datele primite
     *
     * @param[in] len
     *  Lungimea buffarului unde salvam datele
     *
     * @param[in] flags
     *  Flagurile pentru operatia de citire
     *
     * @param[out] src_addr
     *  sockaddr structura umpluta cu informatiile clientului
     *  de la care am citit date
     *
     * @param[in, out] addrlen
     *  La intreare accepta dimensiune structuri src_addr
     *  La intreare returneaza dimensiune structuri src_addr completata
     */
    if (this->sock == 0)
    {
        /* Daca nu am pornit servarul */
        throw ServerNotOpen();
    }

    bzero(src_addr, sizeof(src_addr));
    bzero(buf, len);
    ssize_t msglen;
    msglen = recvfrom(this->sock, buf, len, 0, src_addr, addrlen);

    if (msglen < 0)
    {
        /* Avem o eroare */
        perror("Reading from server !\n");
        throw ServerReadError();
    }

    return msglen;
}
```

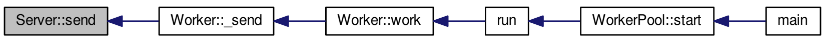
Here is the caller graph for this function:



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

```
{
    /* Trimitem informatii de pe retea.
     *
     * @param[in] buf
     *  Buffer cu infromatiile pe care vrem sa le trimitem
     *
     * @param[in] len
     *  Lungimea buffarului u
     *
     * @param[in] flags
     *  Flagurile pentru operatia de citire
     *
     * @param[in] dest_addr
     *  sockaddr structura umpluta cu informatiile clientului
     *  (unde vrem sa trimitem)
     *
     * @param[in] addrlen
     *  Lungimea structuri sockaddr
     */
    if (this->sock == 0)
    {
        /* Daca nu am pornit servarul */
        throw ServerNotOpen();
    }

    ssize_t msglen = 1;
    /* NOTE(mmicu): msglen nu va fi nici o data negativ in
     * while de asta putem face cast
     */
    while (static_cast<size_t>(msglen) < len)
    {
        /* Cat timp nu s-au trimiti toate informatiile */
        msglen = sendto(this->sock, buf, len, 0,
                        dest_addr, addrlen);

        if (msglen < 0)
        {
            /* Avem o eroare */
            perror("Sending from server !\n");
            throw ServerReadError();
        }
    }


    return msglen;
}
```

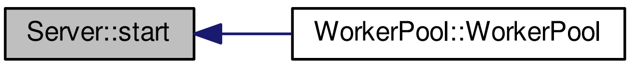
Here is the caller graph for this function:



5.29.2.4 void Server::start ( )

```
{
    /* Porneste un server UDP pe local host */
    if ((this->sock = socket(this->server.sin_family, SOCK_DGRAM, 0)) == -1)
    {
      perror ("[server]Eroare la socket().\n");
      throw SocketException();
    }

    /* Accepta reutilizarea portului */
    int on=1;
    setsockopt(this->sock, SOL_SOCKET, SO_REUSEADDR, &on, sizeof(on));

    /* facem bind la socket cu (adresa, port) */
    if (bind (this->sock, (struct sockaddr *) &this->server, sizeof (struct sockaddr)) == -1)
    {
      perror ("[server]Eroare la bind().\n");
      throw BindException();
    }
}
```

Here is the caller graph for this function:

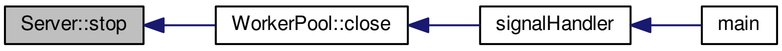


5.29.2.5 void Server::stop ( )

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

```
            throw NotOpenException();

        }
        close(this->sock);
        this->sock = 0;
    }
```

Here is the caller graph for this function:



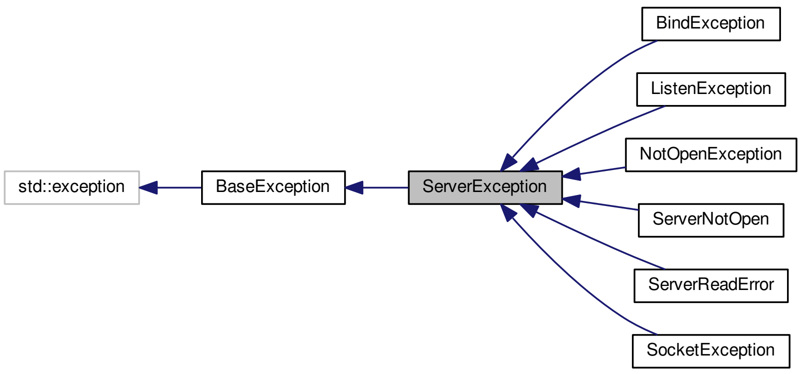
The documentation for this class was generated from the following files:

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

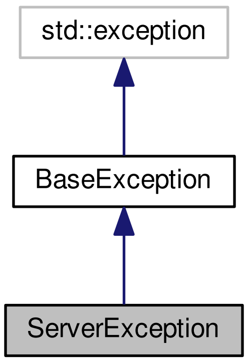
5.30 ServerException Class Reference

`#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)

```
{
    return "Eroare la crearea serverului !";
}
```

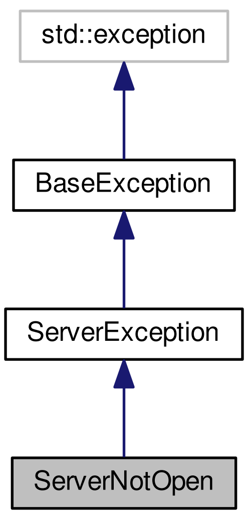
The documentation for this class was generated from the following files:

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

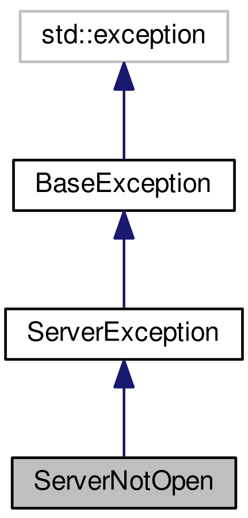
5.31 ServerNotOpen Class Reference

`#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)

```
{
    return "Servarul nu este deschis !";
}
```

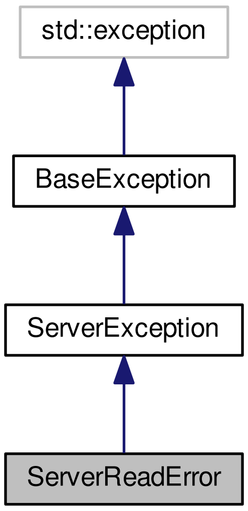
The documentation for this class was generated from the following files:

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

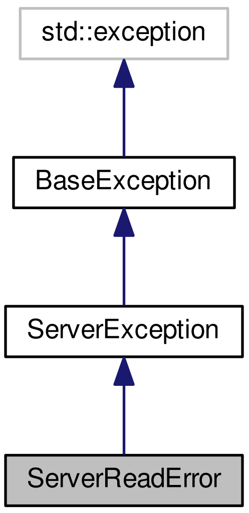
5.32 ServerReadError Class Reference

`#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)

```
{
    return "Eroare la citire !";
}
```

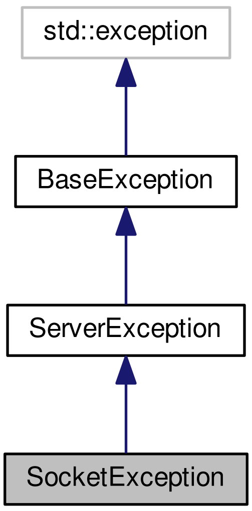
The documentation for this class was generated from the following files:

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

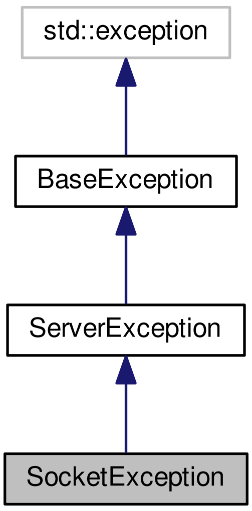
5.33 SocketException Class Reference

```
#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)

```
{
    return "Eroare la crearea socketului !";
}
```

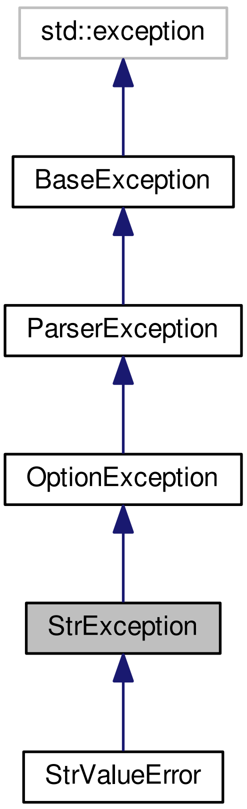
The documentation for this class was generated from the following files:

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

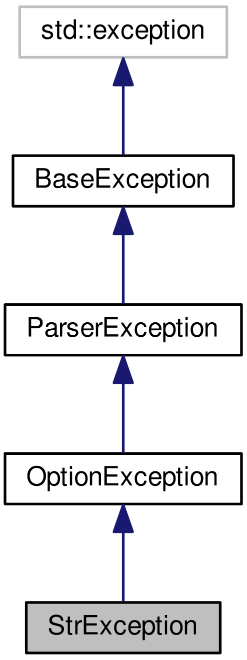
5.34 StrException Class Reference

`#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*)

```
                                              : OptionException(
      primit)
{
}
```

5.34.2 Member Function Documentation

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

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

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

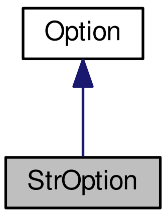
- dns/exceptions.h

- dns/exceptions.cpp

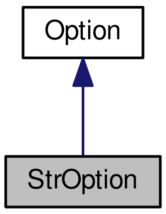
5.35 StrOption Class Reference

`#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*)

```
                                                             :
  Option(short_name, long_name, help_message,
required)
{
  /* Constructor pentru StrOption
   *
   * @param[in] short_name
   *  Un char care reprezinta varianta prescurtata a parametrului
   *  Ex: -d
   *
   * @param[in] long_name
   *  Un string care reprezinta varianta lunga aparametrului
   *  Ex: --delimitator
   *
   * @param[in] help_string
   *  Un string care reprezinta descrierea optiuni
   *
   * @param[in] required
   *  Daca un parametru este necesar sau nu, implicit este setat pe false
   */
}
```

5.35.2 Member Function Documentation

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

Reimplemented from Option.

```
{
  /* \return{Returneaza valoarea parametrului} */
  return this->value;
}
```

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

Reimplemented from Option.

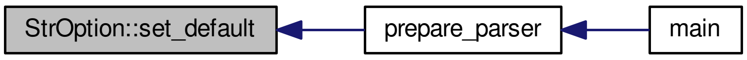
```
{
  /* Returneaza un string care reprezinta tipul optinui */
  return std::string("str");
}
```

5.35.2.3 bool StrOption::is_required ( )

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

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

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.

```
{
    /* Seteaza valoarea unui parametru convorm unui string
     *
     * @param parameter
     *  Stringul care contine valoarea parametrului
     */
    this->value = parameter;
    this->is_set = true;
}
```

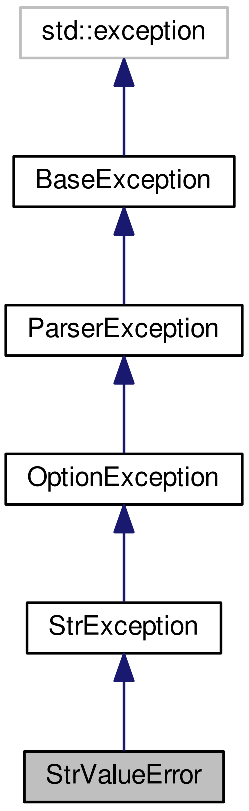
The documentation for this class was generated from the following files:

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

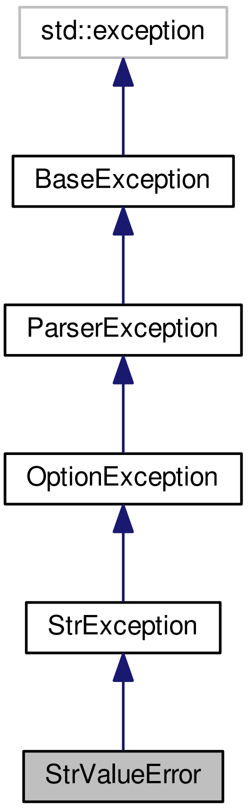
5.36 StrValueError Class Reference

`#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*)

```
                                          : StrException(primit)
{
}
```

5.36.2 Member Function Documentation

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

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

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 ( )

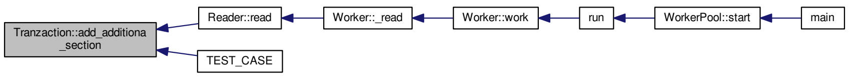
```
{
    /* Initializeaza o tranzactie */
    bzero(this->id, 2);
    bzero(this->flags, 2);
    bzero(this->qcount, 2);
    bzero(this->ancount, 2);
    bzero(this->nscount, 2);
    bzero(this->arcount, 2);
    this->questions.clear();
    this->answers.clear();
    this->authority.clear();
    this->additional_sections.clear();
}
```

5.37.2 Member Function Documentation

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

```
{
    /* Adauga resurse aditionale
     *
     * @param res
     *  Resursa , o instanta a clasei `Resource`
     */

    this->additional_sections.push_back(res);

    /* Updateaza numarul de autoritati */
    unsigned short size = this->additional_sections.size();
    memcpy(this->arcount, (char*)&(size), 2);
}
```

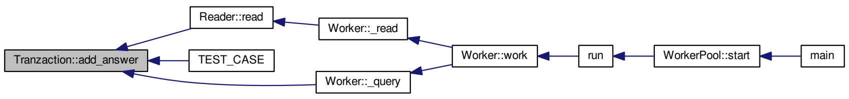
Here is the caller graph for this function:



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

```
{
    /* Adauga un raspuns
     *
     * @param aut
     *  Raspuns, o instanta a clasei `Resource`
     */

    this->answers.push_back(ans);

    /* Updateaza numarul de raspunsuri */
    unsigned short size = this->answers.size();
    memcpy(this->ancount, (char*)&(size), 2);
}
```

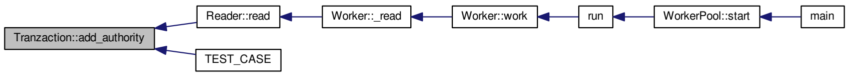
Here is the caller graph for this function:



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

```
{
    /* Adauga o autoritate tranzactiei
     *
     * @param aut
     *  Autoritatea, o instanta a clasei `Resource`
     */

    this->authority.push_back(aut);

    /* Updateaza numarul de autoritati */
    unsigned short size = this->authority.size();
    memcpy(this->nscount, (char*)&(size), 2);
}
```

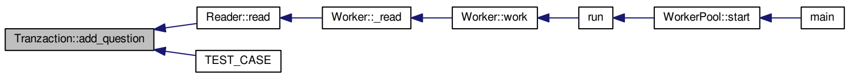
Here is the caller graph for this function:



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

```
{
    /* Adauga un query (intrebare) tranzactiei
     *
     * @param qt
     *  Intrebarea, o instanta a clasei `Question`
     */
    this->questions.push_back(qt);

    /* Updateaza numarul de intrebari */

    /* NOTE(mmicu): Acest lucru ar trebui eliminat, putem avea doar un getter
     * peste `qcount` care apeleaza `.length` pe `this->questions`
     */
    unsigned short size = this->questions.size();
    memcpy(this->qcount, (char*)&(size), 2);
}
```

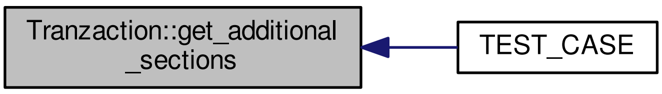
Here is the caller graph for this function:



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

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

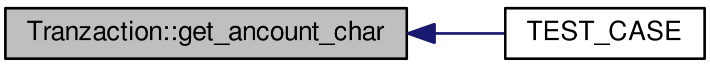
Here is the caller graph for this function:



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

```
{
    /* Returneza numarul de raspunsuri
     * sub forma unui char[2]
     *
     * @param r_ancount
     *  Numarul de raspunsuri
     */
    memcpy(r_ancount, this->ancount, 2);
}
```

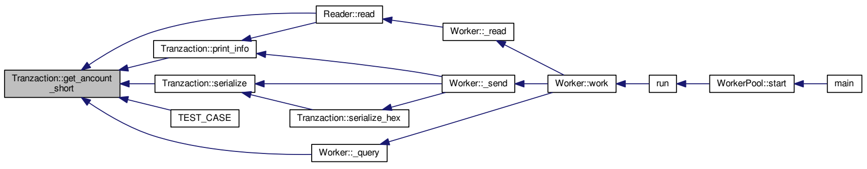
Here is the caller graph for this function:



5.37.2.7 unsigned short Tranzaction::get_ancount_short ( )

```
{
    /* Returneza numarul de raspunsuri
     * sub forma unui unsigned short
     */
    unsigned short count = this->_get_short_from_char(this->ancount);
    return count;
}
```

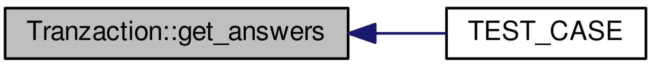
Here is the caller graph for this function:



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

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

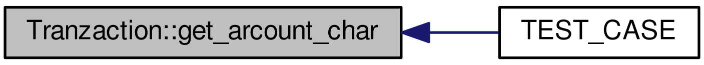
Here is the caller graph for this function:



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

```
{
    /* Returneza numarul de additional records
     * sub forma unui char[2]
     *
     * @param r_arcount
     *  Numarul de additional records
     */
    memcpy(r_arcount, this->nscount, 2);
}
```

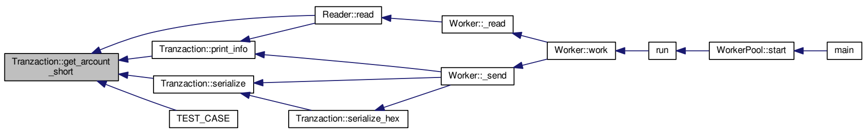
Here is the caller graph for this function:



5.37.2.10 unsigned short Tranzaction::get_arcount_short ( )

```
{
    /* Returneza numarul de name server authority
     * sub forma unui unsigned short
     */
    unsigned short count = this->_get_short_from_char(this->arcount);
    return count;
}
```

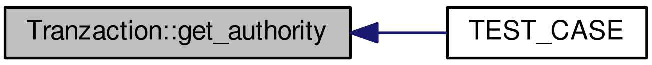
Here is the caller graph for this function:



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

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

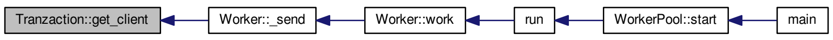
Here is the caller graph for this function:



5.37.2.12 sockaddr Tranzaction::get_client ( )

```
{
    /* Returnam clientul */
    return this->client;
}
```

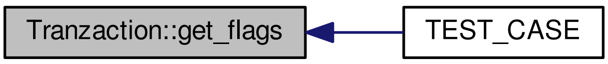
Here is the caller graph for this function:



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

```
{
    /* Returneaza flagurile unei tranzactii
     *
     * @param[out] flags
     *  Flagurile tranzactiei
     */
    memcpy(flags, this->flags, 2);
}
```

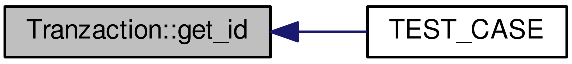
Here is the caller graph for this function:



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

```
{
    /* Returneaza id-ul unei tranzactii
     *
     * @param[out] id
     *  Id-ul tranzactiei
     */
    memcpy(id, this->id, 2);
}
```

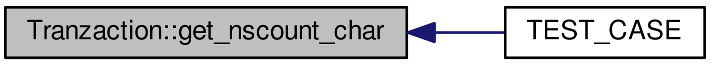
Here is the caller graph for this function:



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

```
{
    /* Returneza numarul de name server authority
     * sub forma unui char[2]
     *
     * @param r_nscount
     *  Numarul de name server authority
     */
    memcpy(r_nscount, this->nscount, 2);
}
```

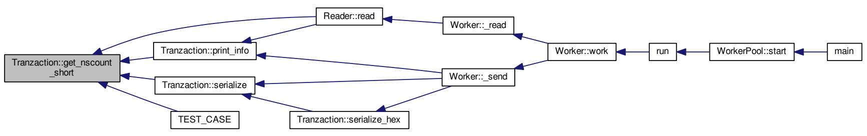
Here is the caller graph for this function:



5.37.2.16 unsigned short Tranzaction::get_nscount_short ( )

```
{
    /* Returneza numarul de name server authority
     * sub forma unui unsigned short
     */
    unsigned short count = this->_get_short_from_char(this->nscount);
    return count;
}
```

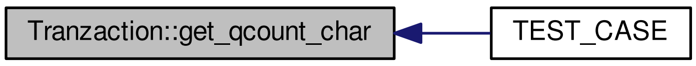
Here is the caller graph for this function:



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

```
{
    /* Returneza numarul de query(intrebari)
     * sub forma unui char[2]
     *
     * @param r_qcount
     *  Numarul de query
     */
    memcpy(r_qcount, this->qcount, 2);

}
```

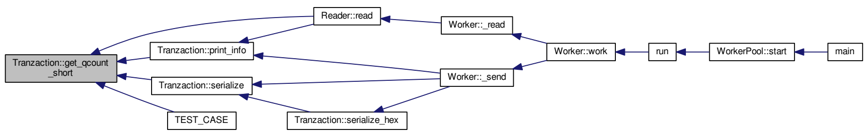
Here is the caller graph for this function:



5.37.2.18 unsigned short Tranzaction::get_qcount_short ( )

```
{
    /* Returneaza numarul de query(intrebari)
     * sub forma unui unsigned short
     */
    unsigned short count = this->_get_short_from_char(this->qcount);
    return count;
}
```

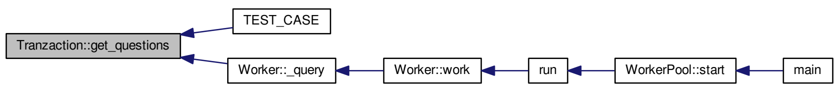
Here is the caller graph for this function:



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

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

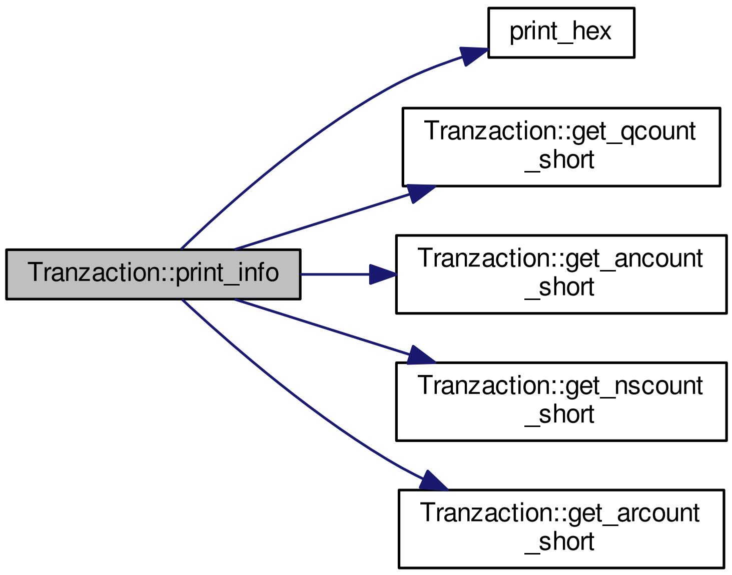
Here is the caller graph for this function:



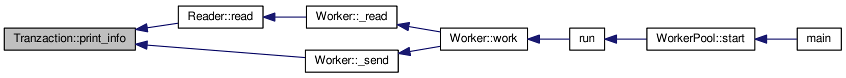
5.37.2.20 void Tranzaction::print_info ( )

```
{
    /* Printeaza infomatii despre tranzactie */
    std::cout << " ------------------ " << std::endl;
    std::cout << "Tranzactie id :";
    print_hex(this->id, 2, true);

    std::cout << "Flags :";
    print_hex(this->flags, 2, true);

    std::cout << "Qcount :" << this->get_qcount_short() << std::endl;
    std::cout << "Ancount:" << this->get_ancount_short() << std::endl;
    std::cout << "Nscount:" << this->get_nscount_short() << std::endl;
    std::cout << "Arcount :" << this->get_arcount_short() << std::endl;

    std::cout << "Questions :" << std::endl;
    for (std::vector<Question>::iterator it = this->questions.begin();
         it != this->questions.end(); ++it)
    {
        (*it).print_info();
    }

    std::cout << "Answers:" << std::endl;
    for (std::vector<Resource>::iterator it = this->answers.begin();
         it != this->answers.end(); ++it)
    {
        (*it).print_info();
    }

    std::cout << "Authority:" << std::endl;
    for (std::vector<Resource>::iterator it = this->authority.begin();
         it != this->authority.end(); ++it)
    {
        (*it).print_info();
    }

    std::cout << "Additional sections:" << std::endl;
    for (std::vector<Resource>::iterator it = this->additional_sections.begin();
         it != this->additional_sections.end(); ++it)
    {
        (*it).print_info();
    }

    std::cout << " ------------------ " << std::endl << std::endl;
}
```

Here is the call graph for this function:


print_hex
Tranzaction::get_qcount_short
Tranzaction::print_info
Tranzaction::get_ancount_short
Tranzaction::get_nscount_short
Tranzaction::get_arcount_short

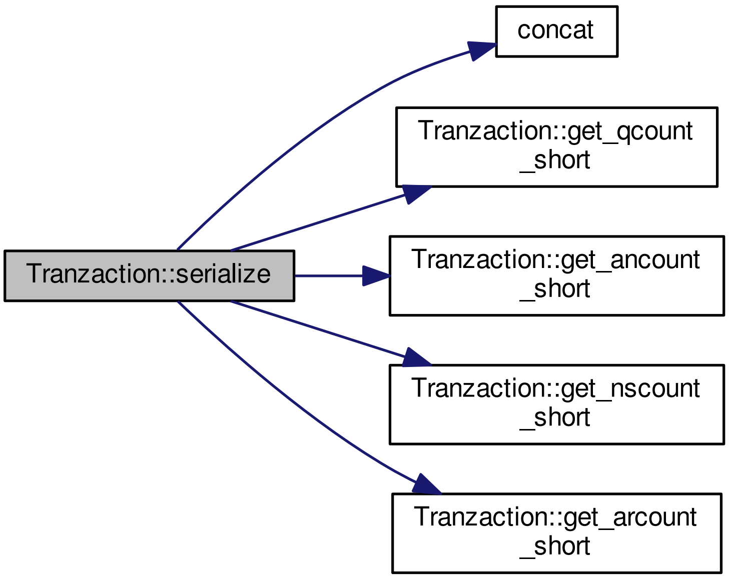
Here is the caller graph for this function:



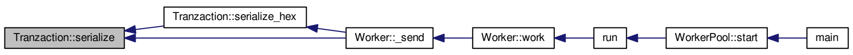
5.37.2.21 void Tranzaction::serialize (char ** *data*, unsigned short & *len*)

```
{
    /* Serializeaza obiectul curent
     *
     * @param[out] data
     * pointer catre array
     *
     * @param[out] len
     * lungimea
     */

    /* id */
    char aux_2[3];
    bzero(aux_2, sizeof(aux_2));
    memcpy(aux_2, this->id, 2);
    concat(data, len, aux_2, 2, NULL, 0);

    /* flags */
    bzero(aux_2, sizeof(aux_2));
    memcpy(aux_2, this->flags, 2);
    concat(data, len, *data, len, aux_2, 2);

    unsigned short s = htons(this->get_qcount_short());
    bzero(aux_2, sizeof(aux_2));
    memcpy(aux_2, (char*)&s, 2);
    concat(data, len, *data, len, aux_2, 2);


    s = htons(this->get_ancount_short());
    bzero(aux_2, sizeof(aux_2));
    memcpy(aux_2, (char*)&s, 2);
    concat(data, len, *data, len, aux_2, 2);

    s = htons(this->get_nscount_short());
    bzero(aux_2, sizeof(aux_2));
    memcpy(aux_2, (char*)&s, 2);
    concat(data, len, *data, len, aux_2, 2);

    s = htons(this->get_arcount_short());
    bzero(aux_2, sizeof(aux_2));
    memcpy(aux_2, (char*)&s, 2);
    concat(data, len, *data, len, aux_2, 2);

    char *aux_data;
    unsigned short aux_data_len;

    for (std::vector<Question>::iterator it = this->questions.begin();
         it != this->questions.end(); ++it)
    {
        (*it).serialize(&aux_data, aux_data_len);
        concat(data, len, *data, len, aux_data, aux_data_len);
    }

    for (std::vector<Resource>::iterator it = this->answers.begin();
         it != this->answers.end(); ++it)
    {
        (*it).serialize(&aux_data, aux_data_len);
        concat(data, len, *data, len, aux_data, aux_data_len);
    }

    for (std::vector<Resource>::iterator it = this->authority.begin();
         it != this->authority.end(); ++it)
    {
        (*it).serialize(&aux_data, aux_data_len);
        concat(data, len, *data, len, aux_data, aux_data_len);
    }

    for (std::vector<Resource>::iterator it = this->additional_sections.begin();
         it != this->additional_sections.end(); ++it)
    {
        (*it).serialize(&aux_data, aux_data_len);
        concat(data, len, *data, len, aux_data, aux_data_len);
    }
}
```

Here is the call graph for this function:


concat
Tranzaction::get_qcount_short
Tranzaction::serialize
Tranzaction::get_ancount_short
Tranzaction::get_nscount_short
Tranzaction::get_arcount_short

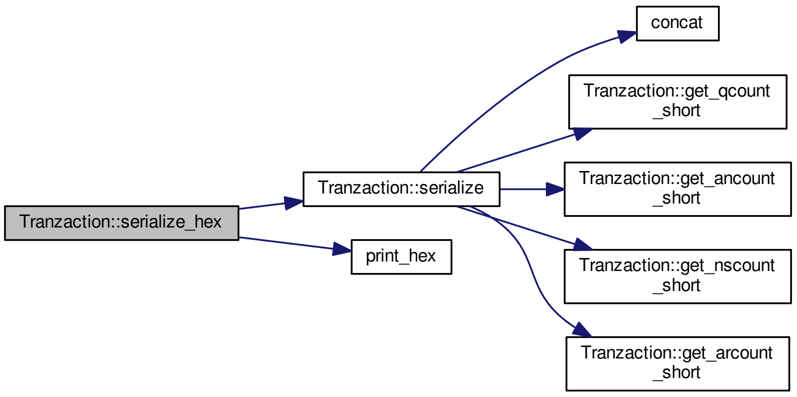
Here is the caller graph for this function:



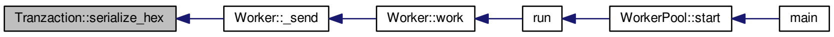
5.37.2.22 void Tranzaction::serialize_hex ( )

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

Here is the call graph for this function:


concat
Tranzaction::get_qcount_short
Tranzaction::serialize
Tranzaction::get_ancount_short
Tranzaction::serialize_hex
print_hex
Tranzaction::get_nscount_short
Tranzaction::get_arcount_short

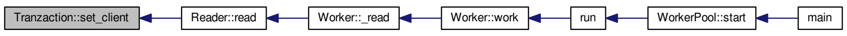
Here is the caller graph for this function:



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

```
{
    /* Setam un client pentru tranzactia asta
     *
     * @param[in] client
     *  Clientul pe care dorim sa il setam
     */
    this->client = client;
}
```

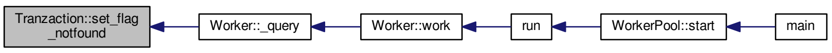
Here is the caller graph for this function:



5.37.2.24 void Tranzaction::set_flag_notfound ( )

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

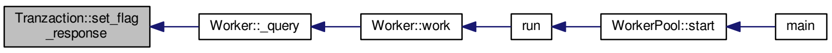
Here is the caller graph for this function:



5.37.2.25 void Tranzaction::set_flag_response ( )

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

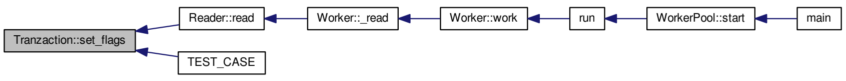
Here is the caller graph for this function:



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

```
{
    /* Seteaza flagurile unei tranzactii
     *
     * @param[in] flags
     *  Flagurile tranzactiei
     * */
    memcpy(this->flags, flags, 2);
}
```

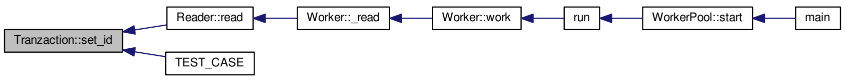
Here is the caller graph for this function:



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

```
{
    /* Seteaza id-ul unei tranzactii
     *
     * @param[in] id
     *  Id-ul tranzactiei
     */
    memcpy(this->id, id, 2);
}
```

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*)

```
                                                                        :
      parser(pr)
{
      /* Pregatim un worker
       *
       * @param[in] db_name
       *  Numele bazei de date
       *
       * @param[in] *lock
       *  Lock-ul pentru pool
       *
       * @param[in] *rd
       *  Un obiect de tip reader
       *
       * @param[in] *pr
       *  Un parser sa vedem optiunile de la CLI
       *
       * @param[in] id
       *  Id-ul pentru acest worker
       */
      this->id = id;
      std::cout << "Pornim Worker - "<< this->id << std::endl;
      this->parser = pr;
      this->db = new DB(db_name);
      this->lock = lock;

      this->rd = rd;
      this->stop = false;
}
```

5.38.1.2 Worker::∼Worker ( )

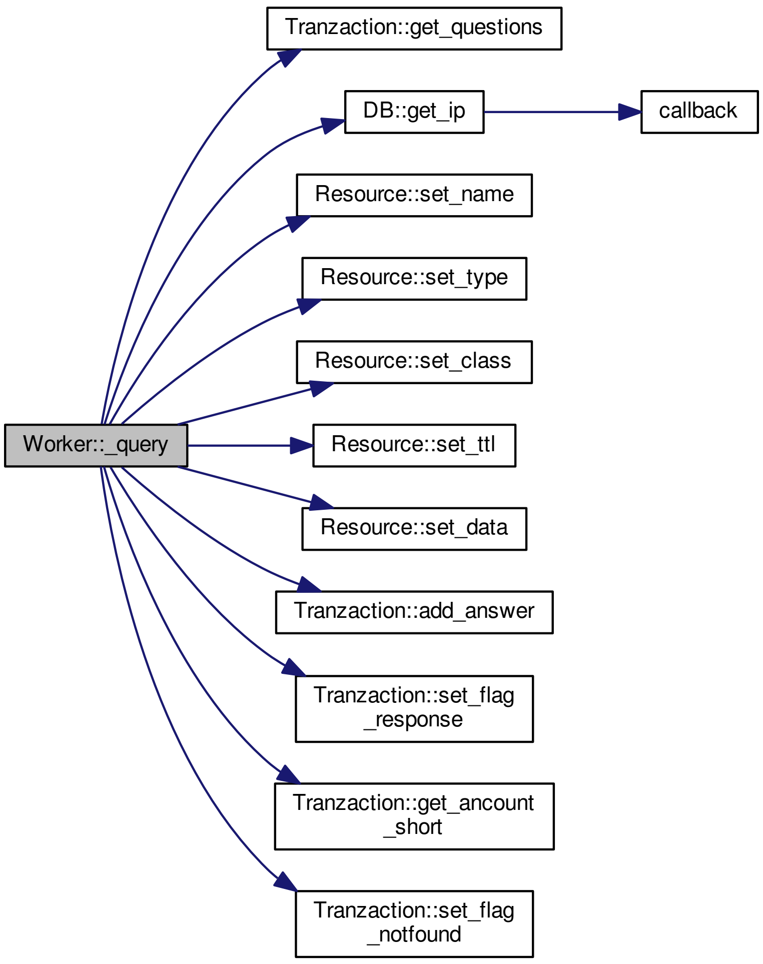
```
{
    /* Deletam tot */
    delete this->db;
}
```

5.38.2 Member Function Documentation

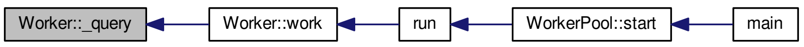
5.38.2.1 void Worker::_query (Tranzaction ∗ *tr*)

```
{
    /* Pregateste raspunsurile
     *
     * @param[in, out] tr
     *  Tranzactia care are intrebarile, la iesire
     *  va fi populata cu raspunsuri
     */

    std::vector<Question> aux = tr->get_questions();
    for (std::vector<Question>::iterator it = aux.begin();
         it != aux.end(); ++it)
    {
        char *domain;
        unsigned short len;
        (*it).get_name(&domain, len);
        std::string ip = this->db->get_ip(domain, len);


        if (ip != std::string(""))
        {
            /* Daca am gasit un ip */
            Resource res;
            res.set_name(domain, len);

            char generic[2]; /* generic 00 01 */
            generic[0] = 0;
            generic[1] = 1;

            res.set_type(generic);
            res.set_class(generic);

            char generic_ttl[4]; /* generic 00 00 00 30 */
            generic_ttl[0] = 0;
            generic_ttl[1] = 0;
            generic_ttl[2] = 0;
            generic_ttl[3] = 30;

            res.set_ttl(generic_ttl);

            in_addr_t ip_addr = inet_addr(ip.c_str());
            /* NOTE(mmicu): 4 e hardcodat, desi asta o sa fie dimensiuena unui IPv4
             * mereu ar trebui sa folosim sizeof */

            std::cout << "ip_addr" << ip_addr << std::endl;
            unsigned short len = 4;
            len = htons(len);
            res.set_data((char*)&ip_addr, len);

            tr->add_answer(res);
        }

        delete domain;
    }

    tr->set_flag_response();
    if (tr->get_ancount_short() == 0)
    {
        /* Nu avem nici un raspuns */
        tr->set_flag_notfound();
    }
}
```

Here is the call graph for this function:



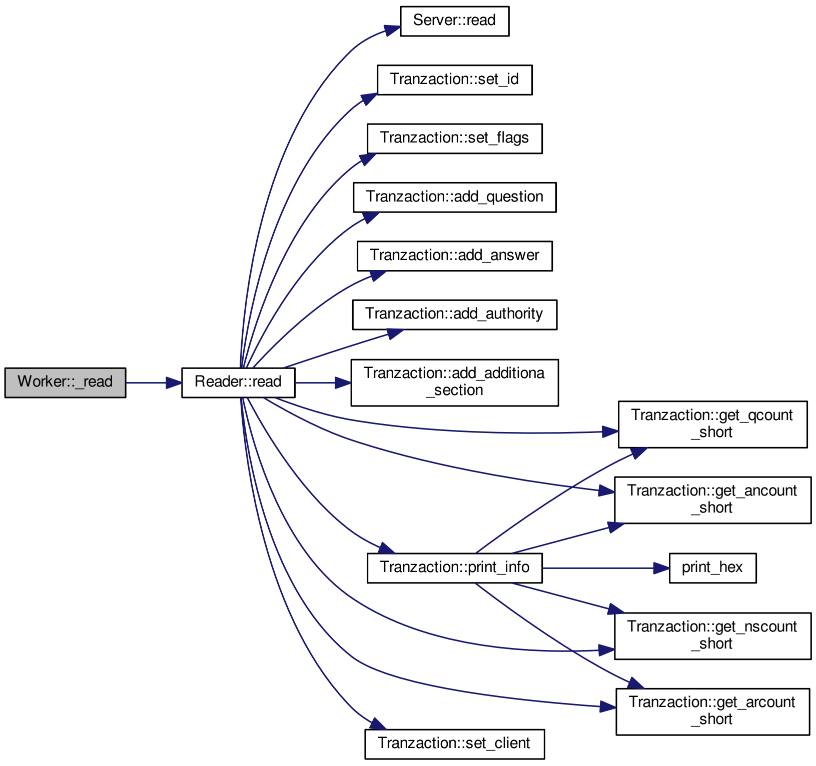
Here is the caller graph for this function:



5.38.2.2 Tranzaction ∗ Worker::_read ( )

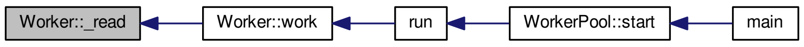
```
{
    /* Citim o noua tranzactie */
    Tranzaction *tr = NULL;
    try
    {
        tr = this->rd->read();
    }
    catch (BaseException* ex)
    {
        std::cout << "[" << this->id << "] Eroare la citire !" << std::endl;
    }


    return tr;
}
```

Here is the call graph for this function:



Worker::_read
Reader::read
Server::read
Tranzaction::set_id
Tranzaction::set_flags
Tranzaction::add_question
Tranzaction::add_answer
Tranzaction::add_authority
Tranzaction::add_additiona _section
Tranzaction::get_qcount _short
Tranzaction::get_ancount _short
Tranzaction::print_info
print_hex
Tranzaction::get_nscount _short
Tranzaction::get_arcount _short
Tranzaction::set_client

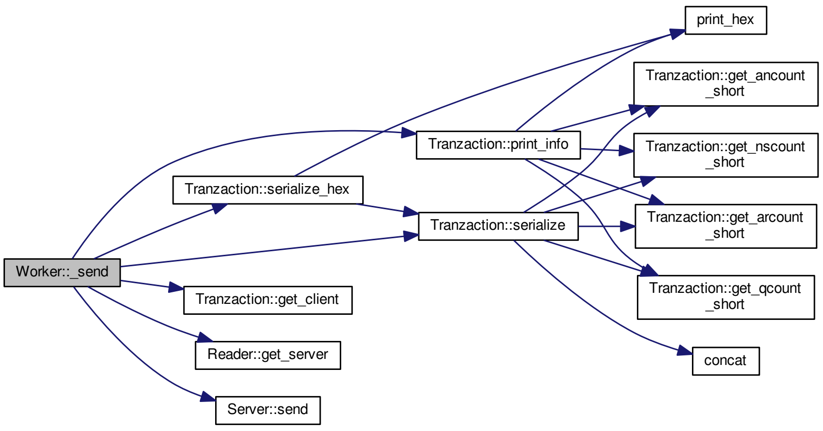
Here is the caller graph for this function:



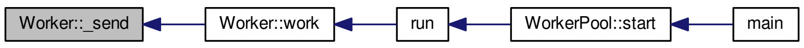
5.38.2.3 void Worker::_send (Tranzaction * *tr*)

```
{
    /* Serializeaza tranzactia si trimite raspunsul */
    tr->print_info();

    char* data;
    unsigned short data_len;
    tr->serialize(&data, data_len);

    std::cout << "Serializare :";
    tr->serialize_hex();

    struct sockaddr client = tr->get_client();


    try
    {
        this->rd->get_server()->send(data, data_len, 0, (struct sockaddr*)&client, sizeof(
client));
    }
    catch (BaseException* ex)
    {
        std::cout << "[" << this->id << "] Trimitere - eroare" << std::endl;
    }


}
```

Here is the call graph for this function:


print_hex
Tranzaction::get_ancount_short
Tranzaction::print_info
Tranzaction::get_nscount_short
Tranzaction::serialize_hex
Tranzaction::serialize
Tranzaction::get_arcount_short
Worker::_send
Tranzaction::get_qcount_short
Tranzaction::get_client
Reader::get_server
concat
Server::send

Here is the caller graph for this function:



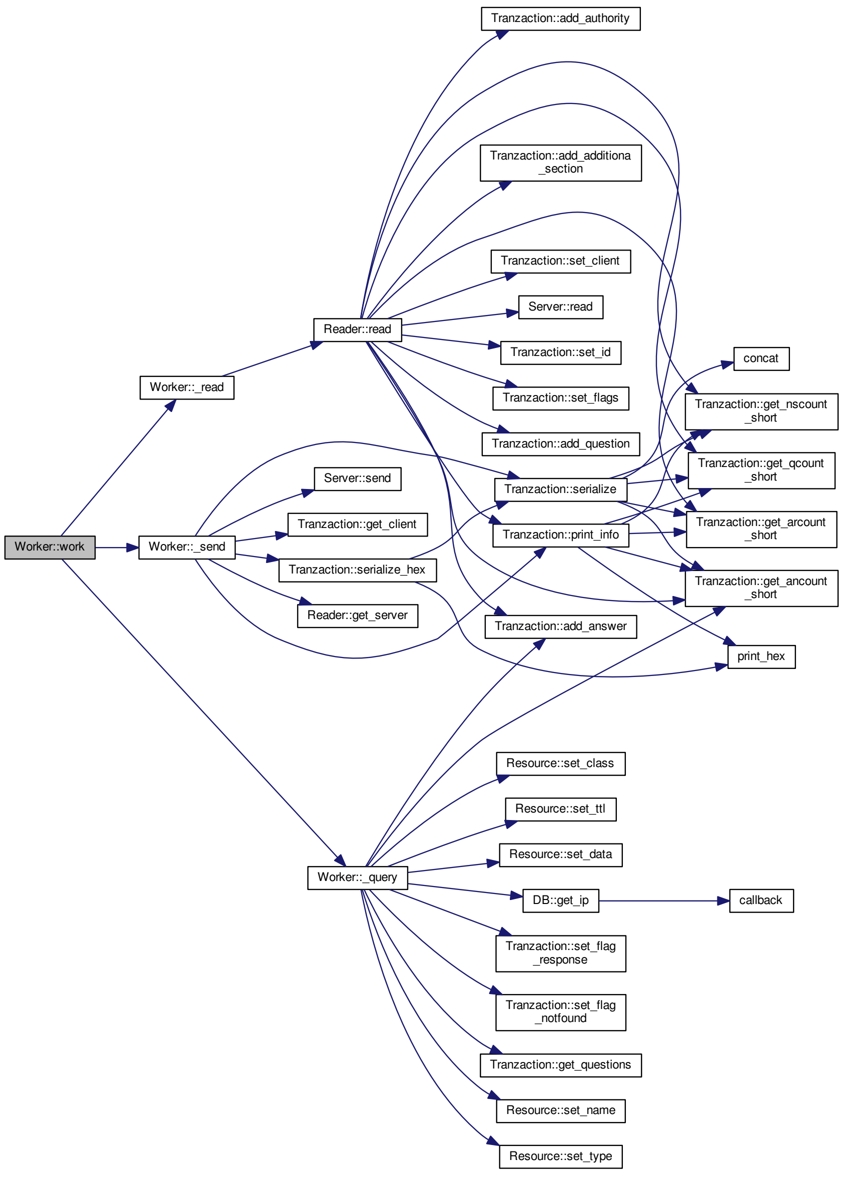
5.38.2.4 void Worker::sign_stop ( )

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

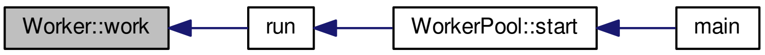
5.38.2.5 void Worker::work ( )

```
{
    /* Realizam un ciclu */
    std::cout << "[" << this->id << "] Work apel !" << std::endl;
    while (this->stop == false)
    {
        std::cout << "[" << this->id << "] Lock Work !" << std::endl;
        this->lock->lock();
        /* Citim o tranzactie */
        Tranzaction* tr = this->_read();

        /* Raspundem la tranzactie */
        this->_query(tr);

        /* Trimitem informatiile serializate */
        this->_send(tr);

        /* Eliberam memoria */
        delete tr;

        std::cout << "[" << this->id << "] Unlock Work !" << std::endl;
        this->lock->unlock();
    }
}
```

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

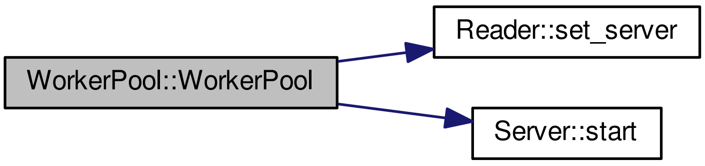
- WorkerPool (Parser ∗pr)
- void close ()
- void start ()

5.39.1 Constructor & Destructor Documentation

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

```
{
    /* Pregatim Pool-ul de workers
     *
     * @param pr
     * Un obiect de tip parser
     */
    std::cout << "Pornim WorkerPool" << std::endl;
    this->pr = pr;

    this->server = new Server((*this->pr)["port"]->get_int(),
                              (*this->pr)["backlog"]->get_int());
    this->rd = new Reader();
    this->rd->set_server(this->server);

    this->server->start();


    std::cout << "Generam workes  (" <<
                 (*this->pr)["min_threads"]->get_int() <<
                 "):" << std::endl;
    for (int i = 0; i<(*this->pr)["min_threads"]->get_int(); ++i)
    {
        /* Generam numarul de workers */
        std::cout <<" Suntem la " << i << std::endl;
        this->workers.push_back(
            new Worker((char*)(*this->pr)["db_name"]->get_string().c_str(),
                &(this->lock), this->rd, this->pr, i));
    }
}
```

Here is the call graph for this function:

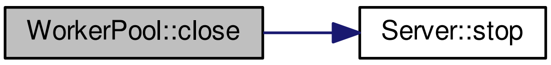


5.39.2 Member Function Documentation

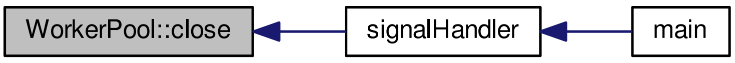
5.39.2.1 void WorkerPool::close ( )

```
{
    std::cout << "Inchide " << std::endl;
    this->finish = true;
    /* Trimitem semnalul de oprire */
    for (std::vector<Worker*>::iterator it = this->workers.begin();
        it < this->workers.end(); ++it)
    {
        (*it)->sign_stop();
    }

    /* Asteptam dupa fiecare thread */
    for (std::vector<std::thread*>::iterator it = this->threads.begin();
        it < this->threads.end(); ++it)
    {
        (*it)->join();
    }


    /* Inchidem servarul */
    this->lock.lock();
    this->server->stop();
    this->lock.unlock();
}
```

Here is the call graph for this function:



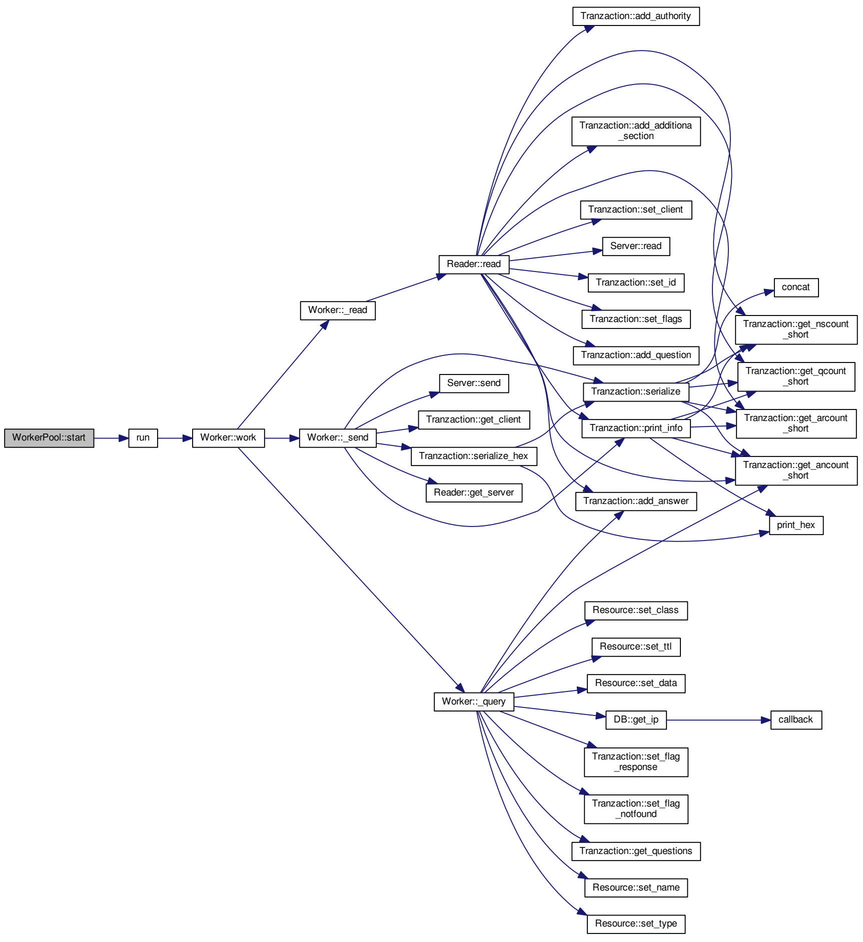
Here is the caller graph for this function:



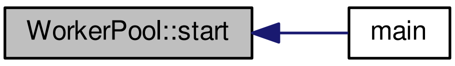
5.39.2.2 void WorkerPool::start ( )

```
{
    /* Pornim Pool-ul */
    std::cout << "Generam threads :" << std::endl;
    /* Spawnam threadurile */
    for (std::vector<Worker*>::iterator it = this->workers.begin();
        it < this->workers.end(); ++it)
    {
        this->threads.push_back(
            new std::thread(run, (*it)));
    }

    this->finish = false;

    std::cout << "Start !" << std::endl;
    while (this->finish == false)
    {
        /* Astepata */
    }
}
```

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

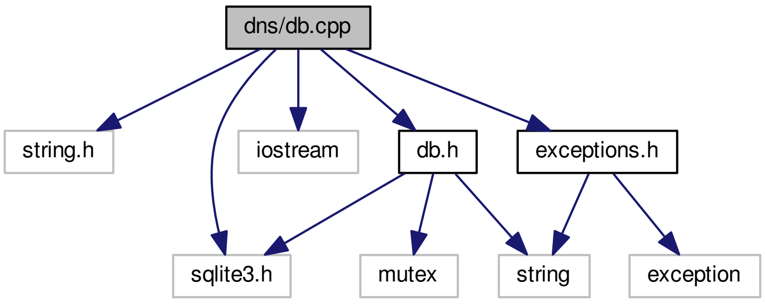
Chapter 6

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

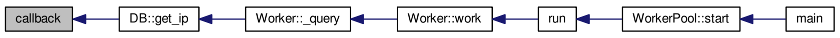
- char TABLE_NAME [] = "my_dns"
- char DOMAIN [] = "domain"
- char IP [] = "ip"

6.1.1 Function Documentation

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

```
{
    /* Functia de callback pentru SQLite3
     *
     * @param[in, out] data
     *  Parametru setat de noi
     *
     * @param[in] rownum
     *  Numarul de coloane
     *
     * @param[in] field_data
     *  Array cu informatiile din coloane
     *
     * @param[in] field_name
     *  Array cu numele coloanelor
     */
    if (colnum == 1)
    {
        /* Verificam existenta */
        if (atoi(field_data[0]) == 0)
        {
            memset(data, '0', DB::IP_MAX_SIZE);
        }
        else
        {
            memset(data, '1', DB::IP_MAX_SIZE);
        }
    }
    else
    {
        /* Returnam un IP */
        if (colnum != 2 || (strcmp(field_name[0], DOMAIN) != 0) ||
            strcmp(field_name[1], IP) != 0)
        {
            /* Nu avem numarul dorit de coloane */
            return 1;
        }

        strcpy((char*)data, field_data[1]);
    }
    return 0;
}
```

Here is the caller graph for this function:



6.1.2 Variable Documentation

6.1.2.1 char DOMAIN[ ] = "domain"

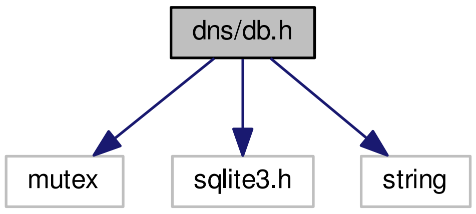
6.1.2.2 char IP[ ] = "ip"

6.1.2.3 char TABLE_NAME[ ] = "my_dns"

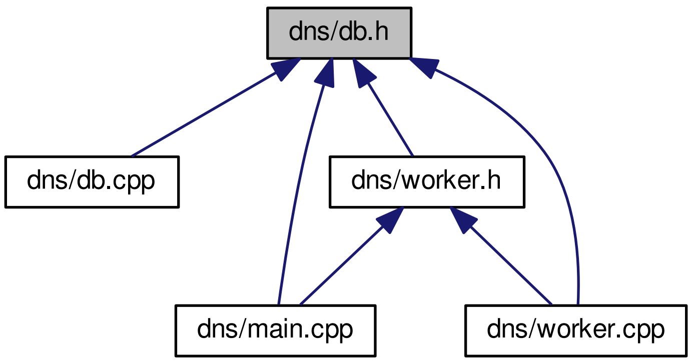
6.2 dns/db.h File Reference

```
#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

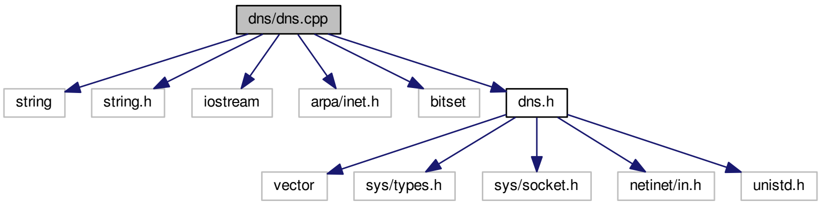
6.2.1.1 #define DB_H value

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

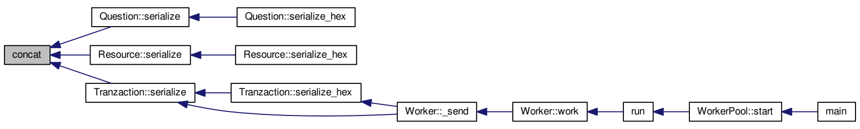
- void print_hex (char ∗ch, int len, bool endl)
- void concat (char ∗∗data, unsigned short &len, char ∗part1, unsigned short len_1, char ∗part2, unsigned short len_2)
- void print_char (char ∗ch, int len, bool endl)
- void print_int (char ∗ch, int len, bool endl)

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*)

```
{
    /* Concataneaza 2 char* in unul nou */
    char *data_aux= new char[len_1 + len_2];
    memcpy(data_aux, part1, len_1);
    memcpy((data_aux+len_1), part2, len_2);

    *data = data_aux;
    len = len_1 + len_2;
}
```

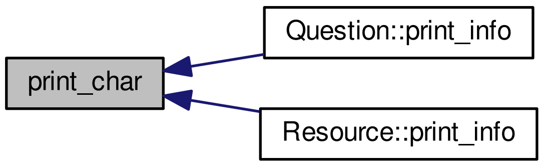
Here is the caller graph for this function:



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

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

    if (endl == true)
    {
        std::cout << std::endl;
    }
}
```

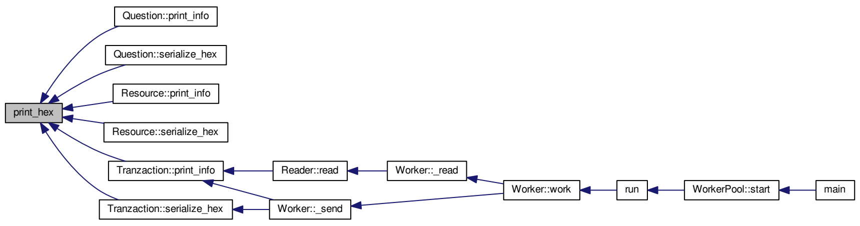
Here is the caller graph for this function:



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

```
{
    /* Afiseaxa caracter cu caracter in hexa */
    for (int i = 0; i < len; ++i)
    {
        std::cout << std::hex << (int)ch[i] << " ";
    }

    if (endl == true)
    {
        std::cout << std::endl;
    }
}
```

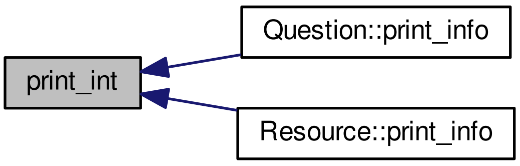
Here is the caller graph for this function:



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

```
{
    /* Afiseaxa caracter cu caracter in int*/
    for (int i = 0; i < len; ++i)
    {
        std::cout << (int)ch[i] << " ";
    }

    if (endl == true)
    {
        std::cout << std::endl;
        }
    }
```

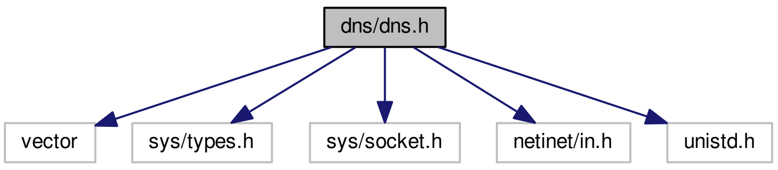
Here is the caller graph for this function:



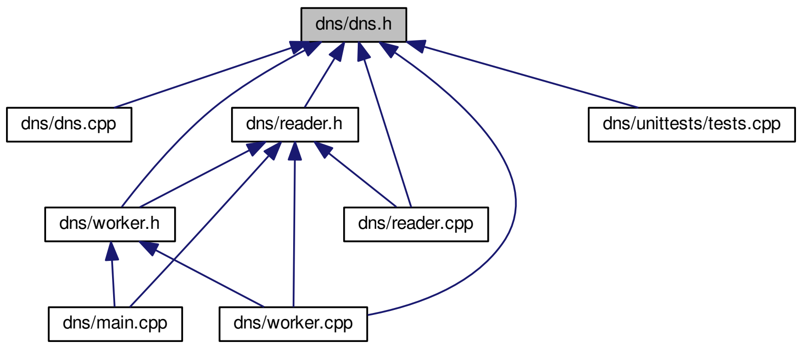
6.4 dns/dns.h File Reference

```
#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

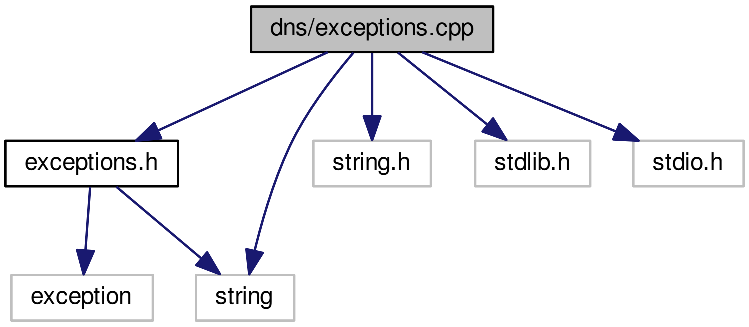
6.4.1 Macro Definition Documentation

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

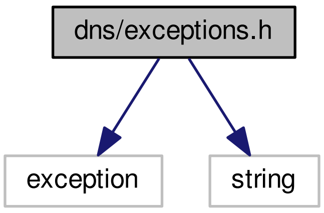
6.5.1.1 const char∗ what ( ) throw ( )

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

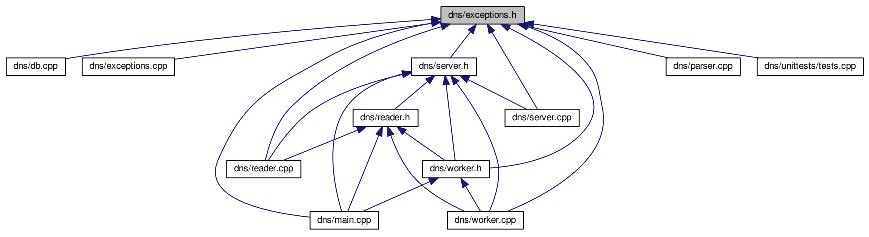
6.6 dns/exceptions.h File Reference

```
#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

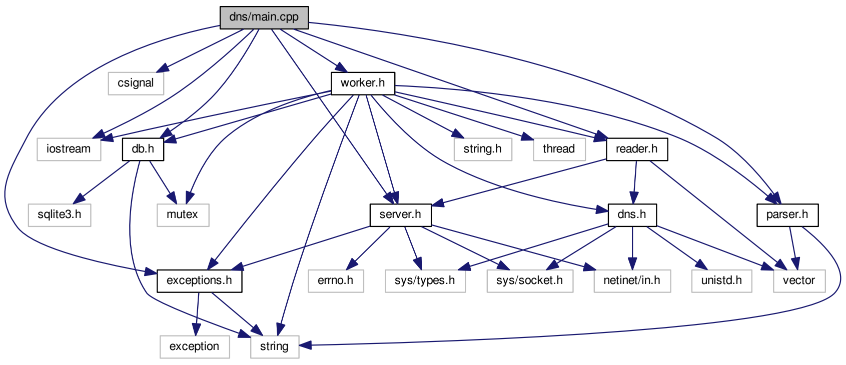
6.6.1 Macro Definition Documentation

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

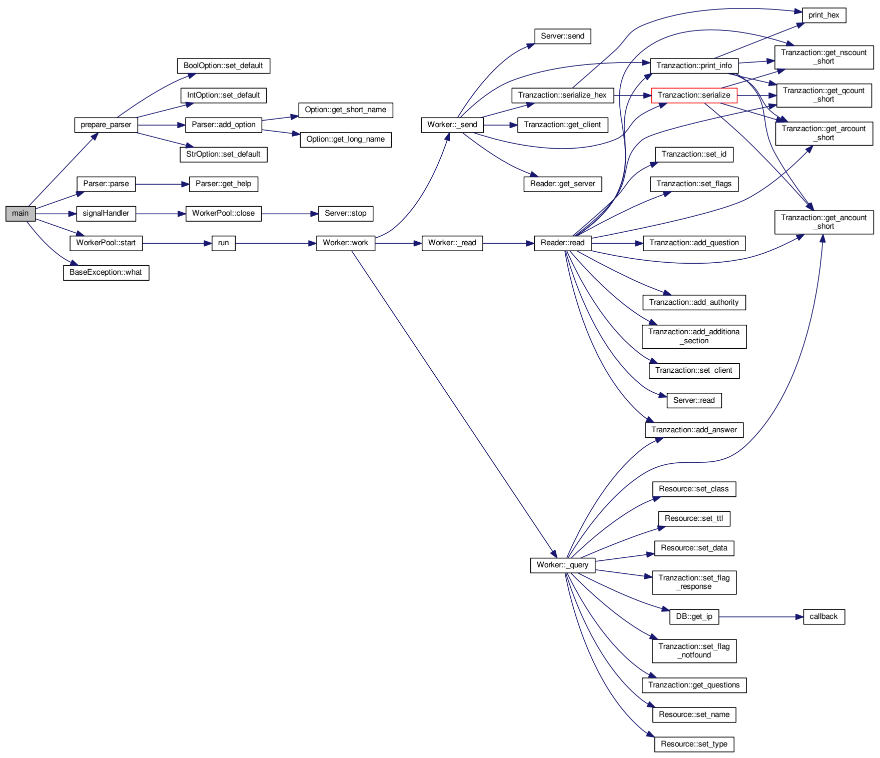
- WorkerPool ∗ MAIN = NULL

6.7.1 Function Documentation

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

```
{
    try
    {
        Parser p = prepare_parser();
        p.parse(argc, argv);

        if (p["soft"]->get_bool() == true)
        {
            /* Inregistreaza handler pentru CTRL+C */
            signal(SIGINT, signalHandler);

            /* Inregistreaza handler pentru pentru gracefully shutdown */
            signal(SIGTERM, signalHandler);
        }


        WorkerPool pool(&p);
        MAIN = &pool;
        pool.start();

    }
    catch (BaseException& ex)
    {
        std::cout << "O exceptie nu a fost prinsa !" << std::endl;
        ex.what();
    }

    return 0;
}
```

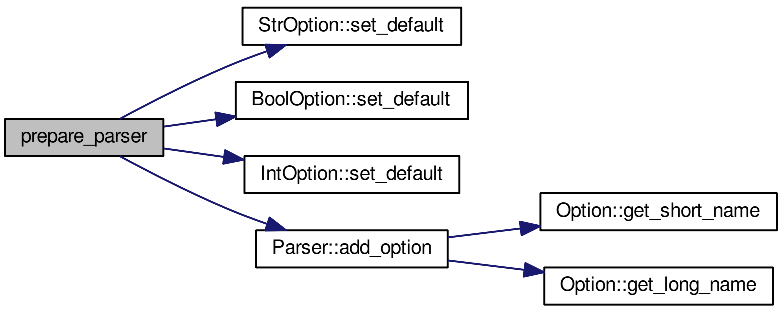
Here is the call graph for this function:



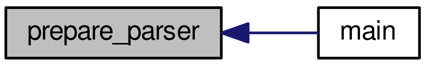
6.7.1.2 Parser prepare_parser ( )

```
{
    /* Pregateste parserul cu toate valorile */
    Parser parser;
    StrOption* verbosity = new StrOption('v', "verbose", "Verbosity level(v, vv, vvv).",
   false);
    verbosity->set_default("v");

    BoolOption* debug = new BoolOption('d', "debug", "Debug mode(default false).",
   false);
    debug->set_default(false);

    IntOption* port = new IntOption('p', "port", "The port.", false);

    IntOption* min_threads = new IntOption(
        't', "min_threads", "The min. number of threads, or the base threads.", true);

    IntOption* backlog = new IntOption(
        'b', "backlog", "The size of the listen queue ( defaut 10).", false);
    backlog->set_default(10);


    StrOption* db_name = new StrOption('f', "db_name", "Fisierul cu baza de date", true);
    BoolOption* soft = new BoolOption(
        's', "soft", "Daca dorim sa asteptam fiecare thread inainte de terminare.", true);

    parser.add_option(verbosity);
    parser.add_option(debug);
    parser.add_option(port);
    parser.add_option(min_threads);
    parser.add_option(db_name);
    parser.add_option(soft);
    parser.add_option(backlog);
    return parser;
}
```

Here is the call graph for this function:



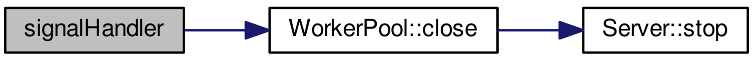
Here is the caller graph for this function:



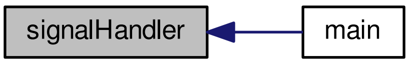
6.7.1.3 void signalHandler (int *signum*)

```
{
    std::cout << "Interrupt signal (" << signum << ") received." << std::endl;

    if (MAIN == NULL)
    {
        std::cout << "Inca nu s-a pornit servarul " << std::endl;
    }
    else
    {
        MAIN->close();
    }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



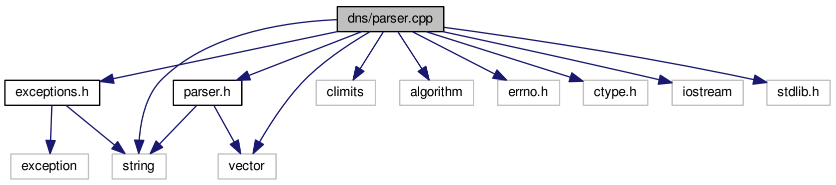
6.7.2 Variable Documentation

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

```
{
    S2I_SUCCESS,
    S2I_OVERFLOW,
    S2I_UNDERFLOW,
    S2I_INCONVERTIBLE
};
```

6.8.2 Function Documentation

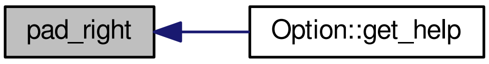
6.8.2.1 std::string pad_left (std::string const & *str*, size_t *s*)

```
{
    /* Adauga padding la stanga un string pana la o dimensiune anume
     *
     * @param[in] str
     *   Stringul caruia vrem sa ii adaugam padding
     *
     * @param[in] s
     *   Cat pading dorim sa ii adaugam
     */
    if ( str.length() < s )
    {
        return std::string(s-str.length(), ' ') + str;
    }
    else
    {
        return str;
    }
}
```

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

```
{
    /* Adauga padding la dreapta un string pana la o dimensiune anume
     *
     * @param[in] str
     *   Stringul caruia vrem sa ii adaugam padding
     *
     * @param[in] s
     *   Cat pading dorim sa ii adaugam
     */
    if (str.length() < s)
    {
        return str + std::string(s-str.length(), ' ');
    }
    else
    {
        return str;
    }
}
```

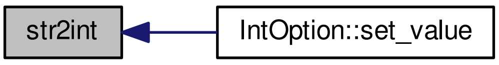
Here is the caller graph for this function:



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

```
{
    char *end;
    long  val;
    errno = 0;
    val = strtol(s, &end, base);
    if ((errno == ERANGE && val == LONG_MAX) || val > INT_MAX)
    {
        return S2I_OVERFLOW;
    }
    if ((errno == ERANGE && val == LONG_MIN) || val < INT_MIN)
    {
        return S2I_UNDERFLOW;
    }
    if (*s == '\0' || *end != '\0')
    {
        return S2I_INCONVERTIBLE;
    }
    i = val;
    return S2I_SUCCESS;
}
```

Here is the caller graph for this function:



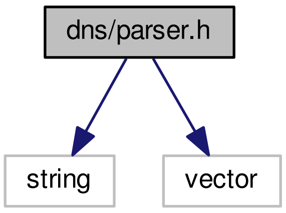
6.8.3 Variable Documentation

6.8.3.1 int errno

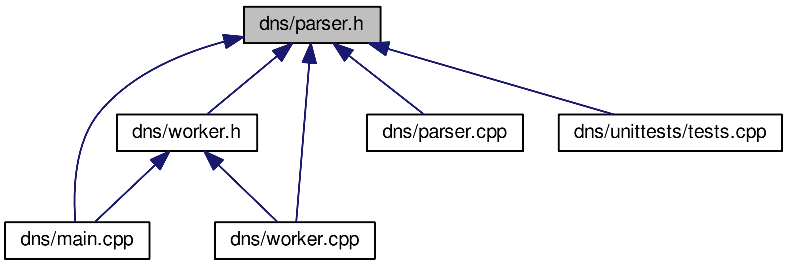
6.9 dns/parser.h File Reference

```
#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

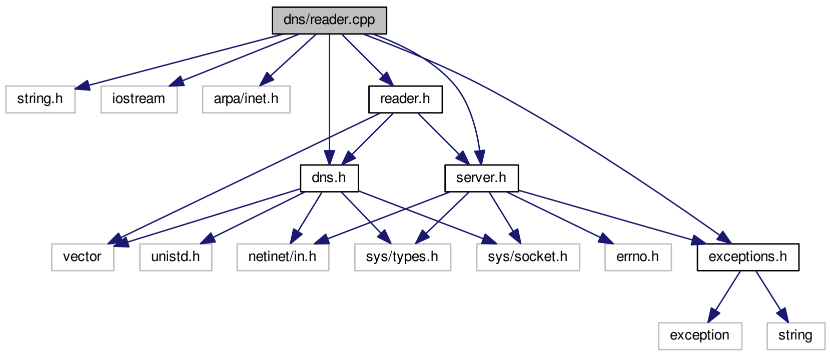
6.9.1.1 #define PARTSE_H value

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"
```

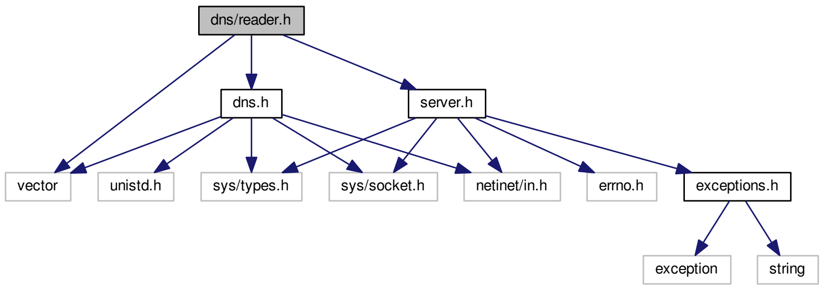
Include dependency graph for reader.cpp:



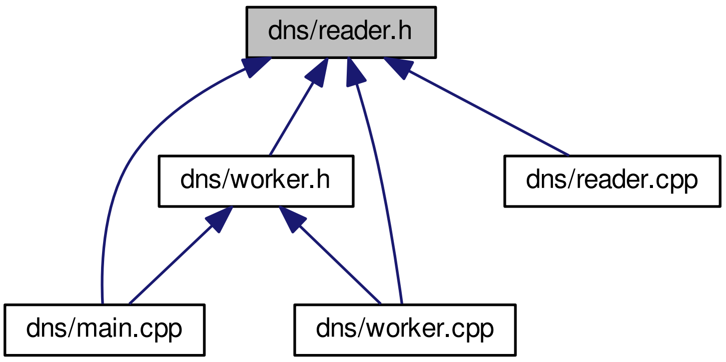
6.11 dns/reader.h File Reference

```
#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

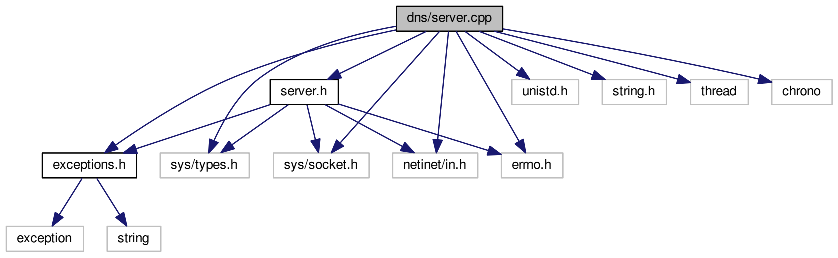
6.11.1 Macro Definition Documentation

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

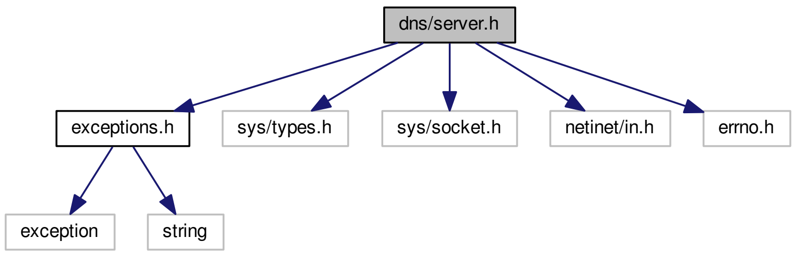
6.12.1 Variable Documentation

6.12.1.1 int errno

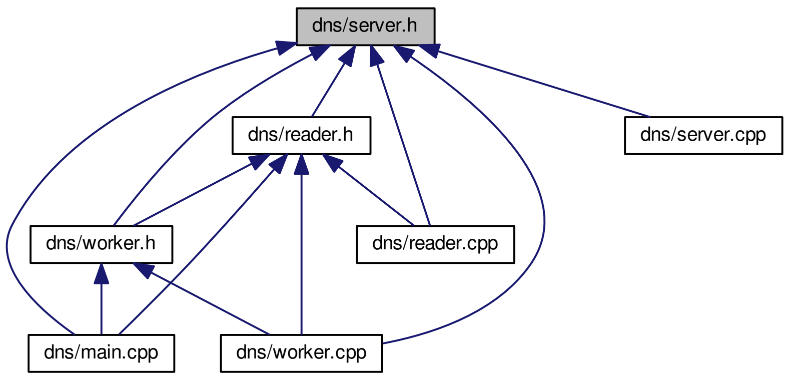
6.13 dns/server.h File Reference

```
#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

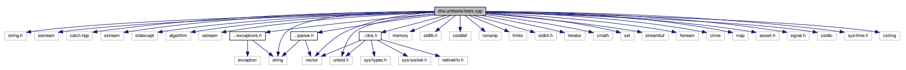
6.13.1 Macro Definition Documentation

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

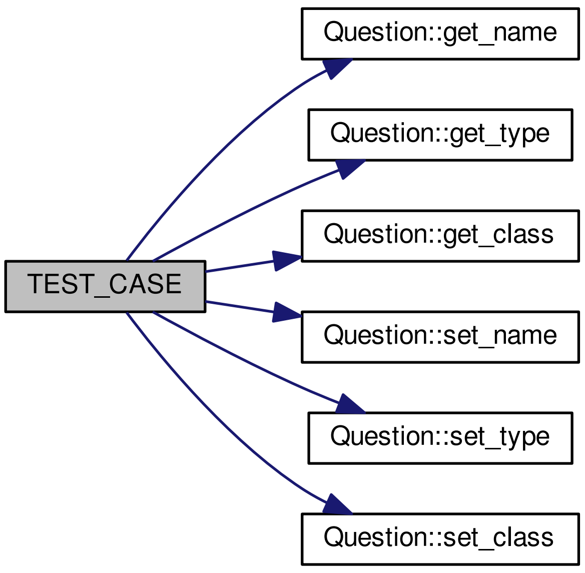
6.14.1.1 #define CATCH_CONFIG_MAIN

6.14.2 Function Documentation

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

```
{
    Question q;
    unsigned short len = 2;
    int cmp = -100;

    SECTION("Test default constructor - name")
    {
        char* name;
        name = (char*)1; /* Trebuie sa fie diferit de NULL */
        unsigned short length;

        q.get_name(&name, length);
        REQUIRE(length == 0);
        REQUIRE(name == NULL);
    }

    SECTION("Test default constructor - type")
    {
        char type[len];
        memset(type, 'f',  sizeof(type));
        q.get_type(type);
        cmp = strncmp(type, "\0\0", len);

        REQUIRE(cmp == 0);
    }

    SECTION("Test default constructor - class")
    {
        char cls[len];
        memset(cls, 'f',  sizeof(cls));
        q.get_class(cls);
        cmp = strncmp(cls, "\0\0", len);

        REQUIRE(cmp == 0);
    }

    SECTION("Test setter - name")
    {

        for (unsigned short i = 1; i <= 10; i++)
        {
            unsigned short aux_len = 0;
            char name_set[i];
            char* name_get = NULL;
            memset(name_set, 'f',  i);

            q.set_name(name_set, i);
            q.get_name(&name_get, aux_len);

            cmp = strncmp(name_set, name_get, aux_len);

            REQUIRE(i == aux_len);
            REQUIRE(cmp == 0);
        }
    }

    SECTION("Test setter - type")
    {
        char type_set[len];
        char type_get[len];
        memset(type_set, 'f',  sizeof(type_set));
        memset(type_set, 'x',  sizeof(type_get));
        q.set_type(type_set);
        q.get_type(type_get);

        cmp = strncmp(type_set, type_get, len);

        REQUIRE(cmp == 0);
    }

    SECTION("Test setter - class")
    {
        char cls_set[len];
        char cls_get[len];
        memset(cls_set, 'f',  sizeof(cls_set));
        memset(cls_set, 'x',  sizeof(cls_get));
        q.set_class(cls_set);
        q.get_class(cls_get);

        cmp = strncmp(cls_set, cls_get, len);

        REQUIRE(cmp == 0);
    }

}
```

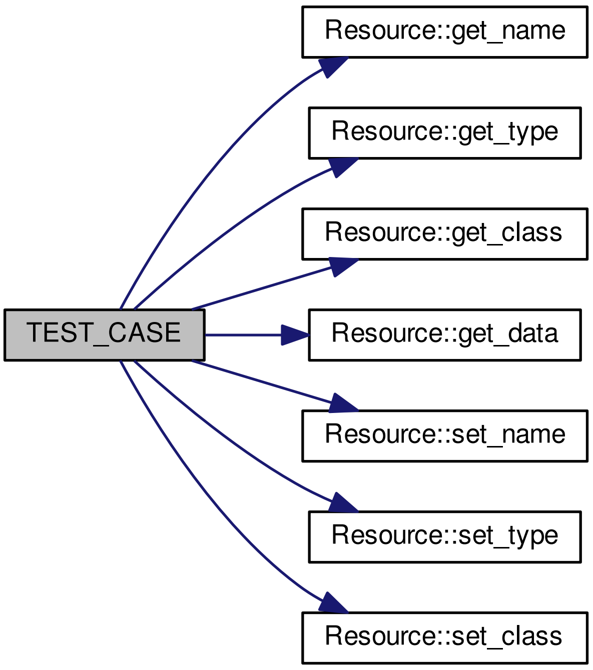
Here is the call graph for this function:



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

```
{
    Resource res;
    unsigned short len = 2;
    int cmp = -100;

    SECTION("Test default constructor - name")
    {
        char* name;
        name = (char*)1; /* Trebuie sa fie diferit de NULL */
        unsigned short length;

        res.get_name(&name, length);
        REQUIRE(length == 0);
        REQUIRE(name == NULL);
    }

    SECTION("Test default constructor - type")
    {
        char type[len];
        memset(type, 'f',  sizeof(type));
        res.get_type(type);
        cmp = strncmp(type, "\0\0", len);

        REQUIRE(cmp == 0);
    }

    SECTION("Test default constructor - class")
    {
        char cls[len];
        memset(cls, 'f',  sizeof(cls));
        res.get_class(cls);
        cmp = strncmp(cls, "\0\0", len);

        REQUIRE(cmp == 0);
    }

    SECTION("Test default constructor - data")
    {
        char *data = (char*)1;
        unsigned short len = 1;

        res.get_data(&data, len);
        REQUIRE(len == 0);
        REQUIRE(data == NULL);
    }

    SECTION("Test setter - name")
    {

        for (unsigned short i = 1; i <= 10; i++)
        {
            unsigned short aux_len = 0;
            char name_set[i];
            char* name_get = NULL;
            memset(name_set, 'f',  i);

            res.set_name(name_set, i);
            res.get_name(&name_get, aux_len);

            cmp = strncmp(name_set, name_get, aux_len);

            REQUIRE(i == aux_len);
            REQUIRE(cmp == 0);
        }
    }

    SECTION("Test setter - type")
    {
        char type_set[len];
        char type_get[len];
        memset(type_set, 'f',  sizeof(type_set));
        memset(type_set, 'x',  sizeof(type_get));
        res.set_type(type_set);
        res.get_type(type_get);

        cmp = strncmp(type_set, type_get, len);

        REQUIRE(cmp == 0);
    }

    SECTION("Test setter - class")
    {
        char cls_set[len];
        char cls_get[len];
        memset(cls_set, 'f',  sizeof(cls_set));
        memset(cls_set, 'x',  sizeof(cls_get));
        res.set_class(cls_set);
        res.get_class(cls_get);

        cmp = strncmp(cls_set, cls_get, len);

        REQUIRE(cmp == 0);
    }

    SECTION("Test setter - data")

    {

        for (unsigned short i = 1; i <= 10; i++)
        {
            unsigned short aux_len = 0;
            char data_set[i];
            char* data_get = NULL;
            memset(data_set, 'f',  i);

            res.set_name(data_set, i);
            res.get_name(&data_get, aux_len);

            cmp = strncmp(data_set, data_get, aux_len);

            REQUIRE(i == aux_len);
            REQUIRE(cmp == 0);
        }
    }

}
```

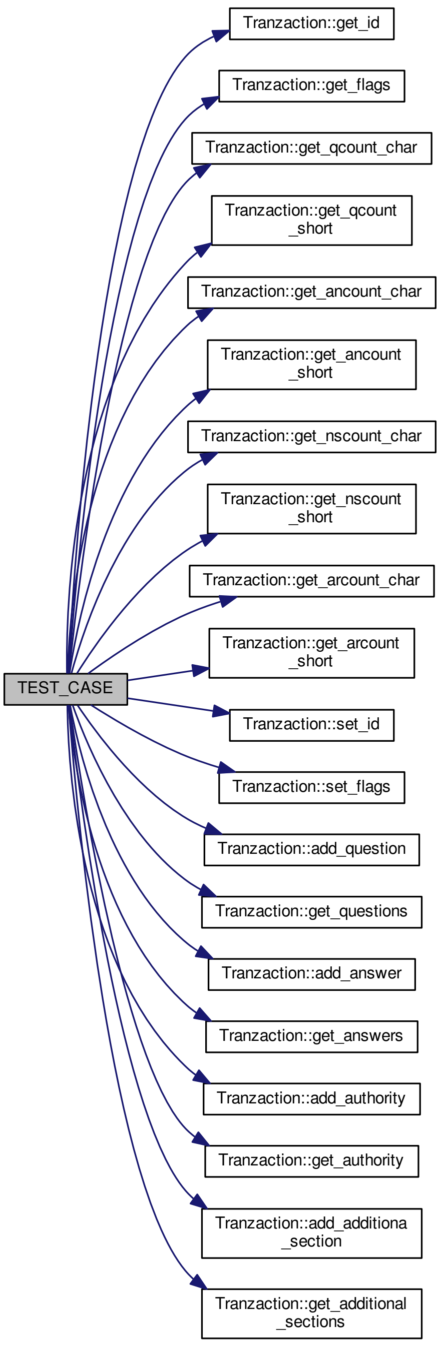
Here is the call graph for this function:



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

```
{
    Tranzaction tr;
    unsigned short len = 2;
    int cmp = -100;

    SECTION("Test default constructor - id")
    {
        char id[len];
        memset(id, 'f',  sizeof(id));
        tr.get_id(id);
        cmp = strncmp(id, "\0\0", len);

        REQUIRE(cmp == 0);
    }

    SECTION("Test default constructor - flags")
    {
        char flags[len];
        memset(flags, 'f',  sizeof(flags));
        tr.get_flags(flags);
        cmp = strncmp(flags, "\0\0", len);

        REQUIRE(cmp == 0);
    }

    SECTION("Test default constructor - qcount")
    {
        char qcount[len];
        memset(qcount, 'f', len);

        unsigned short rez;
        tr.get_qcount_char(qcount);
        rez = tr.get_qcount_short();

        cmp = strncmp(qcount, "\0\0", len);
        REQUIRE(cmp == 0);
        REQUIRE(rez == 0);
    }

    SECTION("Test default constructor - ancount")
    {
        char ancount[len];
        memset(ancount, 'f', len);

        unsigned short rez;
        tr.get_ancount_char(ancount);
        rez = tr.get_ancount_short();

        cmp = strncmp(ancount, "\0\0", len);
        REQUIRE(cmp == 0);
        REQUIRE(rez == 0);
    }

    SECTION("Test default constructor - nscount")
    {
        char nscount[len];
        memset(nscount, 'f', len);

        unsigned short rez;
        tr.get_nscount_char(nscount);
        rez = tr.get_nscount_short();

        cmp = strncmp(nscount, "\0\0", len);
        REQUIRE(cmp == 0);
        REQUIRE(rez == 0);
    }

    SECTION("Test default constructor - arcount")
    {
        char arcount[len];
        memset(arcount, 'f', len);

        unsigned short rez;
        tr.get_arcount_char(arcount);
        rez = tr.get_arcount_short();

        cmp = strncmp(arcount, "\0\0", len);
        REQUIRE(cmp == 0);
        REQUIRE(rez == 0);
    }

    SECTION("Test setter - id")
    {
        char id_set[2],
             id_get[2];

        for (char i ='a'; i <= 'c'; i++)
        {
            id_set[0] = i;
            id_set[1] = i+1;

            tr.set_id(id_set);
            tr.get_id(id_get);

            cmp = strncmp(id_set, id_get, 2);
            REQUIRE(cmp == 0);
        }
    }

    SECTION("Test setter - flags")
    {
        char flags_set[2],
             flags_get[2];

        for (char i ='a'; i <= 'c'; i++)
        {
            flags_set[0] = i;
            flags_set[1] = i+1;

            tr.set_flags(flags_set);
            tr.get_flags(flags_get);
            cmp = strncmp(flags_set, flags_get, 2);
            REQUIRE(cmp == 0);
        }
    }


    SECTION("Test setter - questions")
    {
        Question q1, q2;

        REQUIRE(tr.get_qcount_short() == 0);

        /* One question */
        tr.add_question(q1);
        std::vector<Question> questions;
        questions = tr.get_questions();

        REQUIRE(tr.get_qcount_short() == 1);
        REQUIRE(questions.size() == 1);

        /* Two question */
        tr.add_question(q2);
        questions = tr.get_questions();

        REQUIRE(tr.get_qcount_short() == 2);
        REQUIRE(questions.size() == 2);

    }

    SECTION("Test setter - answers")
    {
        Resource r1, r2;

        REQUIRE(tr.get_ancount_short() == 0);

        /* One resources */
        tr.add_answer(r1);
        std::vector<Resource> resource;
        resource = tr.get_answers();

        REQUIRE(tr.get_ancount_short() == 1);
        REQUIRE(resource.size() == 1);

        /* Two resouces */
        tr.add_answer(r2);
        resource = tr.get_answers();

        REQUIRE(tr.get_ancount_short() == 2);
        REQUIRE(resource.size() == 2);
    }

    SECTION("Test setter - authority")
    {
        Resource r1, r2;

        REQUIRE(tr.get_nscount_short() == 0);

        /* One resources */
        tr.add_authority(r1);
        std::vector<Resource> resource;
        resource = tr.get_authority();

        REQUIRE(tr.get_nscount_short() == 1);
        REQUIRE(resource.size() == 1);

        /* Two resouces */
        tr.add_authority(r2);
        resource = tr.get_authority();

        REQUIRE(tr.get_nscount_short() == 2);
        REQUIRE(resource.size() == 2);
    }

    SECTION("Test setter - additional_sections")
    {
        Resource r1, r2;

        REQUIRE(tr.get_arcount_short() == 0);

        /* One resources */
        tr.add_additiona_section(r1);
        std::vector<Resource> resource;
        resource = tr.get_additional_sections();

        REQUIRE(tr.get_arcount_short() == 1);
        REQUIRE(resource.size() == 1);

        /* Two resouces */
        tr.add_additiona_section(r2);
        resource = tr.get_additional_sections();

        REQUIRE(tr.get_arcount_short() == 2);
        REQUIRE(resource.size() == 2);
    }

}
```

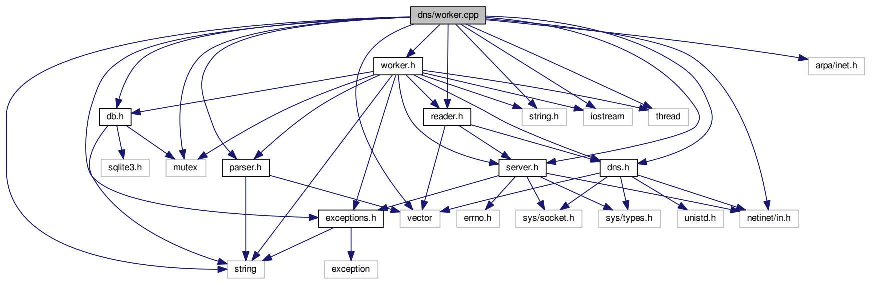
Here is the call graph for this function:


Tranzaction::get_id
Tranzaction::get_flags
Tranzaction::get_qcount_char
Tranzaction::get_qcount _short
Tranzaction::get_ancount_char
Tranzaction::get_ancount _short
Tranzaction::get_nscount_char
Tranzaction::get_nscount _short
Tranzaction::get_arcount_char
Tranzaction::get_arcount _short
TEST_CASE
Tranzaction::set_id
Tranzaction::set_flags
Tranzaction::add_question
Tranzaction::get_questions
Tranzaction::add_answer
Tranzaction::get_answers
Tranzaction::add_authority
Tranzaction::get_authority
Tranzaction::add_additiona _section
Tranzaction::get_additional _sections

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

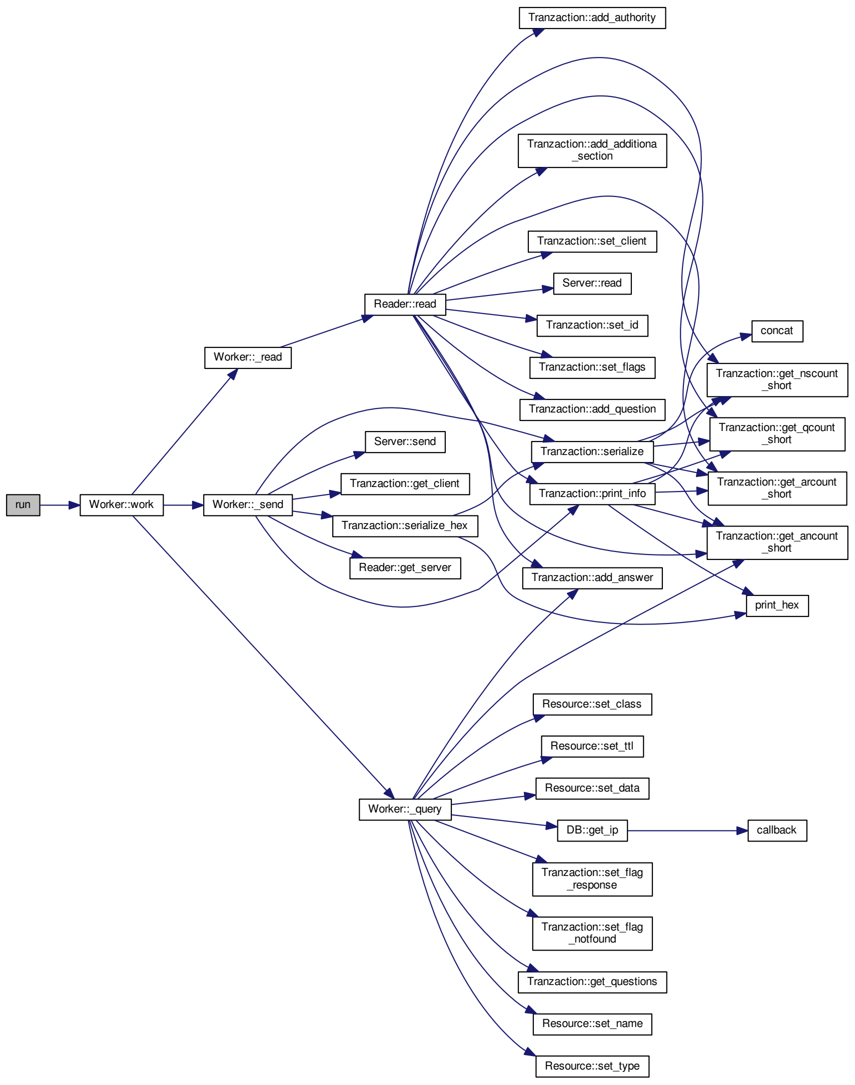
- void run (Worker ∗w)

6.15.1 Function Documentation

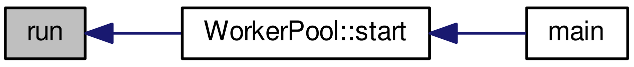
6.15.1.1 void run (Worker ∗ *w*)

```
{
    /* Pornim workerul
     *
     * @param w
     *  Workerul pe care dorim sa il pornim
     */
    std::cout << "Apelam work !" << std::endl;
    w->work();
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

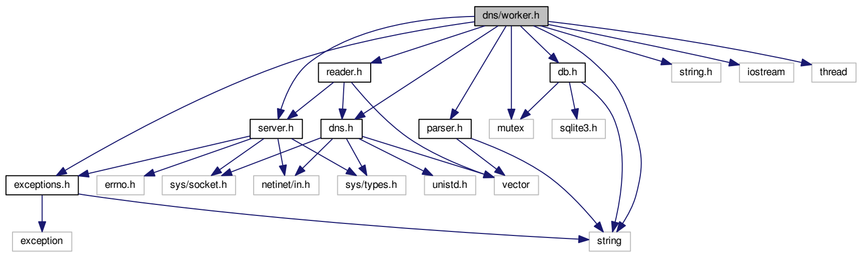


6.16 dns/worker.h File Reference

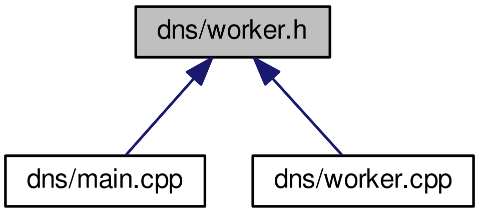
```
#include <string>
```

```
#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