

УКРАИНСКИЙ
АНТИВИРУСНЫЙ
ЦЕНТР



Интерфейс подключение антивирусного ядра “Ukrainian National Antivirus”

ВНИМАНИЕ:

Вся информация в этом документе распространяется «как есть». Разработчики не гарантируют отсутствия ошибок в описании и реализации API. Данное API разработано для подключения антивирусного ядра другими приложениями.

Оглавление

Интерфейс подключение антивирусного ядра "Ukrainian National Antivirus"	1
Введение	3
Подключение антивирусного ядра UNA (Windows)	4
Особенности работы API	6
Пример подключения на C++	7
Пример подключения на Delphi	9

Введение

В данном документе содержится информация по методам подключения антивирусного ядра «Украинского Национального Антивируса» другими программами. Для использования антивирусного ядра САЗ UNA вы должны быть зарегистрированным пользователем и иметь хотя бы одну лицензию на использование антивирусного комплекса.

По вопросам приобретения антивирусной программы UNA обращайтесь в офис компании «Украинский Антивирусный Центр» или к нашим партнерам:

Internet: <http://www.unasoft.com.ua/>

e-mail: support@unasoft.com.ua

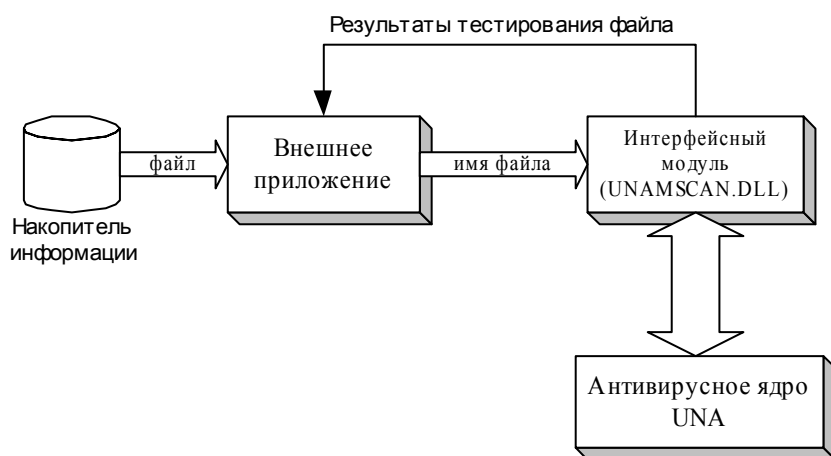
г.Киев 04080, ул.Терехина, 4 офис 5.

т.(044) 468-6650, 468-30-49

Подключение антивирусного ядра UNA (Windows)

Антивирусное ядро поставляется в набора динамических библиотек, которые предоставляют функции сканирования и лечения файлов (если лечение возможно). Входной информацией для антивирусного ядра при использовании данного API является файл. Если вам необходимо протестировать другой объект (сектор диска, область памяти и пр.) вам необходимо сохранить необходимые данные во временный файл и передать их антивирусному ядру на проверку.

Структура взаимодействия антивирусного ядра с внешним ПО приведена ниже:



Таким образом внешние ПО взаимодействует не напрямую с ядром, а с интерфейсным модулем UNAMSCAN.DLL.

Для подключения этих функций необходимо:

1. Объявить структуру данных для возврата информации от функций

```
typedef struct Result_var{unsigned char Infect;
                        char Vname[255];
                        unsigned char Rezh;
                        } Result_var;
```

Infect – переменная используется как флаг о наличии вируса;

Vname - используется для передачи имени вируса, если он был найден;

Rezh - возвращает информацию о действии, которое можно произвести над файлом: лечить или удалять (файлы в архивах и почтовых базах лечить нельзя, так как антивирусное ядро не изменяет структуры архивов и почтовых сообщений).

2. Объявить тип функции. Это необходимо для вызова функций из ядра. Данная функция будет использоваться только для обнаружения вирусного кода в файле, но ни как не для лечения.

```
bool (_stdcall *bcTUnaAPIFileTest)
(char * f_name,
```

```

unsigned char UseEvrlist ,
unsigned char UnPack,
Result_var * APIRez);

```

```

// UnaAPIFileTest – имя нашей процедуры;
// FName – имя проверяемого файла;
// UseEvrlist - использовать ли эвристический анализатор;
// UnPack – распаковывать архивы и почтовые сообщения;
// APIRez - структура, в которую будет возвращён результат тестирования;

```

3. Загрузить библиотеку в память

```

Const char api_dll[] = "UNAMSCAN.DLL";

HINSTANCE hLib = LoadLibrary(api_dll);
If((unsigned)hLib<=HINSTANCE_ERROR)
{
    exit(1);
}

```

4. Получить адреса функций и проассоциировать их с нашей функцией bcTUnaFileAPITest

```

bcTUnaAPIFileTest=(bool (__stdcall *)
                    (char * ,
                     unsigned char,
                     unsigned char,
                     Result_var *))
GetProcAddress(hLib, "UnaAPIFileTest");
if ( bcTUnaAPIFileTest == NULL )
{
    FreeLibrary(hLib);          // Если не можем получить - выход
    printf("Ошибка загрузки Dll");
    exit(1);
}

```

5. Вызвать функцию bcTUnaAPITest

```

bcTUnaAPIFileTest(f_name,Evrlist,Pack,&res);

```

F_Name – массив элементов типа CHAR, который должен содержать имя файла для тестирования (массив должен иметь размер не менее 4096 байт);

Res – возвращаемая структура;

Особенности работы API

Если проверяемый файл является архивом или почтовым сообщением, то в возвращаемую структуру будет возвращено только имя одного вируса (первого вируса, который был обнаружен в данном архиве/почтовой базе).

Для работы антивирусного ядра необходимо иметь установленный антивирусный комплекс "UNA for Win32".

Пример подключения на C++

Данный пример разрабатывался и компилировался на Microsoft Visual C++ 6.0, поэтому мы не можем гарантировать его работы под другими компиляторами C++

```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

// Объявляется структура, которая будет возвращена
// вызывающей программе
typedef struct result_var{
    unsigned char Infect; // переменная используется как флаг о наличии
                          // вируса или подозрения
    char VName[255];      // используется для передачи имени вируса если
                          // он был найден
    unsigned char Rezh;   // возвращает режим обработки данного вируса:
                          // лечить или удалять
}Result_var;             // Имя структуры

// Объявляется функция UnaAPIFileTest со входными параметрами
// ( unsigned char FILE * FName,
//   byte UseEvrlist ,
//   byte UnPack,
//   Result_var * APIRez)
//
// ГДЕ :
//   UnaAPIFileTest - имя нашей процедуры
//   FName - имя проверяемого файла
//   UseEvrlist : используется ли эвристический анализатор
//   UnPack : используется ли распаковщик архивов(баз)
//   APIRez : структура, в которую будет возвращён результат тестирования

bool (_stdcall *bcTUnaAPIFileTest)
    (char * f_name,
     unsigned char UseEvrlist ,
     unsigned char UnPack,
     Result_var * APIRez);

void main(void)
{
    // переменная testDll - константа содержащая имя загружаемой DLL
    const char testDll[] = "C:\\Program Files\\UACenter\\UNAMSCAN.DLL";
    // переменная res типа Result_var, то есть выходная структура
    Result_var res;
    // Имя файла, который будет тестироваться
    char f_name[4095]={"C:\\Program Files\\UACenter\\test.exe"};

    HINSTANCE hLib = LoadLibrary(testDll); // Загружаем библиотеку в память
    if((unsigned)hLib<=HINSTANCE_ERROR)
    {
        exit(1); // Если не можем загрузить - выход
    }

    // Получим адрес функции и проассоциируем их с нашей
    // функцией bcTUnaAPIFileTest
    bcTUnaAPIFileTest=(bool (__stdcall *)
                          (char * ,
```



```

        unsigned char,
        unsigned char,
        Result_var *))
GetProcAddress(hLib, "UnaAPIFileTest");
if ( bctUnaAPIFileTest == NULL )
{
    FreeLibrary(hLib);          // Если не можем получить - выход
    printf("Ошибка загрузки Dll");
    exit(1);
}

//Вызвать функцию bctUnaFileAPITest без эвристики и распаковки
bctUnaAPIFileTest(f_name,0,0,&res);
printf("Infected - %c Name Virus - %s,
        Delete- %c \n",res2.Infect,res2.VName,res2.Rezh);
FreeLibrary(hLib); // Выгружаем из памяти DLL
}

```

Пример подключения на Delphi

Пример создавался и компилировался на Borland Delphi 5.0

```
type
  str128=String[128];
  str255=String[255];
  TVirTypes=array[1..10000] of boolean;

  UNAAPIRez=record
    Infect:byte;
    VName :array[1..255] of char;
    Rezh :byte;
  end;
  PUnaAPIRez=^UNAAPIRez;

TFName=array[1..4096] of char;
TUnaAPIFileTest=Function(FName      :TFName;
                          UseEvrlist:byte;
                          Unpack    :Byte;
                          APIRez    :PUnaAPIRez
                          ):boolean;stdcall;

var
  UNAFFileTest:TUNAApiFileTest;
  EngineHandle:integer;

implementation

procedure TForm1.Button1Click(Sender: TObject);
var
  file_Name:string;
  infect:byte;
  VName:string;
  rezh:byte;
  APIRez:UNAAPIRez;
  FName:TFName;
begin
  file_name:=FilenameEdit1.Text;
  if FileExists(file_name) then begin
    StrPCopy(@FName, file_name);
    UnaFileTest(FName, 0, 1, @APIRez);
  end;
  Label1.Caption:='';
  if APIRez.Infect=1 then begin
    VName:=StrPas(@APIRez.VName);
    Label1.Caption:=VName;
  end;
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
  EngineHandle:=LoadLibrary(PChar('C:\Program Files\UACenter\unamscan.dll'));
  if EngineHandle>=32 then begin
    @UnaFileTest:=GetProcAddress(EngineHandle, 'UnaAPIFileTest');
  end;
  Sleep(100);
end;
```