

Министерство образования и науки Российской Федерации
Санкт-Петербургский Политехнический Университет Петра Великого

—
Институт Кибербезопасности и Защиты Информации

ЛАБОРАТОРНАЯ РАБОТА № 2

«УДАЛЕННЫЙ ФАЙЛОВЫЙ МЕНЕДЖЕР С ИМПЕРСОНАЦИЕЙ КЛИЕНТА»

по дисциплине «Безопасность современных информационных технологий»

Выполнил
студент гр. 3651003/80002

<подпись>

Сошнев М.Д

Преподаватель

<подпись>

Иванов Д.В.

Санкт-Петербург
2020

Оглавление

Цель работы.....	3
Задача.....	4
Ход работы.....	5
Контрольные вопросы	8
Выводы	9
Приложение А – интерфейс хранимых процедур	10
Приложение Б – программа-клиент	12
Приложение С – программа-сервер.....	23
Програма-клиент.....	23

ЦЕЛЬ РАБОТЫ

Получить навыки работы с механизмом удаленного вызова процедур (RPC).

ЗАДАЧА

Написать программу-сервер и программу-клиент, работающие под Windows 7-10. Сервер должен предоставлять доступ локальным и удаленным клиентам к файлам в своей файловой системе.

Требования:

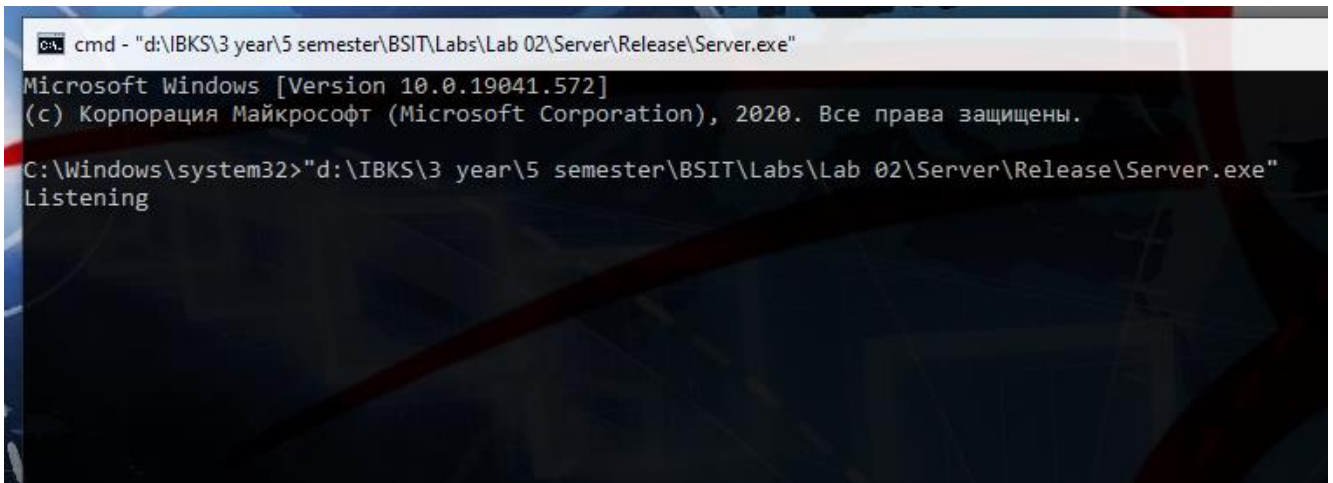
- Statefull сервер;
- Сервер не должен быть интерактивным (интерфейс командной строки);
- Взаимодействие с клиентами должно осуществляться с помощью механизма RPC;
- При обслуживании клиента должна осуществляться его имперсонация;
- Пользователю должны предоставляться следующие операции: копирование указанного файла с клиента на сервер, загрузка указанного файла с сервера на клиента, удаление указанного файла на сервере;
- Имя файла передается в формате UNC.
- В ходе работы должны быть проведены эксперименты с запуском сервера и клиентов под различными учетными записями с демонстрацией работы механизмов контроля доступа. Эксперименты должны показывать, что механизм имперсонации работает

ХОД РАБОТЫ

Сперва, в специальном файле .idl был реализован интерфейс для хранимых процедур сервера (Приложение А – интерфейс хранимых процедур). С помощью компилятора MIDL из этого файла были сгенерированы файлы с исходным кодом на языке СИ – заглушки, которые обращаясь к библиотеке rpcrt4.dll формируют пакеты и передают их от клиента на сервер. С помощью такого механизма, при разработке клиент-серверных приложений можно не задумываться о механизме передачи данных, просто вызывая функции сервера из клиента.

При запуске программы-сервера начинает прослушиваться один порт.

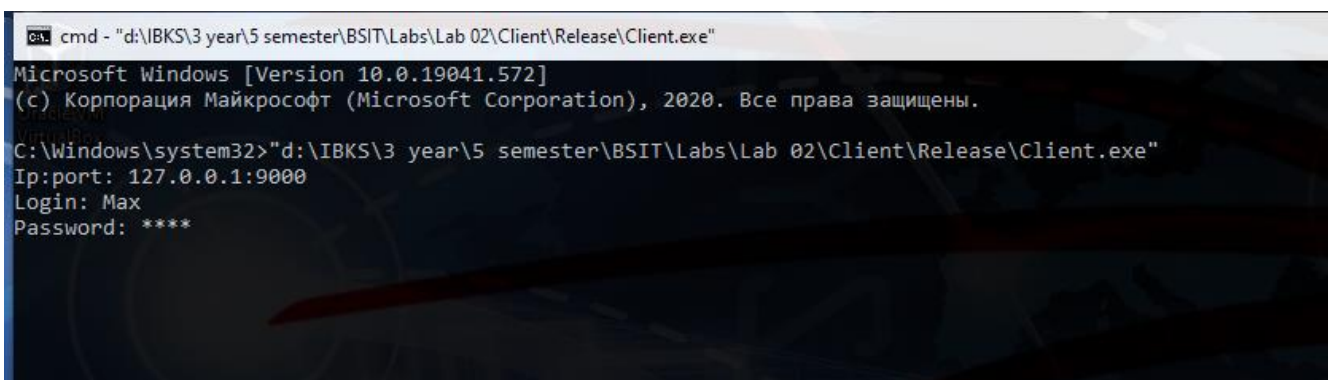
При запуске программы-клиента требуется ввести ipv4-адрес сервера, логин и пароль пользователя в системе windows сервера:



```
C:\> cmd - "d:\IBKS\3 year\5 semester\BSIT\Labs\Lab 02\Server\Release\Server.exe"
Microsoft Windows [Version 10.0.19041.572]
(c) Корпорация Майкрософт (Microsoft Corporation), 2020. Все права защищены.

C:\Windows\system32>"d:\IBKS\3 year\5 semester\BSIT\Labs\Lab 02\Server\Release\Server.exe"
listening
```

Рисунок 1 – запуск программы сервера



```
C:\> cmd - "d:\IBKS\3 year\5 semester\BSIT\Labs\Lab 02\Client\Release\Client.exe"
Microsoft Windows [Version 10.0.19041.572]
(c) Корпорация Майкрософт (Microsoft Corporation), 2020. Все права защищены.

C:\Windows\system32>"d:\IBKS\3 year\5 semester\BSIT\Labs\Lab 02\Client\Release\Client.exe"
Ip:port: 127.0.0.1:9000
Login: Max
Password: ****
```

Рисунок 2 – запуск программы-клиента

После ввода данных на сервере запускается первая хранимая процедура – login, проверяющая корректность введенных данных. В случае успеха, на программе-клиенте открывается псевдо-командная строка:

```
C:\Windows\system32>"d:\IBKS\3 year\5 semester\BSIT\Labs\Lab 02\Client\Release\Client.exe"
Ip:port: 127.0.0.1:9000
Login: Max
Password: 0707
Serv#
Serv#
Serv# |
```

Рисунок 3 – Успешная авторизация

На сервере, в то время, отображается подключение нового пользователя:

```
cmd - "d:\IBKS\3 year\5 semester\BSIT\Labs\Lab 02\Server\Release\Server.exe"
Microsoft Windows [Version 10.0.19041.572]
(c) Корпорация Майкрософт (Microsoft Corporation), 2020. Все права защищены.

C:\Windows\system32>"d:\IBKS\3 year\5 semester\BSIT\Labs\Lab 02\Server\Release\Server.exe"
Listening
New connection: Max|
```

Рисунок 4 – подключение нового пользователя

Имеется 3 команды на программе-клиенте:

```
C:\Windows\system32>"d:\IBKS\3 year\5 semester\BSIT\Labs\Lab 02\Client\Release\Client.exe"
Ip:port: 127.0.0.1:9000
Login: Max
Password: 0707
Serv#
Serv#
Serv# help
      rm [path] -- remove file
      down [path_from] [path_to] -- download file
      up [path_from] [path_to] -- upload file
      !! path without spaces!!
      exit
Serv#
```

Рисунок 5 – доступные клиенту команды

Каждая из них вызывает соответствующую хранимую процедуру на сервере.

На сервере реализованы следующие хранимые процедуры:

Таблица 1 – хранимые процедуры

Название	Описание
<pre>int serv_login(/* [string][in] */ const unsigned char *login, /* [string][in] */ const unsigned char *password);</pre>	<p>Попытка залогиниться в системе под логином LOGIN и паролем PASSWORD. В случае успеха возвращает 0, в случае неудачи – код</p>

	последней ошибки.
<pre>int serv_remove(/* [string][in] */ const unsigned char *path_file);</pre>	Удаление с сервера файла, лежащего по пути PATH_FILE. Возвращает 0 в случае успеха, -1 – в случае отсутствия файла, либо отсутствия прав доступа к файлу.
<pre>int serv_download(/* [string][in] */ const unsigned char *path_from, /* [size_is][out] */ unsigned char *buffer, /* [in] */ int buffer_size);</pre>	Попытка записать файл, лежащего по пути PATH_FROM в буфер BUFFER размером BUFFER_SIZE и передать буфер клиенту. Возвращает 0 в случае успеха, -1 – в случае отсутствия файла, либо отсутствия прав доступа к файлу.
<pre>int serv_upload(/* [string][in] */ const unsigned char *path_to, /* [size_is][in] */ const unsigned char *buffer, /* [in] */ int buffer_size);</pre>	Загрузка данных из буфера BUFFER размером BUFFER_SIZE в файл по пути PATH_TO
<pre>int serv_get_file_size(/* [string][in] */ const unsigned char *path_file);</pre>	Возвращает длину файла, лежащего по пути PATH_FILE в байтах. Вызывается перед скачиванием файла, для определения размера буфера.

Для того, чтобы закрыть соединение с сервером, необходимо ввести команду exit в терминале клиента:

```
cmd
Microsoft Windows [Version 10.0.19041.572]
(c) Корпорация Майкрософт (Microsoft Corporation), 2020. Все права защищены.

C:\Windows\system32>"d:\IBKS\3 year\5 semester\BSIT\Labs\Lab 02\Client\Release\Client.exe"
Ip:port: 127.0.0.1:9000
Login: Max
Password: 0707
Serv#
Serv#
Serv# help
    rm [path] -- remove file
    down [path_from] [path_to] -- download file
    up [path_from] [path_to] -- upload file
    !! path without spaces!!
    exit
Serv# up d:\IBKS\man\export.rar d:\IBKS\man\export2.rar
Serv# down d:\IBKS\man\export2.rar d:\IBKS\man\export3.rar
Serv# rm d:\IBKS\man\export2.rar
Serv# rm d:\IBKS\man\export2.rar
Error
Serv# fjkldf;jkl
Incorrect command
Serv# exit

C:\Windows\system32>
```

Рисунок 6 – завершение работы клиента

КОНТРОЛЬНЫЕ ВОПРОСЫ

1) *В чем различия между statefull и stateless серверами?*

Серверы, работающие по stateless протоколу не сохраняют никакой информации о подключенных клиентах, и выполнение каждой команды на клиентской части будет сопровождаться повторным подключением к первому. В протоколе statefull клиент не нуждается в повторных подключениях и имеет возможность выполнять команды одну за другой без лишних операций.

2) *Что такое имперсонация?*

Имперсонация – операция, позволяющая работать какой либо функции или программе от имени представленной учётной записи с её параметрами безопасности.

3) *Что такое LPC? чем этот механизм отличается от RPC?*

Механизм LPC — передает сообщения между клиентским и серверным процессами на одном компьютере. LPC является гибкой, оптимизированной версией RPC (remote procedure call), стандартного механизма взаимодействия между клиентскими и серверными процессами через сеть.

ВЫВОДЫ

В ходе выполнения лабораторной работы был получен опыт в разработке двухуровневых сетей используя интерфейс RCP. Был реализован удаленный файловый менеджер с имперсонацией клиента.

ПРИЛОЖЕНИЕ А

Интерфейс хранимых процедур

```
[  
    uuid(0f3fc7a7-cb51-437d-8a37-edecd921fdd3),  
    version(1.0),  
    implicit_handle(handle_t InterfaceHandle)  
]
```

interface Interface

{

```
int serv_login(  
    [in, string] const wchar_t* login,  
    [in, string] const wchar_t* password  
);
```

```
int serv_remove(  
    [in, string] const char* path_file  
);
```

```
int serv_download(  
    [in, string] const char* path_from,  
    [out, size_is(buffer_size)] char* buffer,  
    [in] int buffer_size  
);
```

```
int serv_upload(  
    [in, string] const char* path_to,  
    [in, size_is(buffer_size)] const char* buffer,  
    [in] int buffer_size
```

```
);
```

```
int serv_get_file_size(  
    [in, string] const char* path_file  
);
```

```
}
```

ПРИЛОЖЕНИЕ Б

Программа-сервер

main.cpp

```
#include "server.h"

void* __RPC_USER midl_user_allocate(size_t size)
{
    return malloc(size);
}

void __RPC_USER midl_user_free(void* p)
{
    free(p);
}

int main()
{
    Server server("9000");
    server.listen();

    return 0;
}
```

server.h

```
#ifndef SERVER_H
#define SERVER_H

#include <iostream>
#include <cassert>

#include "AppInterface.h"
#pragma comment(lib, "rpcrt4.lib")

class Server {
public:
    Server(const char* port);
    ~Server();
    void listen();
};

#endif
```

server.cpp

```
#include "server.h"

#include <iostream>
#include <string>
#include <fstream>

Server::Server(const char* port) {
    RPC_STATUS status = RpcServerUseProtseqEpA(
        (RPC_CSTR)"ncacn_ip_tcp",
        RPC_C_PROTSEQ_MAX_REQS_DEFAULT,
        (RPC_CSTR)port,
        NULL
    );
    assert(status == RPC_S_OK);
}
```

```

status = RpcServerRegisterIf2(
    Interface_v1_0_s_ifspec, // Interface to register.
    NULL, // Use the MIDL generated entry-point vector.
    NULL, // Use the MIDL generated entry-point vector.
    RPC_IF_ALLOW_CALLBACKS_WITH_NO_AUTH, // Forces use of security callback.
    RPC_C_LISTEN_MAX_CALLS_DEFAULT, // Use default number of concurrent calls.
    (unsigned)-1, // Infinite max size of incoming data blocks.
    NULL); // Naive security callback (always allow).
assert(status == RPC_S_OK);

status = RpcServerRegisterAuthInfoA(
    NULL,
    RPC_C_AUTHN_WINNT,
    NULL,
    NULL
); // check client username/password
assert(status == RPC_S_OK);
}

Server::~Server() {
}

void Server::listen()
{
    std::cout << "Listening" << std::endl;

    RPC_STATUS status = RpcServerListen(
        1, // Recommended minimum number of threads.
        RPC_C_LISTEN_MAX_CALLS_DEFAULT, // Recommended maximum number of threads.
        0); // Start listening now.

    assert(status == RPC_S_OK);
}

// Хранимые процедуры
handle_t InterfaceHandle = 0;

int serv_remove(
    /* [string][in] */ const unsigned char* path_file)
{
    int ret = remove((const char*)path_file);
    if (!ret) {
        std::cout << "Remove " << path_file << "\n";
    }

    return ret;
}

int serv_download(
    /* [string][in] */ const unsigned char* path_from,
    /* [size_is][string][out] */ unsigned char* buffer,
    /* [in] */ int buffer_size)
{
    std::ifstream file_from((char*)path_from, std::ios::binary);
    if (!file_from.is_open()) {
        return -1;
    }
    file_from.read((char*)buffer, buffer_size);
    file_from.close();

    std::cout << "Download " << path_from << "\n";

    return 0;
}

```

```

int serv_upload(
    /* [string][in] */ const unsigned char* path_to,
    /* [size_is][in] */ const unsigned char* buffer,
    /* [in] */ int buffer_size)
{
    std::ofstream file_to((char*)path_to, std::ios::binary);
    file_to.write((char*)buffer, buffer_size);
    file_to.close();

    std::cout << "Upload " << path_to << "\n";

    return 0;
}

```

```

int serv_login(
    /* [string][in] */ const unsigned char* login,
    /* [string][in] */ const unsigned char* password)
{
    bool status = false;

    wchar_t login_[10];
    wchar_t password_[10];

    int i;
    for (i = 0; i < strlen((char*)login); ++i) {
        login_[i] = login[i];
    }
    login_[i] = 0;

    i = 0;
    for (i = 0; i < strlen((char*)password); ++i) {
        password_[i] = password[i];
    }
    password_[i] = 0;

    status = LogonUser(
        login_,
        NULL,
        password_,
        LOGON32_LOGON_INTERACTIVE,
        LOGON32_PROVIDER_DEFAULT,
        &InterfaceHandle
    ) && ImpersonateLoggedOnUser(InterfaceHandle);

    if (status) {
        std::cout << "New connection: " << login;

        return 0;
    }
    else return GetLastError();
}

```

```

int serv_get_file_size(
    /* [string][in] */ const unsigned char* path_file)
{
    std::fstream file;
    file.open((const char*)path_file);

    if (!file.is_open()) {
        return -1;
    }
}

```

```

int size = 0;
file.seekg(0, std::ios::end);
size = static_cast<int>(file.tellg());
file.close();

return size;

```

```

}

```

AppInterface.h

```

/* this ALWAYS GENERATED file contains the definitions for the interfaces */

```

```

/* File created by MIDL compiler version 8.01.0622 */
/* at Tue Jan 19 06:14:07 2038
*/
/* Compiler settings for AppInterface.idl:
    Oicf, W1, Zp8, env=Win32 (32b run), target_arch=X86 8.01.0622
    protocol : dce , ms_ext, app_config, c_ext, robust
    error checks: allocation ref bounds_check enum stub_data
    VC __declspec() decoration level:
        __declspec(uuid()), __declspec(selectany), __declspec(novtable)
        DECLSPEC_UUID(), MIDL_INTERFACE()
*/
/* @@MIDL_FILE_HEADING(  ) */

```

```

#pragma warning( disable: 4049 ) /* more than 64k source lines */

```

```

/* verify that the <rpcndr.h> version is high enough to compile this file*/
#ifndef __REQUIRED_RPCNDR_H_VERSION__
#define __REQUIRED_RPCNDR_H_VERSION__ 475
#endif

```

```

#include "rpc.h"
#include "rpcndr.h"

```

```

#ifndef __RPCNDR_H_VERSION__
#error this stub requires an updated version of <rpcndr.h>
#endif /* __RPCNDR_H_VERSION__ */

```

```

#ifndef __AppInterface_h__
#define __AppInterface_h__

```

```

#if defined(_MSC_VER) && (_MSC_VER >= 1020)
#pragma once
#endif

```

```

/* Forward Declarations */

```

```

#ifdef __cplusplus
extern "C"{
#endif

```

```

#ifndef __Interface_INTERFACE_DEFINED__
#define __Interface_INTERFACE_DEFINED__

```

```

/* interface Interface */
/* [implicit_handle][version][uuid] */

```

```

int serv_login(

```

```

    /* [string][in] */ const unsigned char *login,
    /* [string][in] */ const unsigned char *password);

int serv_remove(
    /* [string][in] */ const unsigned char *path_file);

int serv_download(
    /* [string][in] */ const unsigned char *path_from,
    /* [size_is][out] */ unsigned char *buffer,
    /* [in] */ int buffer_size);

int serv_upload(
    /* [string][in] */ const unsigned char *path_to,
    /* [size_is][in] */ const unsigned char *buffer,
    /* [in] */ int buffer_size);

int serv_get_file_size(
    /* [string][in] */ const unsigned char *path_file);

extern handle_t InterfaceHandle;

extern RPC_IF_HANDLE Interface_v1_0_c_ifspec;
extern RPC_IF_HANDLE Interface_v1_0_s_ifspec;
#endif /* __Interface_INTERFACE_DEFINED__ */

/* Additional Prototypes for ALL interfaces */

/* end of Additional Prototypes */

#ifdef __cplusplus
}
#endif

#endif

```

AppInterface_s.c

```

/* this ALWAYS GENERATED file contains the RPC server stubs */

/* File created by MIDL compiler version 8.01.0622 */
/* at Tue Jan 19 06:14:07 2038
*/
/* Compiler settings for AppInterface.idl:
    Oicf, W1, Zp8, env=Win32 (32b run), target_arch=X86 8.01.0622
    protocol : dce , ms_ext, app_config, c_ext, robust
    error checks: allocation ref bounds_check enum stub_data
    VC __declspec() decoration level:
        __declspec(uuid()), __declspec(selectany), __declspec(novtable)
        DECLSPEC_UUID(), MIDL_INTERFACE()
*/
/* @@MIDL_FILE_HEADING(  ) */

#if !defined(_M_IA64) && !defined(_M_AMD64) && !defined(_ARM_)

#pragma warning( disable: 4049 ) /* more than 64k source lines */
#if _MSC_VER >= 1200
#pragma warning(push)
#endif

```



```

#pragma warning( disable: 4211 ) /* redefine extern to static */
#pragma warning( disable: 4232 ) /* dllimport identity*/
#pragma warning( disable: 4024 ) /* array to pointer mapping*/
#pragma warning( disable: 4100 ) /* unreferenced arguments in x86 call */

#pragma optimize("", off )

#include <string.h>
#include "AppInterface.h"

#define TYPE_FORMAT_STRING_SIZE 23
#define PROC_FORMAT_STRING_SIZE 211
#define EXPR_FORMAT_STRING_SIZE 1
#define TRANSMIT_AS_TABLE_SIZE 0
#define WIRE_MARSHAL_TABLE_SIZE 0

typedef struct _AppInterface_MIDL_TYPE_FORMAT_STRING
{
    short        Pad;
    unsigned char Format[ TYPE_FORMAT_STRING_SIZE ];
} AppInterface_MIDL_TYPE_FORMAT_STRING;

typedef struct _AppInterface_MIDL_PROC_FORMAT_STRING
{
    short        Pad;
    unsigned char Format[ PROC_FORMAT_STRING_SIZE ];
} AppInterface_MIDL_PROC_FORMAT_STRING;

typedef struct _AppInterface_MIDL_EXPR_FORMAT_STRING
{
    long         Pad;
    unsigned char Format[ EXPR_FORMAT_STRING_SIZE ];
} AppInterface_MIDL_EXPR_FORMAT_STRING;

static const RPC_SYNTAX_IDENTIFIER _RpcTransferSyntax =
{{0x8A885D04,0x1CEB,0x11C9,{0x9F,0xE8,0x08,0x00,0x2B,0x10,0x48,0x60}},{2,0}};

extern const AppInterface_MIDL_TYPE_FORMAT_STRING AppInterface__MIDL_TypeFormatString;
extern const AppInterface_MIDL_PROC_FORMAT_STRING AppInterface__MIDL_ProcFormatString;
extern const AppInterface_MIDL_EXPR_FORMAT_STRING AppInterface__MIDL_ExprFormatString;

/* Standard interface: Interface, ver. 1.0,
   GUID={0x0f3fc7a7,0xcb51,0x437d,{0x8a,0x37,0xed,0xec,0xd9,0x21,0xfd,0xd3}} */

extern const MIDL_SERVER_INFO Interface_ServerInfo;

extern const RPC_DISPATCH_TABLE Interface_v1_0_DispatchTable;

static const RPC_SERVER_INTERFACE Interface__RpcServerInterface =
{
    sizeof(RPC_SERVER_INTERFACE),
    {{0x0f3fc7a7,0xcb51,0x437d,{0x8a,0x37,0xed,0xec,0xd9,0x21,0xfd,0xd3}},{1,0}},
    {{0x8A885D04,0x1CEB,0x11C9,{0x9F,0xE8,0x08,0x00,0x2B,0x10,0x48,0x60}},{2,0}},
    (RPC_DISPATCH_TABLE*)&Interface_v1_0_DispatchTable,
    0,
    0,
    0,
    &Interface_ServerInfo,
    0x04000000
};
RPC_IF_HANDLE Interface_v1_0_s_ifspec = (RPC_IF_HANDLE)& Interface__RpcServerInterface;

```

```

extern const MIDL_STUB_DESC Interface_StubDesc;

#if !defined(__RPC_WIN32__)
#error Invalid build platform for this stub.
#endif

#if !(TARGET_IS_NT50_OR_LATER)
#error You need Windows 2000 or later to run this stub because it uses these features:
#error /robust command line switch.
#error However, your C/C++ compilation flags indicate you intend to run this app on earlier
systems.
#error This app will fail with the RPC_X_WRONG_STUB_VERSION error.
#endif

static const AppInterface_MIDL_PROC_FORMAT_STRING AppInterface__MIDL_ProcFormatString =
{
    0,
    {

        /* Procedure serv_login */

                0x32,          /* FC_BIND_PRIMITIVE */
                0x48,          /* Old Flags: */
/* 2 */      NdrFcLong( 0x0 ), /* 0 */
/* 6 */      NdrFcShort( 0x0 ), /* 0 */
/* 8 */      NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
/* 10 */     NdrFcShort( 0x0 ), /* 0 */
/* 12 */     NdrFcShort( 0x8 ), /* 8 */
/* 14 */     0x46,          /* Oi2 Flags: clt must size, has return, has ext, */
                0x3,          /* 3 */
/* 16 */     0x8,          /* 8 */
                0x1,          /* Ext Flags: new corr desc, */
/* 18 */     NdrFcShort( 0x0 ), /* 0 */
/* 20 */     NdrFcShort( 0x0 ), /* 0 */
/* 22 */     NdrFcShort( 0x0 ), /* 0 */

        /* Parameter login */

/* 24 */     NdrFcShort( 0x10b ), /* Flags: must size, must free, in, simple ref, */
/* 26 */     NdrFcShort( 0x0 ), /* x86 Stack size/offset = 0 */
/* 28 */     NdrFcShort( 0x4 ), /* Type Offset=4 */

        /* Parameter password */

/* 30 */     NdrFcShort( 0x10b ), /* Flags: must size, must free, in, simple ref, */
/* 32 */     NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 34 */     NdrFcShort( 0x4 ), /* Type Offset=4 */

        /* Return value */

/* 36 */     NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 38 */     NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 40 */     0x8,          /* FC_LONG */
                0x0,          /* 0 */

        /* Procedure serv_remove */

/* 42 */     0x32,          /* FC_BIND_PRIMITIVE */
                0x48,          /* Old Flags: */
/* 44 */     NdrFcLong( 0x0 ), /* 0 */
/* 48 */     NdrFcShort( 0x1 ), /* 1 */
/* 50 */     NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 52 */     NdrFcShort( 0x0 ), /* 0 */
    }
};

```

```

/* 54 */      NdrFcShort( 0x8 ), /* 8 */
/* 56 */      0x46, /* 0i2 Flags: clt must size, has return, has ext, */
                0x2, /* 2 */
/* 58 */      0x8, /* 8 */
                0x1, /* Ext Flags: new corr desc, */
/* 60 */      NdrFcShort( 0x0 ), /* 0 */
/* 62 */      NdrFcShort( 0x0 ), /* 0 */
/* 64 */      NdrFcShort( 0x0 ), /* 0 */

/* Parameter path_file */

/* 66 */      NdrFcShort( 0x10b ), /* Flags: must size, must free, in, simple ref, */
/* 68 */      NdrFcShort( 0x0 ), /* x86 Stack size/offset = 0 */
/* 70 */      NdrFcShort( 0x4 ), /* Type Offset=4 */

/* Return value */

/* 72 */      NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 74 */      NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 76 */      0x8, /* FC_LONG */
                0x0, /* 0 */

/* Procedure serv_download */

/* 78 */      0x32, /* FC_BIND_PRIMITIVE */
                0x48, /* Old Flags: */
/* 80 */      NdrFcLong( 0x0 ), /* 0 */
/* 84 */      NdrFcShort( 0x2 ), /* 2 */
/* 86 */      NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
/* 88 */      NdrFcShort( 0x8 ), /* 8 */
/* 90 */      NdrFcShort( 0x8 ), /* 8 */
/* 92 */      0x47, /* 0i2 Flags: srv must size, clt must size, has return, has ext, */
/*
                0x4, /* 4 */
/* 94 */      0x8, /* 8 */
                0x3, /* Ext Flags: new corr desc, clt corr check, */
/* 96 */      NdrFcShort( 0x1 ), /* 1 */
/* 98 */      NdrFcShort( 0x0 ), /* 0 */
/* 100 */     NdrFcShort( 0x0 ), /* 0 */

/* Parameter path_from */

/* 102 */     NdrFcShort( 0x10b ), /* Flags: must size, must free, in, simple ref, */
/* 104 */     NdrFcShort( 0x0 ), /* x86 Stack size/offset = 0 */
/* 106 */     NdrFcShort( 0x4 ), /* Type Offset=4 */

/* Parameter buffer */

/* 108 */     NdrFcShort( 0x113 ), /* Flags: must size, must free, out, simple ref, */
/* 110 */     NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 112 */     NdrFcShort( 0xa ), /* Type Offset=10 */

/* Parameter buffer_size */

/* 114 */     NdrFcShort( 0x48 ), /* Flags: in, base type, */
/* 116 */     NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 118 */     0x8, /* FC_LONG */
                0x0, /* 0 */

/* Return value */

/* 120 */     NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 122 */     NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
/* 124 */     0x8, /* FC_LONG */

```

```

                                0x0,          /* 0 */

/* Procedure serv_upload */

/* 126 */    0x32,          /* FC_BIND_PRIMITIVE */
                                0x48,          /* Old Flags: */
/* 128 */    NdrFcLong( 0x0 ),    /* 0 */
/* 132 */    NdrFcShort( 0x3 ),    /* 3 */
/* 134 */    NdrFcShort( 0x10 ),    /* x86 Stack size/offset = 16 */
/* 136 */    NdrFcShort( 0x8 ),    /* 8 */
/* 138 */    NdrFcShort( 0x8 ),    /* 8 */
/* 140 */    0x46,          /* Oi2 Flags: clt must size, has return, has ext, */
                                0x4,          /* 4 */
/* 142 */    0x8,          /* 8 */
                                0x5,          /* Ext Flags: new corr desc, srv corr check, */
/* 144 */    NdrFcShort( 0x0 ),    /* 0 */
/* 146 */    NdrFcShort( 0x1 ),    /* 1 */
/* 148 */    NdrFcShort( 0x0 ),    /* 0 */

/* Parameter path_to */

/* 150 */    NdrFcShort( 0x10b ), /* Flags: must size, must free, in, simple ref, */
/* 152 */    NdrFcShort( 0x0 ),    /* x86 Stack size/offset = 0 */
/* 154 */    NdrFcShort( 0x4 ),    /* Type Offset=4 */

/* Parameter buffer */

/* 156 */    NdrFcShort( 0x10b ), /* Flags: must size, must free, in, simple ref, */
/* 158 */    NdrFcShort( 0x4 ),    /* x86 Stack size/offset = 4 */
/* 160 */    NdrFcShort( 0xa ),    /* Type Offset=10 */

/* Parameter buffer_size */

/* 162 */    NdrFcShort( 0x48 ), /* Flags: in, base type, */
/* 164 */    NdrFcShort( 0x8 ),    /* x86 Stack size/offset = 8 */
/* 166 */    0x8,          /* FC_LONG */
                                0x0,          /* 0 */

/* Return value */

/* 168 */    NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 170 */    NdrFcShort( 0xc ),    /* x86 Stack size/offset = 12 */
/* 172 */    0x8,          /* FC_LONG */
                                0x0,          /* 0 */

/* Procedure serv_get_file_size */

/* 174 */    0x32,          /* FC_BIND_PRIMITIVE */
                                0x48,          /* Old Flags: */
/* 176 */    NdrFcLong( 0x0 ),    /* 0 */
/* 180 */    NdrFcShort( 0x4 ),    /* 4 */
/* 182 */    NdrFcShort( 0x8 ),    /* x86 Stack size/offset = 8 */
/* 184 */    NdrFcShort( 0x0 ),    /* 0 */
/* 186 */    NdrFcShort( 0x8 ),    /* 8 */
/* 188 */    0x46,          /* Oi2 Flags: clt must size, has return, has ext, */
                                0x2,          /* 2 */
/* 190 */    0x8,          /* 8 */
                                0x1,          /* Ext Flags: new corr desc, */
/* 192 */    NdrFcShort( 0x0 ),    /* 0 */
/* 194 */    NdrFcShort( 0x0 ),    /* 0 */
/* 196 */    NdrFcShort( 0x0 ),    /* 0 */

/* Parameter path_file */

/* 198 */    NdrFcShort( 0x10b ), /* Flags: must size, must free, in, simple ref, */

```

```

/* 200 */    NdrFcShort( 0x0 ), /* x86 Stack size/offset = 0 */
/* 202 */    NdrFcShort( 0x4 ), /* Type Offset=4 */

    /* Return value */

/* 204 */    NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 206 */    NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 208 */    0x8, /* FC_LONG */
                0x0, /* 0 */

                0x0
    }
};

static const AppInterface_MIDL_TYPE_FORMAT_STRING AppInterface__MIDL_TypeFormatString =
{
    0,
    {
        NdrFcShort( 0x0 ), /* 0 */

/* 2 */
        0x11, 0x8, /* FC_RP [simple_pointer] */

/* 4 */
        0x22, /* FC_C_CSTRING */
        0x5c, /* FC_PAD */

/* 6 */
        0x11, 0x0, /* FC_RP */
/* 8 */    NdrFcShort( 0x2 ), /* Offset= 2 (10) */
/* 10 */
        0x1b, /* FC_CARRAY */
        0x0, /* 0 */

/* 12 */    NdrFcShort( 0x1 ), /* 1 */
/* 14 */    0x28, /* Corr desc: parameter, FC_LONG */
        0x0, /* */

/* 16 */    NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 18 */    NdrFcShort( 0x0 ), /* Corr flags: */
/* 20 */    0x2, /* FC_CHAR */
        0x5b, /* FC_END */

        0x0
    }
};

static const unsigned short Interface_FormatStringOffsetTable[] =
{
    0,
    42,
    78,
    126,
    174
};

static const MIDL_STUB_DESC Interface_StubDesc =
{
    (void *)& Interface__RpcServerInterface,
    MIDL_user_allocate,
    MIDL_user_free,
    0,
    0,
    0,
    0,
    0,
    0,
    AppInterface__MIDL_TypeFormatString.Format,
    1, /* -error bounds_check flag */
    0x50002, /* Ndr library version */

```

```

0,
0x801026e, /* MIDL Version 8.1.622 */
0,
0,
0, /* notify & notify_flag routine table */
0x1, /* MIDL flag */
0, /* cs routines */
0, /* proxy/server info */
0
};

static const RPC_DISPATCH_FUNCTION Interface_table[] =
{
    NdrServerCall2,
    NdrServerCall2,
    NdrServerCall2,
    NdrServerCall2,
    NdrServerCall2,
    0
};
static const RPC_DISPATCH_TABLE Interface_v1_0_DispatchTable =
{
    5,
    (RPC_DISPATCH_FUNCTION*)Interface_table
};

static const SERVER_ROUTINE Interface_ServerRoutineTable[] =
{
    (SERVER_ROUTINE)serv_login,
    (SERVER_ROUTINE)serv_remove,
    (SERVER_ROUTINE)serv_download,
    (SERVER_ROUTINE)serv_upload,
    (SERVER_ROUTINE)serv_get_file_size
};

static const MIDL_SERVER_INFO Interface_ServerInfo =
{
    &Interface_StubDesc,
    Interface_ServerRoutineTable,
    AppInterface__MIDL_ProcFormatString.Format,
    Interface_FormatStringOffsetTable,
    0,
    0,
    0,
    0
};
#if _MSC_VER >= 1200
#pragma warning(pop)
#endif

#endif /* !defined(_M_IA64) && !defined(_M_AMD64) && !defined(_ARM_) */

```

ПРИЛОЖЕНИЕ С

Програма-клиент

main.cpp

```
#include "client.h"

void* __RPC_USER midl_user_allocate(size_t size)
{
    return malloc(size);
}

void __RPC_USER midl_user_free(void* p)
{
    free(p);
}

int main(int argc, char** argv)
{
    std::string ip_port, login, password;

    std::cout << "Ip:port: ";
    std::cin >> ip_port;
    std::cout << "Login: ";
    std::cin >> login;
    std::cout << "Password: ";
    std::cin >> password;

    Client client(ip_port);

    bool check = client.authentication(login.c_str(), password.c_str());
    if (!check) {
        std::cerr << "Authentification fail";
        return -1;
    }

    client.cmd_loop();

    return 0;
}
```

client.h

```
#ifndef CLIENT_H
#define CLIENT_H

#include <cassert>
#include <iostream>
#include <list>
#include <string>
#include <fstream>

#include "AppInterface.h"
#pragma comment(lib, "rpcrt4.lib")

class Client {
public:
    Client(std::string& ip_port);
    ~Client();

    bool authentication(const char* login, const char* password);
    unsigned char* szStringBinding;
    void cmd_loop();
};
```

```

    bool wrong_args(std::string& cmd, std::list<std::string>& t);

private:
    std::string ip, port;
    std::list<std::string> split(std::string& s, char c);

public:
    class Handler {
    public:
        static int remove(std::string& path_file) {
            return serv_remove((const unsigned char*)path_file.c_str());
        }

        static int download(std::string& path_from, std::string& path_to) {
            int file_size = serv_get_file_size((const unsigned
char*)path_from.c_str());
            if (file_size == -1) {
                std::cerr << "No such file, ";
                return -1;
            }

            unsigned char* buffer = new unsigned char[file_size];

            int ret = serv_download((const unsigned char*)path_from.c_str(), buffer,
file_size);

            if (ret) {
                std::cerr << "No such file, ";
                return -1;
            }

            std::ofstream file_to(path_to, std::ios::binary);
            file_to.write((char*)buffer, file_size);
            file_to.close();

            delete[] buffer;

            return ret;
        }

        static int upload(std::string& path_from, std::string& path_to) {
            int file_size = get_file_size(path_from);
            if (file_size == -1) {
                std::cerr << "No such file, ";
                return -1;
            }

            std::ifstream file_from(path_from, std::ios::binary);
            unsigned char* buffer = new unsigned char[file_size];

            file_from.read((char*)buffer, file_size);
            int ret = serv_upload((unsigned char*)path_to.c_str(), buffer, file_size);

            delete[] buffer;

            return ret;
        }

        static int get_file_size(std::string& path) {
            std::fstream file(path);
            if (!file.is_open()) {
                return -1;
            }

            int size = 0;

```



```

        file.seekg(0, std::ios::end);
        size = static_cast<int>(file.tellg());
        file.close();

        return size;
    }
};

#endif

```

client.cpp

```

#include "client.h"

Client::Client(std::string& ip_port) {
    ip = ip_port.substr(0, ip_port.find(':'));
    port = ip_port.substr(ip_port.find(':') + 1, ip_port.length());

    RPC_STATUS status = RpcStringBindingComposeA(
        (RPC_CSTR)"0f3fc7a7-cb51-437d-8a37-edecd921fdd3", // UUID to bind to.
        (RPC_CSTR)("ncacn_ip_tcp"), // Use TCP/IP protocol.
        (RPC_CSTR)(ip.c_str()), // TCP/IP network address
        (RPC_CSTR)(port.c_str()), // TCP/IP port to use.
        NULL, // Protocol
        &szStringBinding // String binding output.
    );
    assert(!status);

    status = RpcBindingFromStringBindingA(szStringBinding, &InterfaceHandle);
    assert(!status);
}

Client::~Client() {
    if (szStringBinding) {
        RpcStringFreeA(&szStringBinding);
    }

    RpcBindingFree(&InterfaceHandle);
}

bool Client::authentication(const char* login, const char* password)
{
    // Filling up identity struct
    SEC_WINNT_AUTH_IDENTITY_A sec;
    sec.Domain = (unsigned char*)"";
    sec.DomainLength = 0;
    sec.Flags = SEC_WINNT_AUTH_IDENTITY_ANSI;
    sec.Password = (unsigned char*)password;
    sec.PasswordLength = strlen(password);
    sec.User = (unsigned char*)login;
    sec.UserLength = strlen(login);

    RPC_S_OK == RpcBindingSetAuthInfo(
        InterfaceHandle,
        NULL,
        RPC_C_AUTHN_LEVEL_PKT_PRIVACY,
        RPC_C_AUTHN_WINNT,
        (RPC_AUTH_IDENTITY_HANDLE)&sec,

```

```

        0
    );

    int check = serv_login((unsigned char*)login, (unsigned char*)password);
    if (check) {
        std::cerr << check;
    }

    return check == 0;
}

bool Client::wrong_args(std::string& cmd, std::list<std::string>& t) {
    // только по количеству аргументов

    if (cmd == "rm") {
        return t.size() != 2;
    }
    else if (cmd == "down") {
        return t.size() != 3;
    }
    else if (cmd == "up") {
        return t.size() != 3;
    }

    return false;
}

void Client::cmd_loop()
{
    std::string line;
    std::list<std::string> tokens;

    while (true) {
        RPC_STATUS status = RpcMgmtIsServerListening(InterfaceHandle);
        if (status == RPC_S_NOT_LISTENING) {
            std::cerr << "RPC server is not listening" << std::endl;
            break;
        }

        std::cout << "Serv# ";

    metka_kostil:
        getline(std::cin, line);

        static int kostil = 0; if (kostil++ == 0) goto metka_kostil;

        if (line.empty()) {
            continue;
        }

        tokens = split(line, ' ');

        std::string& cmd = *tokens.begin();

        std::string first_arg;
        std::string second_arg;

        if (wrong_args(cmd, tokens)) {
            std::cerr << "Incorrect arguments for " << cmd << " see \"help\\\"\\n";
            continue;
        }

        if (tokens.size() > 1)
            first_arg = *(++tokens.begin());

```

```

    if (tokens.size() > 2)
        second_arg = *(++(++tokens.begin()));

    int ret = 0;

    if (cmd == "rm") {
        ret = Handler::remove(first_arg);
    }
    else if (cmd == "down") {
        ret = Handler::download(first_arg, second_arg);
    }
    else if (cmd == "up") {
        ret = Handler::upload(first_arg, second_arg);
    }
    else if (cmd == "exit") {
        return;
    }
    else if (cmd == "help") {
        std::cout << "\trm    [path] -- remove file\n";
        std::cout << "\tdown  [path_from] [path_to] -- download file\n";
        std::cout << "\tup    [path_from] [path_to] -- upload file\n";
        std::cout << "\t!! path without spaces!!\n";
        std::cout << "\texit\n";
    }
    else {
        std::cerr << "Incorrect command\n";
    }

    if (ret != 0) {
        std::cerr << "Error\n";
    }
}
}

```

```

std::list<std::string> Client::split(std::string& s, char c)
{
    std::list<std::string> res;

    size_t i = 0, j = 0;

    while (j != -1) {
        i = j;
        while (i < s.length() && s[i] == c)      i++;
        if (i == s.length()) return res;
        j = s.find(c, i);

        if (j != -1) {
            res.push_back(s.substr(i, j - i));
        }
        else {
            res.push_back(s.substr(i, s.length() - i));
        }
    }

    return res;
}

```

AppInterface_c.c

```
/* this ALWAYS GENERATED file contains the RPC client stubs */

/* File created by MIDL compiler version 8.01.0622 */
/* at Tue Jan 19 06:14:07 2038
*/
/* Compiler settings for AppInterface.idl:
    Oicf, W1, Zp8, env=Win32 (32b run), target_arch=X86 8.01.0622
    protocol : dce , ms_ext, app_config, c_ext, robust
    error checks: allocation ref bounds_check enum stub_data
    VC __declspec() decoration level:
        __declspec(uuid()), __declspec(selectany), __declspec(novtable)
        DECLSPEC_UUID(), MIDL_INTERFACE()
*/
/* @@MIDL_FILE_HEADING(  ) */

#if !defined(_M_IA64) && !defined(_M_AMD64) && !defined(_ARM_)

#pragma warning( disable: 4049 ) /* more than 64k source lines */
#if _MSC_VER >= 1200
#pragma warning(push)
#endif

#pragma warning( disable: 4211 ) /* redefine extern to static */
#pragma warning( disable: 4232 ) /* dllimport identity*/
#pragma warning( disable: 4024 ) /* array to pointer mapping*/
#pragma warning( disable: 4100 ) /* unreferenced arguments in x86 call */

#pragma optimize("", off )

#include <string.h>

#include "AppInterface.h"

#define TYPE_FORMAT_STRING_SIZE    23
#define PROC_FORMAT_STRING_SIZE    211
#define EXPR_FORMAT_STRING_SIZE    1
#define TRANSMIT_AS_TABLE_SIZE    0
#define WIRE_MARSHAL_TABLE_SIZE    0

typedef struct _AppInterface_MIDL_TYPE_FORMAT_STRING
{
    short          Pad;
    unsigned char  Format[ TYPE_FORMAT_STRING_SIZE ];
} AppInterface_MIDL_TYPE_FORMAT_STRING;

typedef struct _AppInterface_MIDL_PROC_FORMAT_STRING
{
    short          Pad;
    unsigned char  Format[ PROC_FORMAT_STRING_SIZE ];
} AppInterface_MIDL_PROC_FORMAT_STRING;

typedef struct _AppInterface_MIDL_EXPR_FORMAT_STRING
{
    long           Pad;
    unsigned char  Format[ EXPR_FORMAT_STRING_SIZE ];
} AppInterface_MIDL_EXPR_FORMAT_STRING;

static const RPC_SYNTAX_IDENTIFIER  _RpcTransferSyntax =
```

```

{{0x8A885D04,0x1CEB,0x11C9,{0x9F,0xE8,0x08,0x00,0x2B,0x10,0x48,0x60}},{2,0}}};

extern const AppInterface_MIDL_TYPE_FORMAT_STRING AppInterface__MIDL_TypeFormatString;
extern const AppInterface_MIDL_PROC_FORMAT_STRING AppInterface__MIDL_ProcFormatString;
extern const AppInterface_MIDL_EXPR_FORMAT_STRING AppInterface__MIDL_ExprFormatString;

#define GENERIC_BINDING_TABLE_SIZE 0

/* Standard interface: Interface, ver. 1.0,
   GUID={0x0f3fc7a7,0xcb51,0x437d,{0x8a,0x37,0xed,0xec,0xd9,0x21,0xfd,0xd3}} */

handle_t InterfaceHandle;

static const RPC_CLIENT_INTERFACE Interface__RpcClientInterface =
{
    sizeof(RPC_CLIENT_INTERFACE),
    {{0x0f3fc7a7,0xcb51,0x437d,{0x8a,0x37,0xed,0xec,0xd9,0x21,0xfd,0xd3}},{1,0}},
    {{0x8A885D04,0x1CEB,0x11C9,{0x9F,0xE8,0x08,0x00,0x2B,0x10,0x48,0x60}},{2,0}},
    0,
    0,
    0,
    0,
    0,
    0x00000000
};
RPC_IF_HANDLE Interface_v1_0_c_ifspec = (RPC_IF_HANDLE)& Interface__RpcClientInterface;

extern const MIDL_STUB_DESC Interface_StubDesc;

static RPC_BINDING_HANDLE Interface__MIDL_AutoBindHandle;

int serv_login(
    /* [string][in] */ const unsigned char *login,
    /* [string][in] */ const unsigned char *password)
{
    CLIENT_CALL_RETURN _RetVal;

    _RetVal = NdrClientCall2(
        (PMIDL_STUB_DESC)&Interface_StubDesc,
        (PFORMAT_STRING)&AppInterface__MIDL_ProcFormatString.Format[0],
        (unsigned char *)&login);
    return (int)_RetVal.Simple;
}

int serv_remove(
    /* [string][in] */ const unsigned char *path_file)
{
    CLIENT_CALL_RETURN _RetVal;

    _RetVal = NdrClientCall2(
        (PMIDL_STUB_DESC)&Interface_StubDesc,
        (PFORMAT_STRING)&AppInterface__MIDL_ProcFormatString.Format[42],
        (unsigned char *)&path_file);
    return (int)_RetVal.Simple;
}

```

```

int serv_download(
    /* [string][in] */ const unsigned char *path_from,
    /* [size_is][out] */ unsigned char *buffer,
    /* [in] */ int buffer_size)
{
    CLIENT_CALL_RETURN _RetVal;

    _RetVal = NdrClientCall2(
        ( PMIDL_STUB_DESC )&Interface_StubDesc,
        (PFORMAT_STRING) &AppInterface__MIDL_ProcFormatString.Format[78],
        ( unsigned char * )&path_from);
    return ( int )_RetVal.Simple;
}

int serv_upload(
    /* [string][in] */ const unsigned char *path_to,
    /* [size_is][in] */ const unsigned char *buffer,
    /* [in] */ int buffer_size)
{
    CLIENT_CALL_RETURN _RetVal;

    _RetVal = NdrClientCall2(
        ( PMIDL_STUB_DESC )&Interface_StubDesc,
        (PFORMAT_STRING) &AppInterface__MIDL_ProcFormatString.Format[126],
        ( unsigned char * )&path_to);
    return ( int )_RetVal.Simple;
}

int serv_get_file_size(
    /* [string][in] */ const unsigned char *path_file)
{
    CLIENT_CALL_RETURN _RetVal;

    _RetVal = NdrClientCall2(
        ( PMIDL_STUB_DESC )&Interface_StubDesc,
        (PFORMAT_STRING) &AppInterface__MIDL_ProcFormatString.Format[174],
        ( unsigned char * )&path_file);
    return ( int )_RetVal.Simple;
}

#if !defined(__RPC_WIN32__)
#error Invalid build platform for this stub.
#endif

#if !(TARGET_IS_NT50_OR_LATER)
#error You need Windows 2000 or later to run this stub because it uses these features:
#error /robust command line switch.
#error However, your C/C++ compilation flags indicate you intend to run this app on earlier
systems.
#error This app will fail with the RPC_X_WRONG_STUB_VERSION error.
#endif

static const AppInterface_MIDL_PROC_FORMAT_STRING AppInterface__MIDL_ProcFormatString =
{

```

```

0,
{

/* Procedure serv_login */

        0x32,          /* FC_BIND_PRIMITIVE */
        0x48,          /* Old Flags: */
/* 2 */      NdrFcLong( 0x0 ), /* 0 */
/* 6 */      NdrFcShort( 0x0 ), /* 0 */
/* 8 */      NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
/* 10 */     NdrFcShort( 0x0 ), /* 0 */
/* 12 */     NdrFcShort( 0x8 ), /* 8 */
/* 14 */     0x46,        /* Oi2 Flags: clt must size, has return, has ext, */
        0x3,          /* 3 */
/* 16 */     0x8,         /* 8 */
        0x1,          /* Ext Flags: new corr desc, */
/* 18 */     NdrFcShort( 0x0 ), /* 0 */
/* 20 */     NdrFcShort( 0x0 ), /* 0 */
/* 22 */     NdrFcShort( 0x0 ), /* 0 */

/* Parameter login */

/* 24 */     NdrFcShort( 0x10b ), /* Flags: must size, must free, in, simple ref, */
/* 26 */     NdrFcShort( 0x0 ), /* x86 Stack size/offset = 0 */
/* 28 */     NdrFcShort( 0x4 ), /* Type Offset=4 */

/* Parameter password */

/* 30 */     NdrFcShort( 0x10b ), /* Flags: must size, must free, in, simple ref, */
/* 32 */     NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 34 */     NdrFcShort( 0x4 ), /* Type Offset=4 */

/* Return value */

/* 36 */     NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 38 */     NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 40 */     0x8,          /* FC_LONG */
        0x0,          /* 0 */

/* Procedure serv_remove */

/* 42 */     0x32,          /* FC_BIND_PRIMITIVE */
        0x48,          /* Old Flags: */
/* 44 */     NdrFcLong( 0x0 ), /* 0 */
/* 48 */     NdrFcShort( 0x1 ), /* 1 */
/* 50 */     NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 52 */     NdrFcShort( 0x0 ), /* 0 */
/* 54 */     NdrFcShort( 0x8 ), /* 8 */
/* 56 */     0x46,        /* Oi2 Flags: clt must size, has return, has ext, */
        0x2,          /* 2 */
/* 58 */     0x8,         /* 8 */
        0x1,          /* Ext Flags: new corr desc, */
/* 60 */     NdrFcShort( 0x0 ), /* 0 */
/* 62 */     NdrFcShort( 0x0 ), /* 0 */
/* 64 */     NdrFcShort( 0x0 ), /* 0 */

/* Parameter path_file */

/* 66 */     NdrFcShort( 0x10b ), /* Flags: must size, must free, in, simple ref, */
/* 68 */     NdrFcShort( 0x0 ), /* x86 Stack size/offset = 0 */
/* 70 */     NdrFcShort( 0x4 ), /* Type Offset=4 */

/* Return value */

```

```

/* 72 */      NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 74 */      NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 76 */      0x8, /* FC_LONG */
                0x0, /* 0 */

    /* Procedure serv_download */

/* 78 */      0x32, /* FC_BIND_PRIMITIVE */
                0x48, /* Old Flags: */
/* 80 */      NdrFcLong( 0x0 ), /* 0 */
/* 84 */      NdrFcShort( 0x2 ), /* 2 */
/* 86 */      NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
/* 88 */      NdrFcShort( 0x8 ), /* 8 */
/* 90 */      NdrFcShort( 0x8 ), /* 8 */
/* 92 */      0x47, /* Oi2 Flags: srv must size, clt must size, has return, has ext, */
/*
                0x4, /* 4 */
/* 94 */      0x8, /* 8 */
                0x3, /* Ext Flags: new corr desc, clt corr check, */
/* 96 */      NdrFcShort( 0x1 ), /* 1 */
/* 98 */      NdrFcShort( 0x0 ), /* 0 */
/* 100 */     NdrFcShort( 0x0 ), /* 0 */

    /* Parameter path_from */

/* 102 */     NdrFcShort( 0x10b ), /* Flags: must size, must free, in, simple ref, */
/* 104 */     NdrFcShort( 0x0 ), /* x86 Stack size/offset = 0 */
/* 106 */     NdrFcShort( 0x4 ), /* Type Offset=4 */

    /* Parameter buffer */

/* 108 */     NdrFcShort( 0x113 ), /* Flags: must size, must free, out, simple ref, */
/* 110 */     NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 112 */     NdrFcShort( 0xa ), /* Type Offset=10 */

    /* Parameter buffer_size */

/* 114 */     NdrFcShort( 0x48 ), /* Flags: in, base type, */
/* 116 */     NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 118 */     0x8, /* FC_LONG */
                0x0, /* 0 */

    /* Return value */

/* 120 */     NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 122 */     NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
/* 124 */     0x8, /* FC_LONG */
                0x0, /* 0 */

    /* Procedure serv_upload */

/* 126 */     0x32, /* FC_BIND_PRIMITIVE */
                0x48, /* Old Flags: */
/* 128 */     NdrFcLong( 0x0 ), /* 0 */
/* 132 */     NdrFcShort( 0x3 ), /* 3 */
/* 134 */     NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
/* 136 */     NdrFcShort( 0x8 ), /* 8 */
/* 138 */     NdrFcShort( 0x8 ), /* 8 */
/* 140 */     0x46, /* Oi2 Flags: clt must size, has return, has ext, */
                0x4, /* 4 */
/* 142 */     0x8, /* 8 */
                0x5, /* Ext Flags: new corr desc, srv corr check, */
/* 144 */     NdrFcShort( 0x0 ), /* 0 */
/* 146 */     NdrFcShort( 0x1 ), /* 1 */
/* 148 */     NdrFcShort( 0x0 ), /* 0 */

```



```

        /* Parameter path_to */

/* 150 */      NdrFcShort( 0x10b ), /* Flags: must size, must free, in, simple ref, */
/* 152 */      NdrFcShort( 0x0 ), /* x86 Stack size/offset = 0 */
/* 154 */      NdrFcShort( 0x4 ), /* Type Offset=4 */

        /* Parameter buffer */

/* 156 */      NdrFcShort( 0x10b ), /* Flags: must size, must free, in, simple ref, */
/* 158 */      NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 160 */      NdrFcShort( 0xa ), /* Type Offset=10 */

        /* Parameter buffer_size */

/* 162 */      NdrFcShort( 0x48 ), /* Flags: in, base type, */
/* 164 */      NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 166 */      0x8, /* FC_LONG */
                0x0, /* 0 */

        /* Return value */

/* 168 */      NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 170 */      NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
/* 172 */      0x8, /* FC_LONG */
                0x0, /* 0 */

        /* Procedure serv_get_file_size */

/* 174 */      0x32, /* FC_BIND_PRIMITIVE */
                0x48, /* Old Flags: */
/* 176 */      NdrFcLong( 0x0 ), /* 0 */
/* 180 */      NdrFcShort( 0x4 ), /* 4 */
/* 182 */      NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 184 */      NdrFcShort( 0x0 ), /* 0 */
/* 186 */      NdrFcShort( 0x8 ), /* 8 */
/* 188 */      0x46, /* Oi2 Flags: clt must size, has return, has ext, */
                0x2, /* 2 */
/* 190 */      0x8, /* 8 */
                0x1, /* Ext Flags: new corr desc, */
/* 192 */      NdrFcShort( 0x0 ), /* 0 */
/* 194 */      NdrFcShort( 0x0 ), /* 0 */
/* 196 */      NdrFcShort( 0x0 ), /* 0 */

        /* Parameter path_file */

/* 198 */      NdrFcShort( 0x10b ), /* Flags: must size, must free, in, simple ref, */
/* 200 */      NdrFcShort( 0x0 ), /* x86 Stack size/offset = 0 */
/* 202 */      NdrFcShort( 0x4 ), /* Type Offset=4 */

        /* Return value */

/* 204 */      NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 206 */      NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 208 */      0x8, /* FC_LONG */
                0x0, /* 0 */

                0x0

    }
};

static const AppInterface_MIDL_TYPE_FORMAT_STRING AppInterface__MIDL_TypeFormatString =
{
    0,
    {

```

```

        NdrFcShort( 0x0 ), /* 0 */
/* 2 */
        0x11, 0x8, /* FC_RP [simple_pointer] */
/* 4 */
        0x22, /* FC_C_CSTRING */
        0x5c, /* FC_PAD */
/* 6 */
        0x11, 0x0, /* FC_RP */
/* 8 */ NdrFcShort( 0x2 ), /* Offset= 2 (10) */
/* 10 */
        0x1b, /* FC_CARRAY */
        0x0, /* 0 */
/* 12 */ NdrFcShort( 0x1 ), /* 1 */
/* 14 */ 0x28, /* Corr desc: parameter, FC_LONG */
        0x0, /* */
/* 16 */ NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 18 */ NdrFcShort( 0x0 ), /* Corr flags: */
/* 20 */ 0x2, /* FC_CHAR */
        0x5b, /* FC_END */
        0x0
    }
};

static const unsigned short Interface_FormatStringOffsetTable[] =
{
    0,
    42,
    78,
    126,
    174
};

static const MIDL_STUB_DESC Interface_StubDesc =
{
    (void *)& Interface___RpcClientInterface,
    MIDL_user_allocate,
    MIDL_user_free,
    &InterfaceHandle,
    0,
    0,
    0,
    0,
    AppInterface___MIDL_TypeFormatString.Format,
    1, /* -error_bounds_check flag */
    0x50002, /* Ndr library version */
    0,
    0x801026e, /* MIDL Version 8.1.622 */
    0,
    0,
    0, /* notify & notify_flag routine table */
    0x1, /* MIDL flag */
    0, /* cs routines */
    0, /* proxy/server info */
    0
};
#ifdef _MSC_VER >= 1200
#pragma warning(pop)
#endif

#endif /* !defined(_M_IA64) && !defined(_M_AMD64) && !defined(_ARM_) */

```