RPC - интерфейс в исполняемом файле

Формат RPC-процедур для MIDL'а можно достать как из сервера, так и из клиента.
Каждая процедура и составные типы параметров описываются набором байт в MIDL_PROC_FORMAT_STRING и MIDL_TYPE_FORMAT_STRING, **строках формата**, которые используются при маршалинге.
MIDL генерирует для клиента и для сервера заглушки разных типов, тип определяется директивами /Oi, /Oic, /Oicf командной строки MIDL'а. Здесь описан современный /Oicf-формат, который по умолчанию использует MIDL version 6.0.359 и младше.

.idl файл для примера

```
[
uuid(e02e2ede-e5e9-11d1-996c-10400526dbea),
version(1.0)
]
interface MOTD
{
typedef char CHAR_VEC[50];
        unsigned long Foo([in,out,string,size_is(num)] char*msg,[in]int num,[out] CHAR_VEC backmsg);
}
```

Код клиента

В C-файле, сгенерированном MIDL'ом Foo будет выглядеть примерно так:

```
unsigned long Foo(
    /* [in] */ handle_t IDL_handle,
    /* [size_is][string][out][in] */ unsigned char __RPC_FAR *msg,
    /* [in] */ int num,
    /* [out] */ CHAR_VEC backmsg)
{
    CLIENT_CALL_RETURN _RetVal;

    _RetVal = NdrClientCall2(
                  ( PMIDL_STUB_DESC  )&MOTD_StubDesc,
                  (PFORMAT_STRING) &__MIDL_ProcFormatString.Format[32],
                  ( unsigned char __RPC_FAR * )&IDL_handle);
    return ( unsigned long  )_RetVal.Simple;
}
```

Параметры NdrClientCall2:

- **&MOTD_StubDesc**
  Указатель на структуру MIDL_STUB_DESC с информацией о клиенте.
- **&__MIDL_ProcFormatString.Format[32]**
  Указатель на строку формата для **этой** процедуры
- **&IDL_handle**
  Указатель на стек параметров, передаваемых в процедуру.

Нам нужны первые два. В клиенте находим вызов NdrClientCall2

```
lea     eax, [esp+arg_0]
push    eax
push    60400C90h ; указатель на MIDL_PROC_FORMAT_STRING
push    60401AC0h ; указатель на MIDL_STUB_DESC
call    _NdrClientCall2
```

60400C90h - адрес строки формата MIDL_PROC_FORMAT_STRING, точнее, элемента в строке, с которого начинается описание именно этой процедуры. Поэтому если требуется конкретная процедура, лучше доставать ее из клиента.

По адресу **60401AC0h** находится структура

MIDL_STUB_DESC

Описана в MSDN'e. Вид в дизассембле:

```
        dd 60400A40h            ; RpcInterfaceInformation
        dd 60475C1Bh            ; pfnAllocate
        dd 60475C33h            ; pfnFree
        dd 6064B1DCh            ; IMPLICIT_HANDLE_INFO
        dd 0                    ; apfnNdrRundownRoutines
        dd 0                    ; aGenericBindingRoutinePairs
        dd 60401A4Ch            ; apfnExprEval
        dd 0                    ; aXmitQuintuple
        dd 60401832h            ; pFormatTypes
        dd 1                    ; fCheckBounds
        dd 50002h               ; Version
        dd 0                    ; pMallocFreeStruct
        dd 600015Bh             ; MIDLVersion
        dd 0                    ; CommFaultOffsets
        dd 0                    ; aUserMarshalQuadruple
        dd 0                    ; Reserved1
        dd 1                    ; Reserved2
        dd 0                    ; Reserved3
        dd 0                    ; Reserved4
        dd 0                    ; Reserved5
```

Большинство полей, скорее всего, не понадобится, за исключением:

- **RpcInterfaceInformation**
  Указатель на структуру RPC_CLIENT_INTERFACE с информацией об интерфейсе.
- **pFormatTypes**
  Указатель на структуру MIDL_TYPE_FORMAT_STRING, содержащую описание составных типов параметров.

Таким образом получаем адреса трех структур:

```
NdrClientCall2   - > MIDL_PROC_FORMAT_STRING
  |
  v
MIDL_STUB_DESC   - > RPC_CLIENT_INTERFACE
                 - > MIDL_TYPE_FORMAT_STRING
```

Необходимых для дальнейшей работы.

Структура RPC_CLIENT_INTERFACE / RPC_SERVER_INTERFACE

Вот что говорит по этому поводу первоисточник:

The RPC_CLIENT_INTERFACE structure is part of the private interface between the run-time libraries and the stubs. Most distributed applications that use Microsoft RPC do not need this structure.
The data structure is defined in the header file Rpcdcep.h. See the header file for syntax block and member definitions.

RPC_SERVER_INTERFACE не упоминается вообще. Структуры идентичны и выглядят так:

```
typedef struct _RPC_SERVER_INTERFACE
{
    unsigned int Length;
    RPC_SYNTAX_IDENTIFIER InterfaceId;
    RPC_SYNTAX_IDENTIFIER TransferSyntax;
    PRPC_DISPATCH_TABLE DispatchTable;
    unsigned int RpcProtseqEndpointCount;
    PRPC_PROTSEQ_ENDPOINT RpcProtseqEndpoint;
    RPC_MGR_EPV __RPC_FAR *DefaultManagerEpv;
    void const __RPC_FAR *InterpreterInfo;
    unsigned int Flags ;
} RPC_SERVER_INTERFACE, __RPC_FAR * PRPC_SERVER_INTERFACE;
```

Или в дизассембле

```
        dd 44h                  ; Length ;
        dd 0E02E2EDEh            ; InterfaceId.SyntaxGUID.Data1
        dw 0E5E9h                ; InterfaceId.SyntaxGUID.Data2
```

```
dw 11D1h                ; InterfaceId.SyntaxGUID.Data3
db 99h, 6Ch, 10h, 40h, 5, 26h, 0DBh, 0EAh; InterfaceId.SyntaxGUID.Data4
dw 1                    ; InterfaceId.SyntaxVersion.MajorVersion
dw 0                    ; InterfaceId.SyntaxVersion.MinorVersion
dd 8A885D04h            ; TransferSyntax.SyntaxGUID.Data1
dw 1CEBh                ; TransferSyntax.SyntaxGUID.Data2
dw 11C9h                ; TransferSyntax.SyntaxGUID.Data3
db 9Fh, 0E8h, 8, 0, 2Bh, 10h, 48h, 60h; TransferSyntax.SyntaxGUID.Data4
dw 2                    ; TransferSyntax.SyntaxVersion.MajorVersion
dw 0                    ; TransferSyntax.SyntaxVersion.MinorVersion
dd 0                    ; DispatchTable
dd 0                    ; RpcProtseqEndpointCount
dd 0                    ; RpcProtseqEndpoint
dd 0                    ; Reserved
dd 0                    ; InterpreterInfo
dd 0                    ; Flags
```

Если это клиент, нас интересует только поле **InterfaceId**. Это GUID интерфейса в обычном формате + версия.
В данном случае это **uuid(e02e2ede-e5e9-11d1-996c-10400526dbea) версии 1.0** и мы уже можем записать

```
[
uuid(e02e2ede-e5e9-11d1-996c-10400526dbea),
version(1.0)
]
interface InterfaceName
```

MIDL_PROC_FORMAT_STRING

Она даже немного описана в MSDN'e(Platform SDK: Remote Procedure Call (RPC)). Это строка формата, для каждой процедуры сначала идет заголовок (Procedure Header Descriptor), а затем описание параметров.

Procedure Header Descriptor

В угловых скобках - размер поля в байтах.

handle_type<1>	Тип RPC handle'a, указанный в .acf файле.	31 FC_BIND_GENERIC 32 FC_BIND_PRIMITIVE 33 FC_AUTO_HANDLE 34 FC_CALLBACK_HANDLE 0 explicit handle
Oi_flags<1>	Маска флагов. Везде равно 0x48. Возможные значения есть в MSDN'e.	0x08 Oi_HAS_RPCFLAGS присутствует поле [rpc_flags<4>] 0x40 Oi_USE_NEW_INIT_ROUTINES Uses Windows NT3.5 Beta2+ init routines
[rpc_flags<4>]	Необязательное поле. Присутствует, если **Oi_flags<1>&0x08**. Видимо, оставлено для совместимости. Везде 0.	**MSDN:** The rpc_flags<4> field describes how to set the RpcFlags field of the RPC_MESSAGE structure.
proc_num<2>	Порядковый номер процедуры, начиная с нулевого.	
stack_size<2>	Размер стека значений, передаваемых в процедуру.	
[explicit_handle_description<4>]	Необязательное поле. Присутствует, если handle_type<1>=0 (explicit_handle).	0x32 FC_BIND_PRIMITIVE
constant_client_buffer_size<2>		
constant_server_buffer_size<2>		

INTERPRETER_OPT_FLAGS<1>	Набор флагов. Описан в MSDN'e.	typedef struct
		{
		unsigned char ServerMustSize : 1; // 0x01
		unsigned char ClientMustSize : 1; // 0x02
		unsigned char HasReturn : 1; // 0x04
		unsigned char HasPipes : 1; // 0x08
		unsigned char Unused : 1;
		unsigned char HasAsyncUuid : 1; // 0x20
		unsigned char HasExtensions : 1; // 0x40
		//В заголовке имеется Win 2000 extension
		unsigned char HasAsyncHandle : 1; // 0x80
		} INTERPRETER_OPT_FLAGS;

number_of_params<1>	Число параметров, **включает возвращаемое значение**.	
[Win 2000 extension<8>]	Присутствует, если в поле INTERPRETER_OPT_FLAGS<1> есть флаг HasExtensions.	Описано в MSDN'e, не особенно интересно, для 64-битных стабов имеет размер<10>.

Пример структуры в файле:

```
db 0   ; handle_type<1>
db 48h ; Oi_flags<1>
dd 0   ; [rpc_flags<4>]
dw 0   ; proc_num<2>
dw 10h ; stack_size<2>
dd 32h ; [explicit_handle_description<4>]
dw 0   ; constant_client_buffer_size<2>
dw 42h ; constant_server_buffer_size<2>
db 6   ; INTERPRETER_OPT_FLAGS<1>
db 3   ; number_of_params<1>
```

содержит информацию о том, что:

- Тип хэндла вызова - explicit_handle
- Это первая процедура интерфейса (proc_num<2>=0)
- Процедура имеет возвращаемое значение и два параметра

Parameter Descriptors

Далее, один за другим описываются параметры процедуры, включая, если есть, **возвращаемое значение**. Дескриптор каждого представляет собой структуру в шесть байт.

```
PARAM_ATTRIBUTES<2>
stack_offset<2>
type_offset/base_type<2>
```

- **PARAM_ATTRIBUTES<2>**
  Набор флагов, определяющих тип параметра. Описание есть в MSDN'e.

```
typedef struct
  {
  unsigned short  MustSize            : 1;    // 0x0001
  unsigned short  MustFree            : 1;    // 0x0002
  unsigned short  IsPipe              : 1;    // 0x0004
  unsigned short  IsIn                : 1;    // 0x0008
  unsigned short  IsOut               : 1;    // 0x0010
  unsigned short  IsReturn            : 1;    // 0x0020
  unsigned short  IsBasetype          : 1;    // 0x0040
  unsigned short  IsByValue           : 1;    // 0x0080
  unsigned short  IsSimpleRef         : 1;    // 0x0100
  unsigned short  IsDontCallFreeInst  : 1;    // 0x0200
  unsigned short  SaveForAsyncFinish  : 1;    // 0x0400
  unsigned short  Unused              : 2;
  unsigned short  ServerAllocSize     : 3;    // 0xe000
  } PARAM_ATTRIBUTES, *PPARAM_ATTRIBUTES;
```

Значения некоторых флагов

MustSize	Аргумент переменного размера, имеет атрибуты [size_is()], [max_is()], или [length_is()].
MustFree	Относится к array'ам
IsIn, IsOut	Соответственно [in], [out], [in, out] аргументы
IsReturn	Возвращаемое значение
IsBasetype	Аргумент базового типа (FC_LONG, FC_SMALL, etc.)
IsByValue	Структура, передаваемая по значению(?)
IsSimpleRef	Простой указатель

- **stack_offset<2>**
  Смещение в стеке аргументов.
- **type_offset/base_type**
  Если это аргумент основного типа (установлен флаг IsBasetype в PARAM_ATTRIBUTES), то это поле описывает тип, принимая значения

```
FC_BYTE,                    // 0x01
FC_CHAR,                    // 0x02
FC_SMALL,                   // 0x03
FC_USMALL,                  // 0x04
FC_WCHAR,                   // 0x05
FC_SHORT,                   // 0x06
FC_USHORT,                  // 0x07
FC_LONG,                    // 0x08
FC_ULONG,                   // 0x09
FC_FLOAT,                   // 0x0a
FC_HYPER,                   // 0x0b
FC_DOUBLE,                  // 0x0c
FC_ENUM16,                  // 0x0d
FC_ENUM32,                  // 0x0e
FC_ERROR_STATUS_T,          // 0x10
FC_INT3264,                 // 0xb8
FC_UINT3264,                // 0xb9
```

  Если нет, то это смещение в строке формата параметров MIDL_TYPE_FORMAT_STRING.

MIDL_TYPE_FORMAT_STRING

Содержит описание типов параметров. Есть в MSDN'e
Масса значений, в общем выглядит как:

```
type<1> //FC_STRUCT, FC_CSTRING, etc.
description<>
```

Код сервера

Интерфейс сервера регистрируется функцией RpcServerRegisterIf

```
void RPC_ENTRY RpcServerRegisterIf(  RPC_IF_HANDLE IfSpec,
                                     UUID* MgrTypeUuid,
                                     RPC_MGR_EPV* MgrEpv );
```

Так как практически используется только первый параметр, он нам и нужен. **RPC_IF_HANDLE IfSpec** - это указатель на структуру RPC_SERVER_INTERFACE, идентичную RPC_CLIENT_INTERFACE. Для клиента поле InterpreterInfo будет нулевым, а для сервера это - указатель на структуру

MIDL_SERVER_INFO

Описанную в rpcndr.h как

```
typedef struct  _MIDL_SERVER_INFO_
    {
    PMIDL_STUB_DESC             pStubDesc;
    const SERVER_ROUTINE *      DispatchTable;
```

```
PFORMAT_STRING              ProcString;
const unsigned short *      FmtStringOffset;
const STUB_THUNK *          ThunkTable;
PFORMAT_STRING              LocalFormatTypes;
PFORMAT_STRING              LocalProcString;
const unsigned short *      LocalFmtStringOffset;
} MIDL_SERVER_INFO, *PMIDL_SERVER_INFO;
```

- **pStubDesc**
  Указатель на структуру MIDL_STUB_DESC, содержащую кое-какие полезные данные, в т.ч. указатель на MIDL_TYPE_FORMAT_STRING.
- **DispatchTable**
  Указатель на массив указателей на все процедуры интерфейса.
- **ProcString**
  Указатель на MIDL_PROC_FORMAT_STRING, где описания процедур идут в том же порядке.
- **FmtStringOffset**
  Смещения описаний всех составных типов в MIDL_TYPE_FORMAT_STRING.

Ну и структура

MIDL_STUB_DESC

из rpcndr.h

```
typedef struct _MIDL_STUB_DESC
    {
    void __RPC_FAR *    RpcInterfaceInformation;
    void __RPC_FAR *    (__RPC_FAR __RPC_API * pfnAllocate)(size_t);
    void                (__RPC_FAR __RPC_API * pfnFree)(void __RPC_FAR *);
    union
        {
        handle_t __RPC_FAR *            pAutoHandle;
        handle_t __RPC_FAR *            pPrimitiveHandle;
        PGENERIC_BINDING_INFO           pGenericBindingInfo;
        } IMPLICIT_HANDLE_INFO;
    const NDR_RUNDOWN __RPC_FAR *                   apfnNdrRundownRoutines;
    const GENERIC_BINDING_ROUTINE_PAIR __RPC_FAR *  aGenericBindingRoutinePairs;
    const EXPR_EVAL __RPC_FAR *                     apfnExprEval;
    const XMIT_ROUTINE_QUINTUPLE __RPC_FAR *        aXmitQuintuple;
    const unsigned char __RPC_FAR *                 pFormatTypes;
    int                                             fCheckBounds;
    unsigned long                                   Version;
    MALLOC_FREE_STRUCT __RPC_FAR *                  pMallocFreeStruct;
    long                                MIDLVersion;
    const COMM_FAULT_OFFSETS __RPC_FAR *    CommFaultOffsets;
    const USER_MARSHAL_ROUTINE_QUADRUPLE __RPC_FAR * aUserMarshalQuadruple;

    long                                    Reserved1;
    long                                    Reserved2;
    long                                    Reserved3;
    long                                    Reserved4;
    long                                    Reserved5;
    } MIDL_STUB_DESC;
```

В дизассембле

```
        dd 4D4580h              ; RpcInterfaceInformation
        dd 487103h              ; pfnAllocate
        dd 487108h              ; pfnFree
        dd 0                    ; IMPLICIT_HANDLE_INFO
        dd 0                    ; apfnNdrRundownRoutines
        dd 0                    ; aGenericBindingRoutinePairs
        dd 4D33F0h              ; apfnExprEval
        dd 0                    ; aXmitQuintuple
        dd 4D33FAh              ; pFormatTypes
        dd 1                    ; fCheckBounds
        dd 50002h               ; Version
        dd 0                    ; pMallocFreeStruct
        dd 600015Bh             ; MIDLVersion
```

```
        dd 0                    ; CommFaultOffsets
        dd 0                    ; aUserMarshalQuadruple
        dd 0                    ; Reserved1
        dd 1                    ; Reserved2
        dd 0                    ; Reserved3
        dd 0                    ; Reserved4
        dd 0                    ; Reserved5
```

Нам отсюда нужно в основном поле pFormatTypes - указатель на MIDL_TYPE_FORMAT_STRING. Итак, в сервере необходимые структуры находятся так:

```
RpcServerRegisterIf
   |
   v
MOTD_v1_0_s_ifspec - это всего лишь указатель
   |
   v
RPC_SERVER_INTERFACE (отсюда берется GUID интерфейса)
   |
   v
MIDL_SERVER_INFO      -> ServerRoutineTable
   |                  -> MIDL_PROC_FORMAT_STRING
   v
MIDL_STUB_DESC        ->MIDL_TYPE_FORMAT_STRING
```

Формат процедур и типы параметров вычисляются, как и для клиента, отличие в том, что из клиента можно получить смещение в MIDL_PROC_FORMAT_STRING для конкретной процедуры.