IDL everywhere

Andrew Tridgell
Samba Team
tridge@osdl.org
from last year ...

• Last year I presented on our new IDL based DCE/RPC implementation
  • new IDL compiler called 'pidl'
  • extensions to cope with non-traditional IDL
  • new RPC test suite in smbtorture

• Since then our use of IDL has expanded greatly
  • now used for several new non-RPC protocol libraries
  • used for an internal RPC system called IRPC
  • used for some on-disk structures
IDL and licensing

- Last year ...
  - announced intention to use a very liberal license for IDL files
  - legal work not completed in time for that conference

- License done
  - The legal issues have now been resolved, and the IDL files are now available under a very liberal license
  - We hope that all vendors will be able to use them
  - see source/librpc/idl/IDL_LICENSE.txt
IDL for non-RPC protocols

- DCE/RPC used IDL from the beginning
  - structures map to IDL very well
  - using IDL in new implementations is an obvious choice

- What about other protocols?
  - with small extensions, IDL can be used for other well structured protocols
  - not suitable for all protocols, depending on how well the protocol elements map onto IDL constructs

- NBT, DGRAM, WINS and CLDAP
  - we have found these to all be very suitable for IDL implementations
... IDL for non-RPC protocols

• Why use IDL?
  • leverages existing code generation framework
  • can automatically produce packet printing routines
  • provides for more robust parsing code
  • single source for both marshalling and unmarshalling code

• Disadvantages?
  • some constructs are awkward to put into in an IDL form
  • can be more difficult to cope with other broken implementations
NBT in IDL

• NBT is the most widely used protocol for CIFS name resolution
  • defined in RFC1001/1002
  • traditionally coded by hand
  • quite a regular structure, with some minor exceptions

• Comprehensive coverage
  • nbt.idl defines more of the NBT protocol than Samba has ever supported in the past
  • easy to read and simple to understand
  • name compression hand coded as it does not fit well into an IDL framework
typedef [bitmap16bit] bitmap {
    NBT_RCODE = 0x000F,
    NBT_FLAG_BROADCAST = 0x0010,
    NBT_FLAG_RECURSION_AVAIL = 0x0080,
    NBT_FLAG_RECURSION_DESIRED = 0x0100,
    NBT_FLAG_TRUNCATION = 0x0200,
    NBT_FLAG_AUTHORITIVE = 0x0400,
    NBT_OPCODE = 0x7800,
    NBT_FLAG_REPLY = 0x8000
} nbt_operation;

typedef [enum16bit] enum {
    NBT_QTYPE_ADDRESS = 0x0001,
    NBT_QTYPE_NAMESERVICE = 0x0002,
    NBT_QTYPE_NULL = 0x000A,
    NBT_QTYPE_NETBIOS = 0x0020,
    NBT_QTYPE_STATUS = 0x0021
} nbt_qtype;

typedef struct {
    nbt_name name;
    nbt_qtype question_type;
    nbt_qclass question_class;
} nbt_name_question;

typedef [flag(NDR_NOALIGN|NDR_BIG_ENDIAN|NDR_PAHEX),public] struct {
    uint16 name_trn_id;
    nbt_operation operation;
    uint16 qdcount;
    uint16 ancount;
    uint16 nscount;
    uint16 arcount;
    nbt_name_question questions[qdcount];
    nbt_res_rec answers[ancount];
    nbt_res_rec nsrecs[nscount];
    nbt_res_rec additional[arcount];
    [flag(NDR_REMAINING)] DATA_BLOB padding;
} nbt_name_packet;
Auto-generated packet display code

```c
request: struct nbt_name_packet
  name_trn_id : 0x566e (22126)
  operation : 0x0010 (16)
    0x00: NBT_RCODE (0)
    1: NBT_FLAG_BROADCAST
    0: NBT_FLAG_RECURSION_AVAIL
    0: NBT_FLAG_RECURSION_DESIRED
    0: NBT_FLAG_TRUNCATION
    0: NBT_FLAG_AUTHORITIVE
    0x00: NBT_OPCODE (0)
  qdcount : 0x0001 (1)
  ancount : 0x0000 (0)
  nscount : 0x0000 (0)
  arcount : 0x0000 (0)
  questions: ARRAY(1)
    questions: struct nbt_name_question
      name: struct nbt_name
        name : 'BLU'
        scope : NULL
        type : NBT_NAME_CLIENT (0x0)
      question_type : NBT_QTYPE_NETBIOS (0x20)
      question_class : NBT_QCLASS_IP (0x1)
  answers: ARRAY(0)
  nsrecs: ARRAY(0)
  additional: ARRAY(0)
  padding : DATA_BLOB length=0
```
Using generated NBT library

• 'control block' interface
  • pidl generates a structure oriented 'control block' interface
  • callers fill in fields from the IDL, and call to code generated by pidl to perform marshalling and unmarshalling
  • unlike traditional DCE/RPC, generated code is not tied to a transport, it is 'structure to bytes' and 'bytes to structure' code

• Higher level libraries
  • Higher level name resolution routines are built on top of the generated code
IDL for WINS

• Not just NBT packets
  • WINS replication protocol on TCP/42
  • not previously documented as far as I know
  • IDL for WINS replication in winsrepl.idl

• Some mysteries
  • What is the significance of the 0x7800 opcode bits?
IDL for DGRAM

- NBT UDP/138
  - General purpose datagram protocol
  - Primarily used for netlogon requests
  - Most common payload is a SMB trans packet!

- IDL in nbt.idl
  - defines a minimal SMB packet in IDL
  - defines all netlogon varients
IRPC

• Internal communication
  • A CIFS server needs to be able to communicate internally between its component parts
  • Needed for status monitoring, management and shared protocol elements (such as oplocks)
  • must be fast, flexible and easily extensible

• Can we leverage existing code?
  • Use IDL for message definition?
  • Needs 1-many messaging
  • needs more flexible structure than traditional RPC endpoints
... IRPC

- New transport
  - unix domain sockets, in DGRAM mode
  - typically achieves around 50k ops/sec on a PC
  - allows for multiple replies per request
  - requests encoded using NDR, described with IDL

- Why not ncadg?
  - endpoint model is not well suited to IRPC usage pattern
  - this leads to nca* having a much heavier weight server side impact on the code than is warranted
  - could possibly move to ncadg in the future if endpoint problem is solved
Uses of IRPC

- Status, control and management
  - retrieve status of server components
  - lists of active users, connections, NBT names etc
  - send control messages to components
  - startup, shutdown and general management tasks

- Status databases
  - Samba3 used small status databases for these tasks
  - these had a significant overhead even when not queried
  - data changes are far more frequent than data queries
  - better to only generate overhead when information is needed, not when information changes
js bindings

- Scripting RPC
  - RPC code can be tedious to write
  - a scripting interface makes for simpler development of test and management code

- Why js?
  - widely used, well understood language
  - easy to embed
  - multiple free and portable implementations
  - C-like syntax makes for easy integration with existing code
  - note that js is also known as 'ECMAscript'
ejs

• Which js implementation?
  • needs to have a small footprint
  • needs to be very portable
  • needs to be easily embedded
  • reference counted, not garbage collection

• Chose 'ejs', part of appweb
  • released under GNU GPL
  • beng actively developed
  • very good C extension hooks
  • http://www.appwebserver.org/products/ejs/ejs.html
Generating ejs bindings for RPC

• Use PIDL
  • leverage existing IDL infrastructure
  • bindings only need to do structure to structure mapping
  • types map surprisingly well

• OO interface
  • each IDL interface makes one object
  • all bitmaps, enums and constants mapped to js variables
  • objects can be overlaid, to combine functions
  • connections auto-close when object goes out of scope
js enumerate SAMR domains

/* connect to the SAMR server */
status = samr.connect(binding);
assert(status.is_ok);

/* perform a samr Connect2 operation */
io.input.system_name = NULL;
io.input.access_mask = samr.SEC_FLAG_MAXIMUM_ALLOWED;
status = samr.samr_Connect2(io);
assert(status.is_ok);
handle = io.output.connect_handle;

/* enumerate domains */
io.input.connect_handle = handle;
io.input.resume_handle = 0;
io.input.buf_size = -1;

status = samr.samr_EnumDomains(io);
assert(status.is_ok);

/* print them */
entries = io.output.sam.entries;
for (i=0; i<entries.length; i++) {
    println(entries[i].name);
}
js bindings for IRPC

• Just like RPC?
  • only fundamental difference is connect
  • recognise different binding string form, using IRPC name

• Multiple replies
  • An internal IRPC name can map to many tasks
  • each query logically generates an array of replies
  • replace io.output.* hash with io.results[] array
server side scripting

• js for web interfaces
  • ejs already designed for web server scripting
  • 'esp' (embedded server pages) gives session variables and other modern web scripting capabilities

• Common libraries
  • write js libraries
  • not tied to command line or web
  • provide higher level interfaces to IRPC management calls and RPC pipes
js bindings for SMB

• Obvious next step
  • generate bindings for Samba4 'raw' client library
  • very extensive, well tested, SMB library
  • will allow new torture tests to be written in js

• but its not IDL ...
  • 'raw' library not generated from IDL
  • need to generate bindings from C headers

• Stay tuned for js SMB bindings!
More Info

• Grab the code
  • http://devel.samba.org/
  • See Samba4 instructions

Questions?