Linux MAPI programming over ExchangeRPC

Julien Kerihuel
<j.kerihuel@openchange.org>

OpenChange Project

1 Opening Exchange to a wider world

1 OpenChange Goals

- **"The OpenChange Project aims to provide a portable Open Source implementation of Microsoft Exchange Server and Exchange protocols"**

Three main goals:

- provide a library for interoperability with Exchange protocols
- provide an alternative to Microsoft Exchange Server:
  - uses native Exchange protocols
  - provides exactly equivalent functionality when viewed from Microsoft Outlook clients
- develop a body of knowledge about the most popular groupware protocols in use commercially today

1 MAPI Overview

- MAPI is the glue between Exchange and Outlook
- Not a network protocol but a set of functions call interfaces developed by Microsoft before Microsoft Exchange existed

- **ExchangeRPC**:
  - Proprietary transport protocol for MAPI
  - Closely matches the MAPI calling interface

- 2 main protocols used in MAPI communications:
  - **NSPI**: Address Book protocol
  - **EMSMDB**: Exchange transport

1 MAPI Overview

- **EMSMDB pipe version evolved since Exchange 5.5:**
  - **Exchange 5.5 to Exchange 2000**:
    - **obfuscated content (xor 0xa5)**
    - Use EcDoConnect (0x1) and EcDoRpc function (0x2)
    - **Implemented in libmapi**
  - **Exchange 2003**:
    - AirMAPI compression algorithm introduced (possibly XPRESS based)
    - New pipe functions introduced for EcDoConnect (0xa) and EcDoRpc (0xb)
    - Some packets are still obfuscated with xor 0xa5 while others are compressed
  - **Exchange 2007:**
    - No MAPI captures realized yet

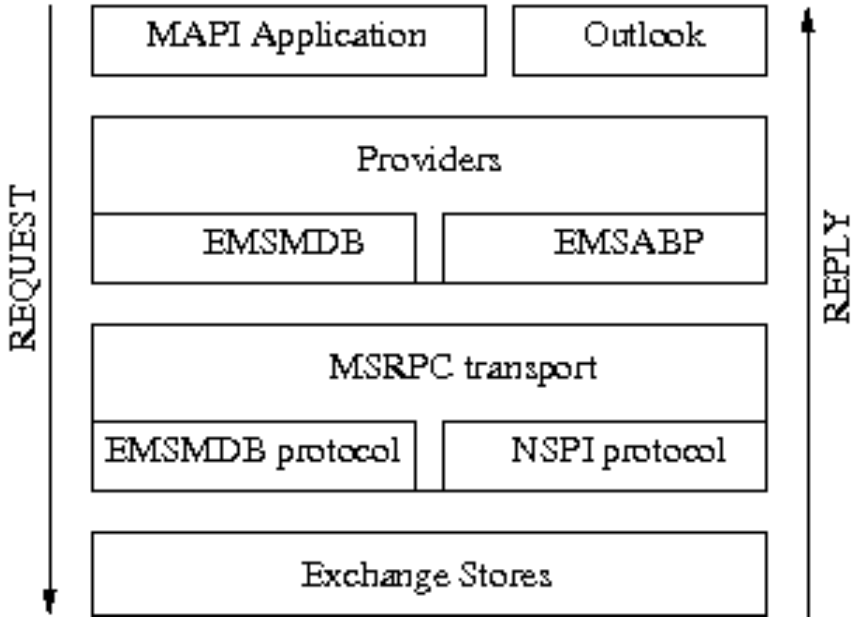
1 MAPI Overview

- OpenChange libmapi implements:
  - NSPI interface
  - EMSMDB interface from Exchange 2000.
  - MAPI API that exposes these protocols

- widely compatible implementation tested against:
  - Exchange 5.5
  - Exchange 2000
  - Exchange 2003
  - Small Business Server 2003
  - Exchange 2007 (once Administrator creates Public Folder store)

1 Describing a MAPI conversation

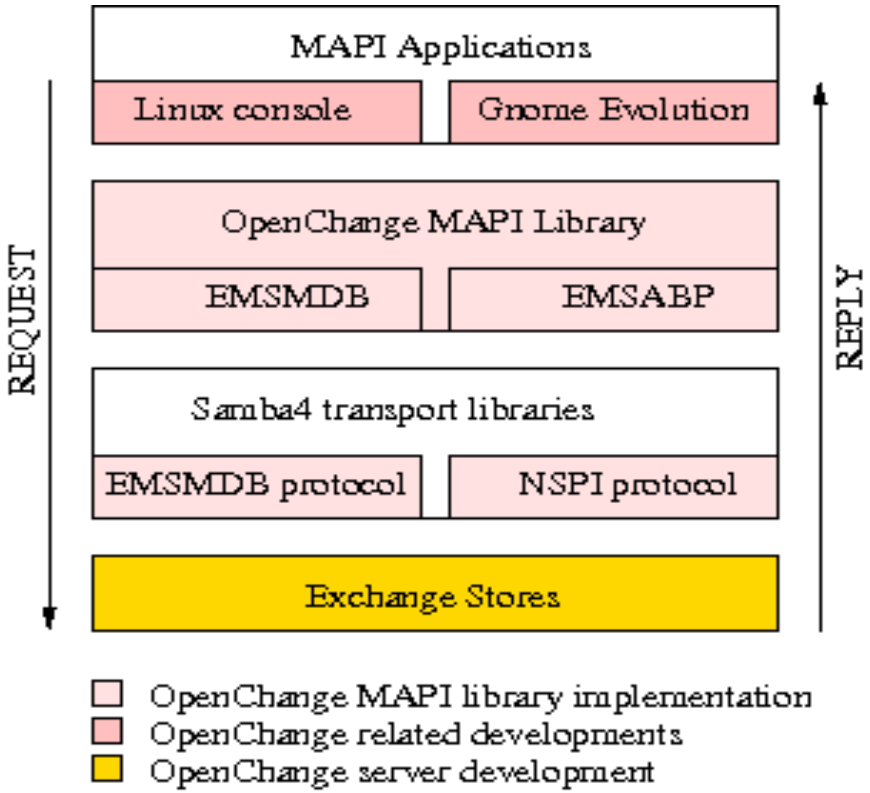
- Client-side:
  - MAPI applications call MAPI providers, using the API to pass data
  - MAPI providers pack the client or server MAPI information in a blob
  - ExchangeRPC protocol is used to transport the MAPI information, using one of two MAPI-specific protocols:
    - **EMSMDB Message Store Protocol**
    - **NSPI Address book Protocol**
- Store provider on server side:
  - extracts the MAPI blob from RPC protocol functions
  - analyzes its content
  - performs operations embedded within it



1 OpenChange Architecture

- Developed in C
- Released under the GNU GPLv2 or later license (upgrade to GPLv3 planned)

- Use Samba4 libraries and modular design:
  - MSRPC transport (dcerpc, ndr)
  - Memory allocation system (talloc)
  - Storage (ldb LDAP database)
  - Server development:
    - EMSABP provider
    - EMSMDB provider



1 MAPI library design

- **Design:**
  - **Provide a set of functions with similar semantics to the Microsoft C++ API**, maximising the benefit of shared information across the two implementations.
  - **Hides implementation details from the user**: passing in a groupware-related object and getting back a response in order to have a groupware conversation.

- **Architecture:**
  - **Global context** rather than separately instantiated objects with private members
  - **Opaque objects** passed by reference between MAPI functions
  - **No object hierarchy** implemented

1 MAPI library capabilities

- MAPI Profiles
  - Similar objective to Microsoft ones
  - Stored in LDB database and access through IProfAdmin interface
  - Store information related to:
    - Connection
    - Credentials
    - User information
- Common MAPI objects support
  - Messages, Calendars, Tasks, Contacts, Notes
  - Fetch, send, delete, move operations available
- MAPI containers
  - Create, Delete, Empty folders

1 MAPI library capabilities

- MAPI tables and search operations
  - Search, filter, restrict results
- MAPI Notifications
  - Over UDP and EMSMDB
  - NEWMAIL notification implemented
- MAPI Permissions
  - Add, Remove, Modify on a given folder
  - Pre-defined Roles and Rights
  - Support permission modifications for Default and Anonymous
- Public Folders
  - Create, Delete, Empty folders
  - Specify directory class (messages, appointments, contacts etc.)

2 Getting Started

2 OpenChange requirements

- “howto.txt from docs directory contains the most up-to-date instructions for installing and configuring openchange.”

- Installation information are provided in the LinuxConf paper associated to this presentation.

2 Setting a MAPI Profile database

- We need to create a MAPI profile database and add a profile prior running any MAPI application
- We will use **mapiprofile** command line tool

*** **Demo** ***

2 Hello Exchange Sample

- See mapi_sample1.c

- **Important Steps:**
  - Initialize MAPI library
  - Open a MAPI Session
  - Uninitialize MAPI library

- Output:

```
$ gcc mapi_sample1.c -o mapi_sample1 ‘pkg-config libmapi --cflags --libs‘
$ ./mapi_sample1
        MAPIInitialize : MAPI_E_SUCCESS (0x0)
        GetDefaultProfile : MAPI_E_SUCCESS (0x0)
        MapiLogonEx : MAPI_E_SUCCESS (0x0)
```

3

MAPI Concepts

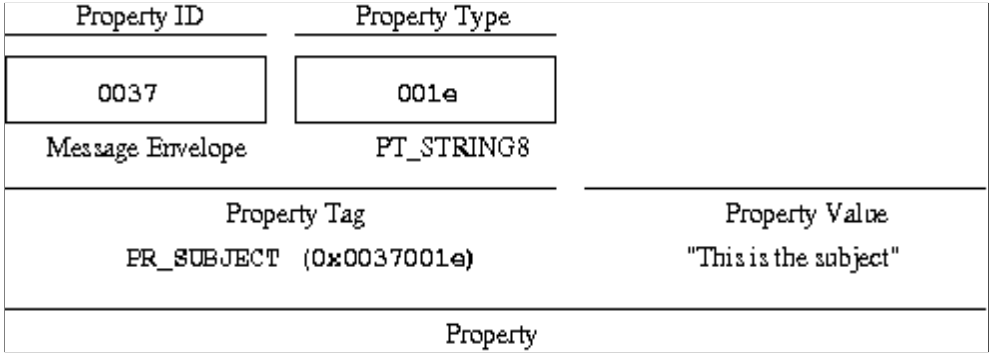
3 MAPI Objects

- "**Any MAPI data you access is associated to objects.**"
- **Design differences**:
  - With Microsoft framework instanciated objects inherits from parent classes.
  - **With OpenChange, objects are opaque**: Linux MAPI developer must know what they are doing.
- **MAPI object related functions**
  - **mapi_object_init:** Must be called prior any MAPI operation on this object
  - **mapi_object_release:** Called when an object reaches its end of life
- **MAPI Handles**
  - temporary identifiers returned by Exchange when you access or create objects on the server.
  - make reference to a particular object all along its session lifetime
  - only links between objects accessed on the client side and efficiently stored on the server side

3 MAPI Properties

- **"attributes of a MAPI object used to describe something associated with the object."**

- **Composed of:**
  - **Property Tag**:
    - Property Type
    - Property ID
  - **Property Value**
    - Value matching property type



- Related functions: **GetProps** / **SetProps**
- Related structure: **SPropValue**

3 MAPI Tables

- **"MAPI tables are used to view MAPI objects as a set of rows and columns; where objects are rows and MAPI properties columns of the table."**
- OpenChange identified 3 kind of tables associated to specific MAPI calls:
  - **GetHierarchyTable**
    - Collect information about child containers
  - **GetContentsTable**
    - Collect information about objects within a container
  - **GetTable**
    - Retrieve information about permissions for a given container or object

4 MAPI applications

4 MAPI fetchmail

- **Initialize OpenChange MAPI**
  - MapiInitialize, GetDefaultProfile and MapiLogonEx
- **Open the Message Store**
  - OpenMsgStore
- **Open Inbox folder**
  - GetDefaultFolder and OpenFolder
- **Retrieve Contents Table**
  - GetContentsTable
- **Customize the MAPI view**
  - SetColumns
- **Browse the table and dump items**
  - QueryRows, GetPropsAll and mapidump convenient functions
- **Clean-up and Exit**
  - MAPIUnitialize

4 Better than a thousand words

- **Demonstration:**
  - openchangepfadmin
  - openchangeclient
  - exchange2mbox

5

Questions?