SyncML Introduction and Implementation

Alan tu

Why you need sync ?

Problem

- Limited band width
  - How to save size of content ?
- Hard to fetch information
  - How to know changes ?
- Hard to switch the same content between all devices
  - How to switch content without the same protocol ?

Why you need sync ?

Sync is extend from fetch.
Sync means smart fetch.

VSION

"Global synchronization and integration of wireless and wireline devices " by Kahn

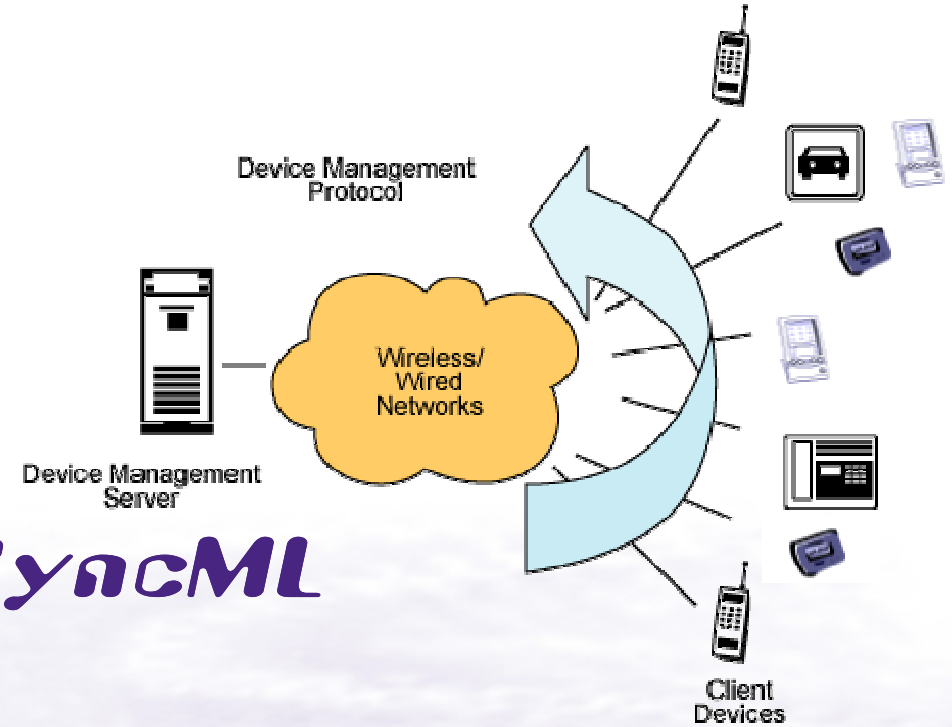
Overview of SyncML

Is a standard from OMA, so you can implement by yourself

It can be binding by may ways

It can contain many kinds of materials

It can sync by any ways



Overview of SyncML

Is a standard from OMA, so you can implement by yourself

So most of devices can implement it.

It can be binding by may ways

So it can use on wireless and wireline devices

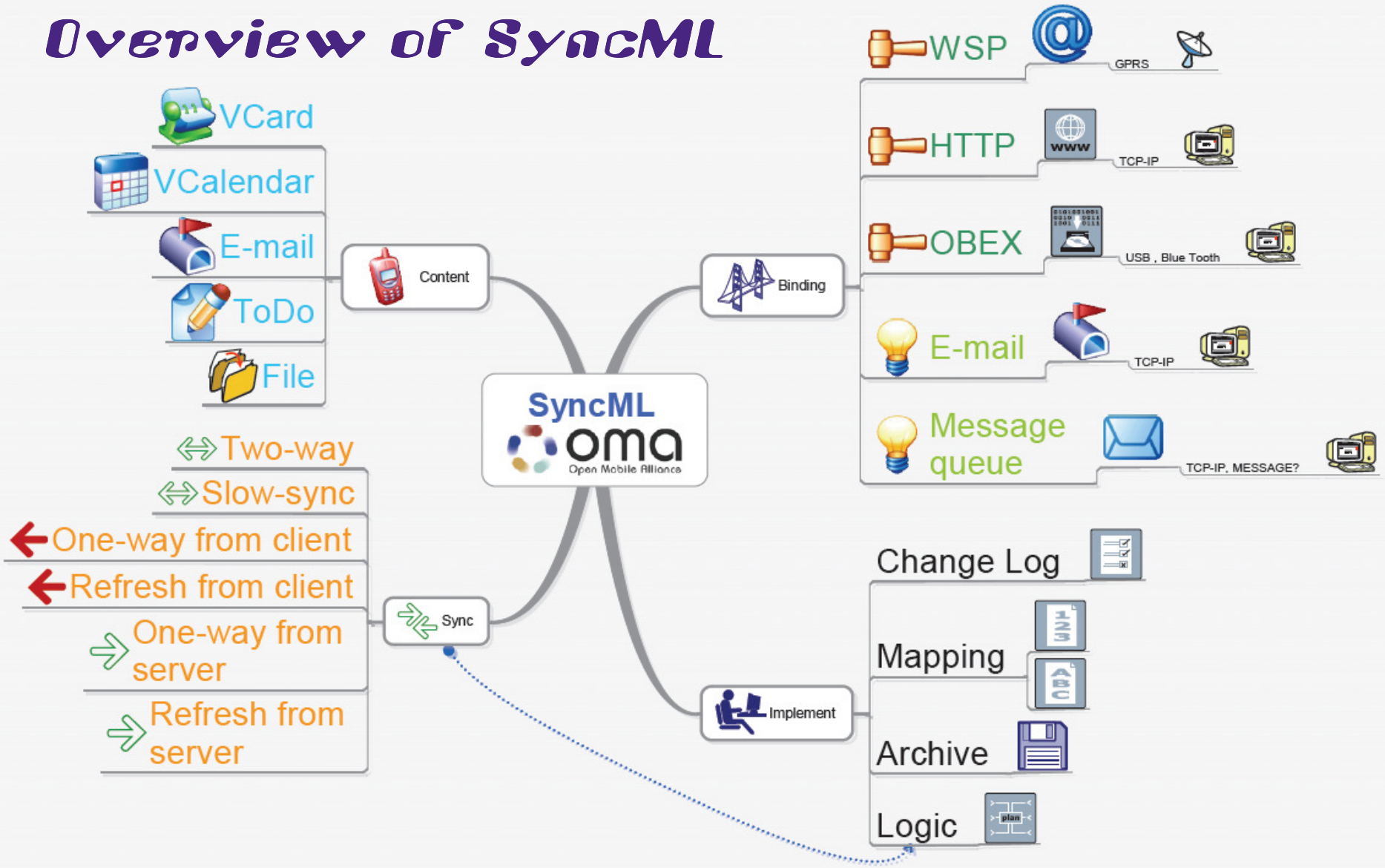
It can contain many kinds of materials

So it satisfies most of scenarios

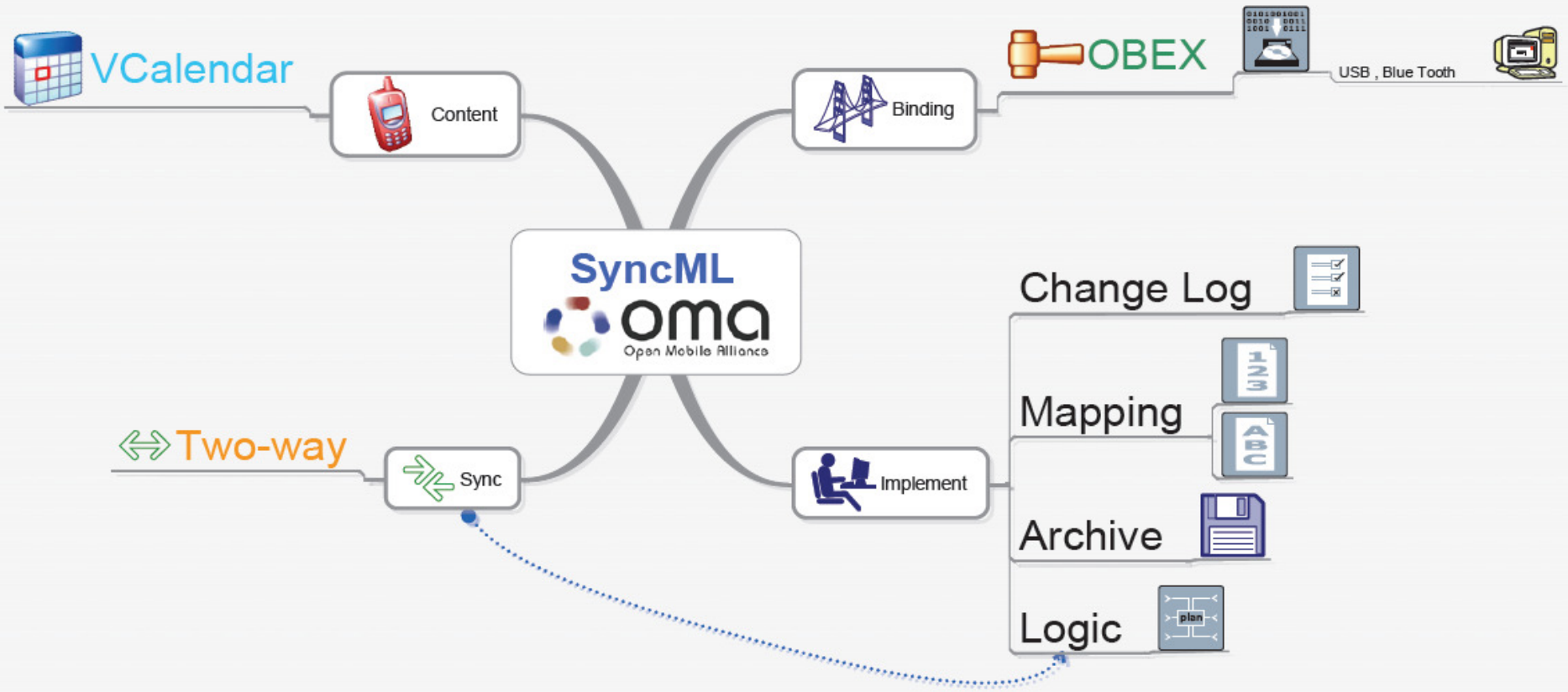
It can sync by any ways

So it is smarter to fetch data, faster, smaller

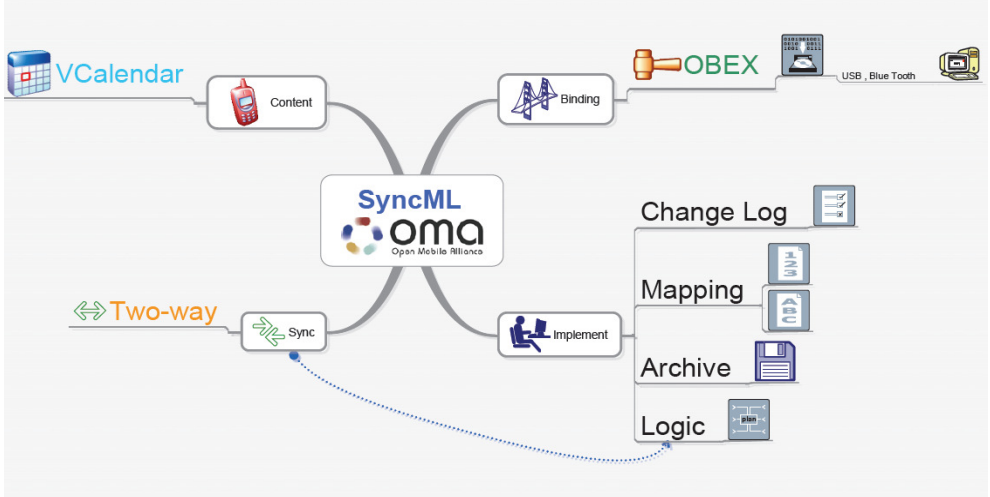
Overview of SyncML



Agenda



Agenda



- What type is VCalendar ?
- What is OBEX ?
- How to binding by OBEX ?
- How is two-way sync's flow ?
- How to implement SyncML-session ?
  - Change log
  - Mapping table
  - Storage

What is VCalendar ?

- **vCalendar** is an older **standard exchange format for calendar data** promulgated by the Internet Mail Consortium (IMC). iCalendar is a newer standard (RFC 2445) for calendar data, heavily based on vCalendar.
- Sample:

```
BEGIN:VCALENDAR
VERSION:1.0
BEGIN:VEVENT
CATEGORIES:MEETING
STATUS:TENTATIVE
DTSTART:19960401T033000Z
DTEND:19960401T043000Z
SUMMARY:Your Proposal Review
DESCRIPTION:Steve and John to review newest proposal material
CLASS:PRIVATE
END:VEVENT END:
VCALENDAR
```

What is OBEX ?

- **OBEX** (abbreviation of **OBject EXchange**, also termed **IrOBEX**) is a communications protocol that facilitates the exchange of binary objects between devices.
- **Transports.** Like tcp-ip, is a basic transport protocol. Over an IrLAP/IrLMP/Tiny TP stack on an IrDA device.
- **Binary transmissions.** Uses binary-formatted type-length-value triplets called "Headers" to exchange information about a request or an object.

Byte 0	Bytes 1, 2	Bytes 3 to n
opcode	packet length	headers or request data

Byte 0	Bytes 1 and 2	Byte 3	Byte 4	Bytes 5 and 6	Byte 7 to n
0x80	connect packet length	OBEX version number	flags	maximum OBEX packet length	optional headers

- **Session support.** A single transport connection may bear many related operations.

Connection - OBEX

Client Request:	bytes	Meaning
opcode	0x80	**CONNECT**, Final bit set
	0x0007	7 bytes is length of packet
	0x10	version 1.0 of OBEX
	0x00	no connect flags
	0x2000	8K max packet size
Server Response:		
response code	0xA0	SUCCESS, Final bit set
	0x0007	packet length of 7
	0x10	version 1.0 of OBEX
	0x00	no connect flags
	0x0800	2K max packet size

Put - OBEX

Client Request:	**bytes**	**Meaning**
opcode	0x02	**PUT**, Final bit not set
	0x0422	1058 bytes is length of packet
	0x01	HI for **Name**
	0x0017	Length of **Name** header
	THING.DOC	name of object, null terminated
	0xC3	HI for **Length**
	0x00006000	Length of object is 0x6000 bytes
	0x48	HI for Object **Body** chunk
	0x0403	Length of **Body** header (1K) plus HI and header length
	0x.....	1K bytes of body
Server Response:		
response code	0x90	CONTINUE, Final bit set
	0x0003	length of response packet

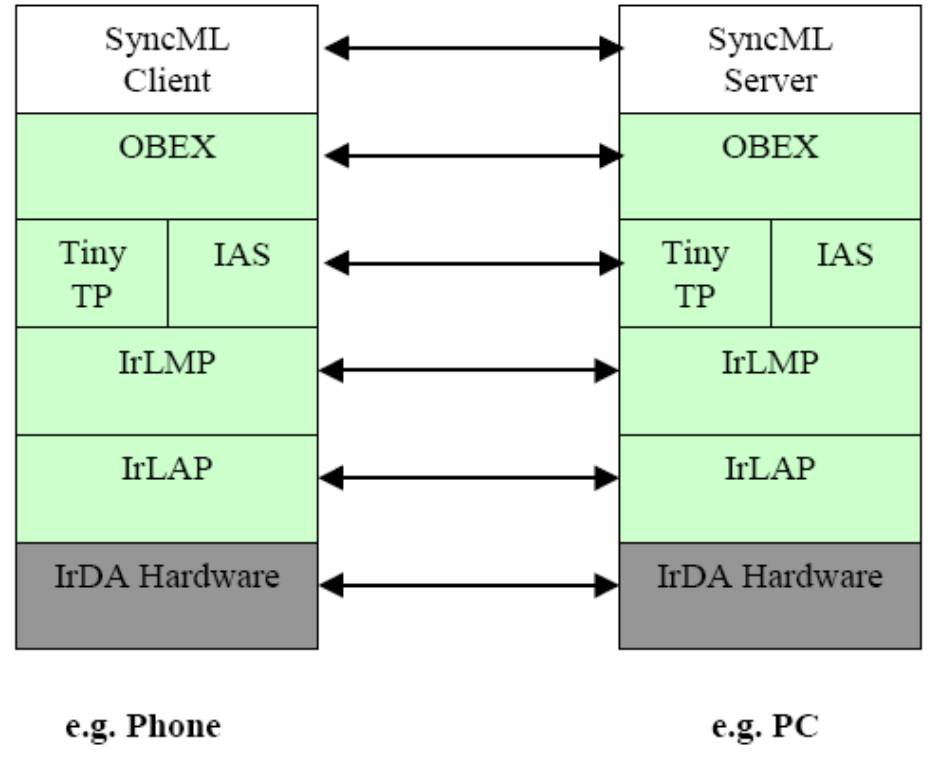
Put - OBEX

opcode	0x02	**PUT**, Final bit not set
	0x0406	1030 bytes is length of packet
	0x48	HI for Object **Body** chunk
	0x0403	Length of **Body** header (1K) plus HI and header length
	0x.....	next 1K bytes of body
Server Response:		
response code	0x90	CONTINUE, Final bit set
	0x0003	length of response packet
Client Request:		
opcode	0x82	**PUT**, Final bit set
	0x0406	1030 bytes is length of packet
	0x49	HI for **End-of-Body** chunk
	0x0403	Length of header (1K) plus HI and header length
	0x.....	next 1K bytes of body
Server Response:		
response code	0xA0	SUCCESS, Final bit sent
	0x0003	length of response packet

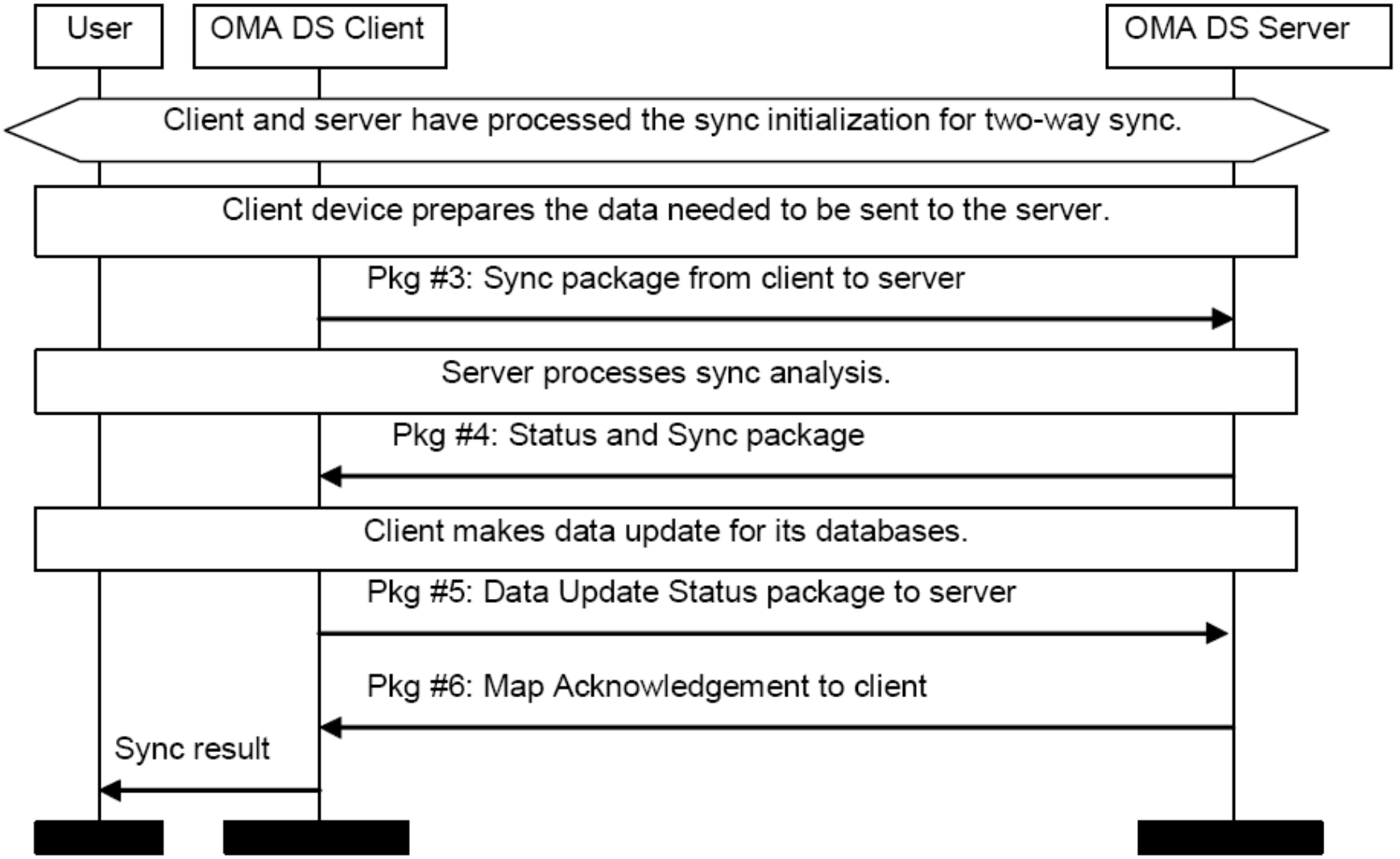
Get - OBEX

Client Request:	**bytes**	**Meaning**
Opcode	0x83	**GET**, Final bit set
	0x0003	length of **GET** packet
Server Response:		
Response code	0xA0	SUCCESS, Final bit set
	0x0038	length of response packet
	0x49	HI for **End-of-Body** chunk
	0x0035	Length of header
	0x.....	0x32 bytes of meter information

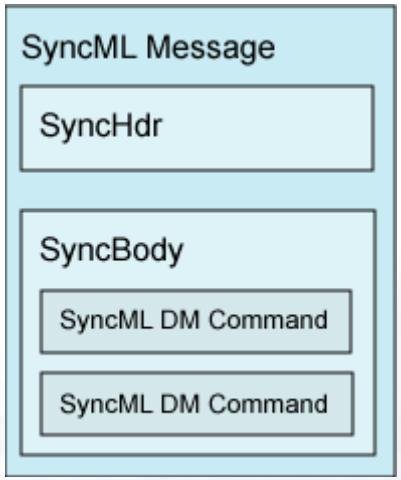
How to binding with OBEX ?



What is Two-way sync's flow ?



Sample : SyncML format




```
<SyncML xmlns='syncml:SYNCML1.1'>
<SyncHdr>
        <VerDTD>1.1</VerDTD>
        <VerProto>SyncML/1.1</VerProto>
        <SessionID>0</SessionID>
        <MsgID>0</MsgID>
        <Target><LocURI>/</LocURI></Target>
        <Source><LocURI>SyncMLDevice1</LocURI></Source>
        <Meta><MaxMsgSize xmlns='syncml:metinf'>10000</MaxMsgSize></Meta>
</SyncHdr>
<SyncBody>
        <Alert>
                <CmdID>1</CmdID>
                <Data>208</Data>
                <Item>
                        <Target><LocURI>events</LocURI></Target>
                        <Source><LocURI>events</LocURI></Source>
                        <Meta>
                                <Type xmlns='syncml:metinf'>text/x-vcalendar</Type>
                                <Anchor xmlns='syncml:metinf'>
                                <Next>0</Next>
                                <Last>1</Last>
                                </Anchor>
                        </Meta>
                </Item>
        </Alert>
        <Final/>
</SyncBody>
</SyncML>
```

Sample : DS server initiate a two way sync

```
<SyncML xmlns='syncml:SYNCML1.1'>
...
<SyncBody>
        <Alert>
                <CmdID>1</CmdID>
                <Data>206</Data>
                <Item>
                <Target>
                        <LocURI>events</LocURI>
                </Target>
                <Source>
                        <LocURI>events</LocURI>
                </Source>
                <Meta>
                        <Type xmlns='syncml:metinf'>text/x-vcalendar</Type>
                        <Anchor xmlns='syncml:metinf'>
                        <Next>0</Next>
                        <Last>1</Last>
                </Anchor>
                </Meta>
                </Item>
        </Alert>
<Final/>
</SyncBody>
</SyncML>
```

It means two-way sync by server.

It means sync for vCalendar.

It means last anchor and next anchor.

Sample : DS client replay for server's sync requirement

```
<Status>
        <CmdID>1001</CmdID>
        <MsgRef>0</MsgRef>
        <CmdRef>1</CmdRef>
        <Cmd>Alert</Cmd>
        <Data>200</Data>
        …
</Status>
<Alert xmlns='syncml:SYNCML1.1'>
        <CmdID>1002</CmdID>
        <Data>200</Data>
        <Item>
        <Target>
                <LocURI>events</LocURI>
        </Target>
        <Source>
                <LocURI>events</LocURI>
        </Source>
        <Meta>
                <Anchor xmlns='syncml:metinf'>
                <Last>62</Last>
                <Next>63</Next>
                </Anchor>
                <MaxObjSize>2048</MaxObjSize>
        </Meta>
        </Item>
</Alert>
```

It means successful.

It means two-way sycn by client.

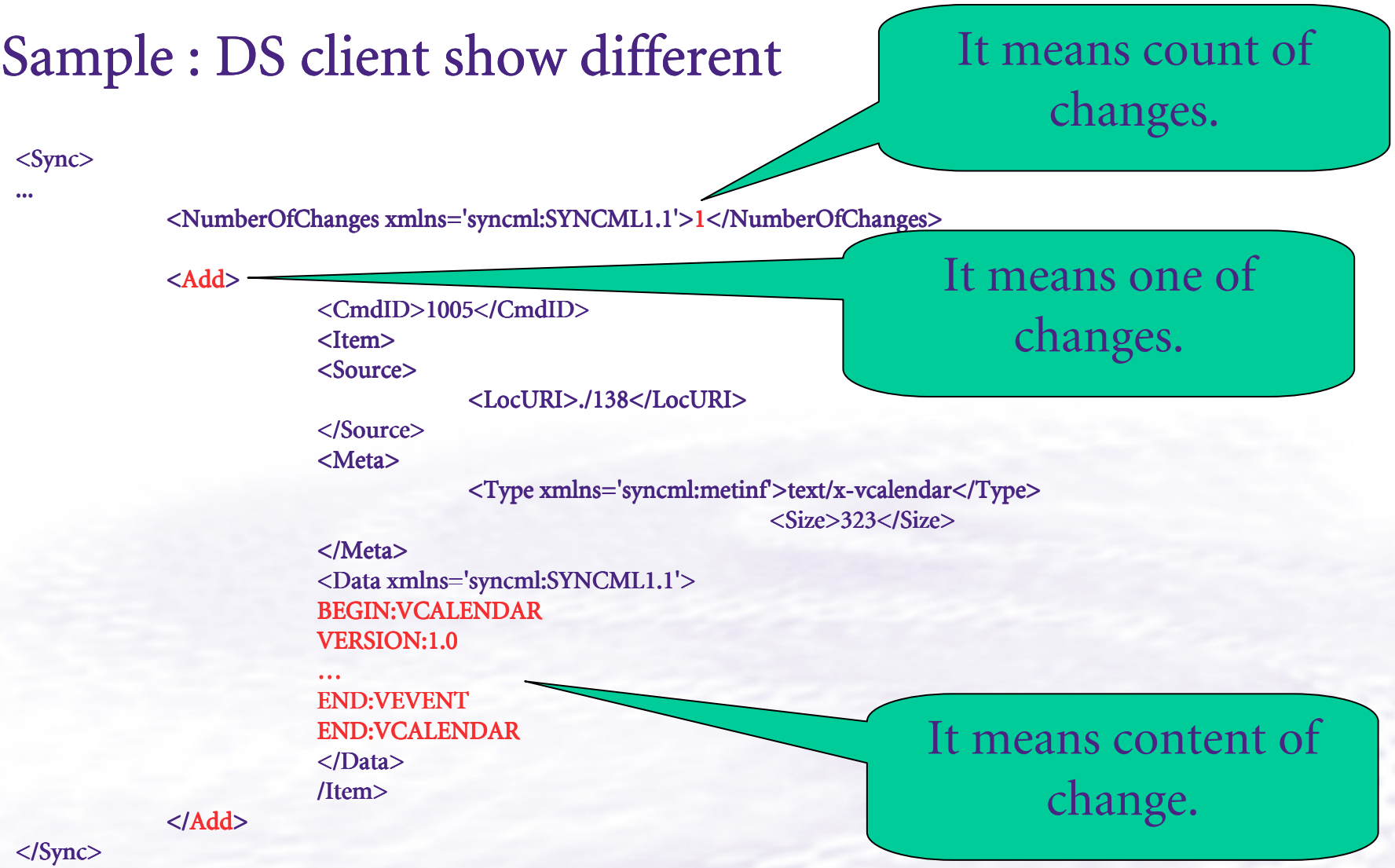
The maximum size of client package.

Sample : DS server replay for client's sync requirement

```
<Status>
        <CmdID>3</CmdID>
        <MsgRef>0</MsgRef>
        <CmdRef>1002</CmdRef>
        <Cmd>Alert</Cmd>
        <TargetRef>IMEI:004401480061544</TargetRef>
        <SourceRef>SyncMLDevice1</SourceRef>
        <Data>200</Data>
        <Item>
        <Data>
                <Anchor xmlns='syncml:metinf'>
                <Next>62</Next>
                </Anchor>
        </Data>
        </Item>
</Status>
```

It means successful.

Sample : DS client show different



Sample : DS server show different

```
<Sync>
...
        <NumberOfChanges xmlns='syncml:SYNCML1.1'>0</NumberOfChanges>
</Sync>
```

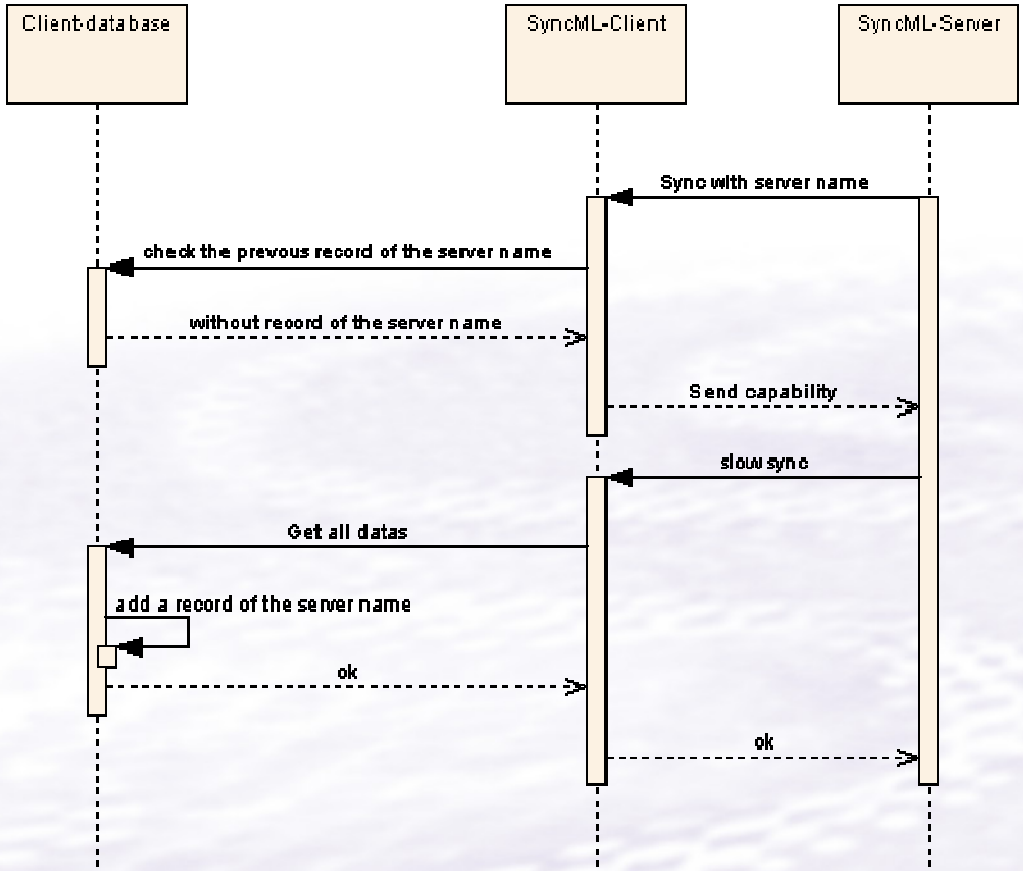
It means not any change.

Implementation

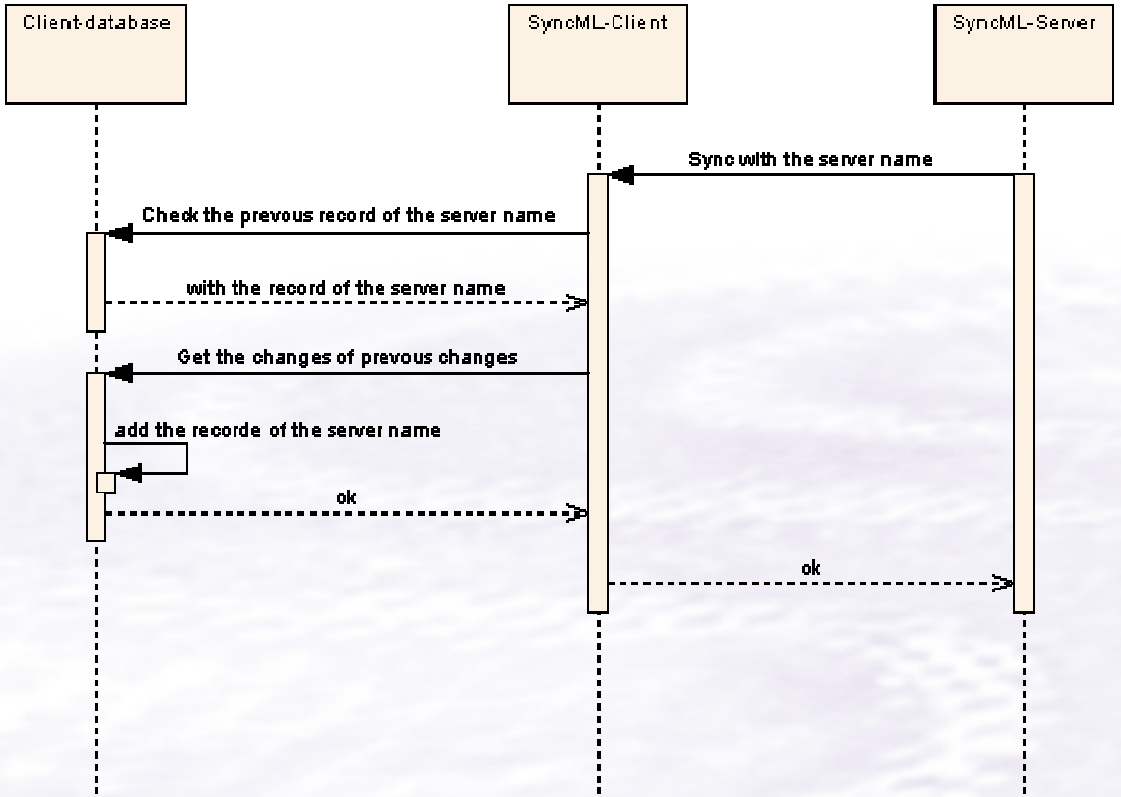
What the real case ?

How we need to do it ?

Detail behavior of client first sync



Detail behavior of client sync



Analysis of what we need

- Do almost the same behaviors of client sync
- Storage for record information of sync
  - Name of target
  - Log of changes
    - Add
    - Replace
    - Delete
  - Mapping table
  - Original data
- Transport data
  - By OBEX protocol *
- Logic
  - By SyncML protocol *

Mapping table

Client Device

Client Database:

LUID	Data
11	Car
22	Bike
33	Truck
44	Shoes

Server Device

Server Database:

GUID	Data
1010101	Car
2121212	Bike
3232323	Truck
4343434	Shoes

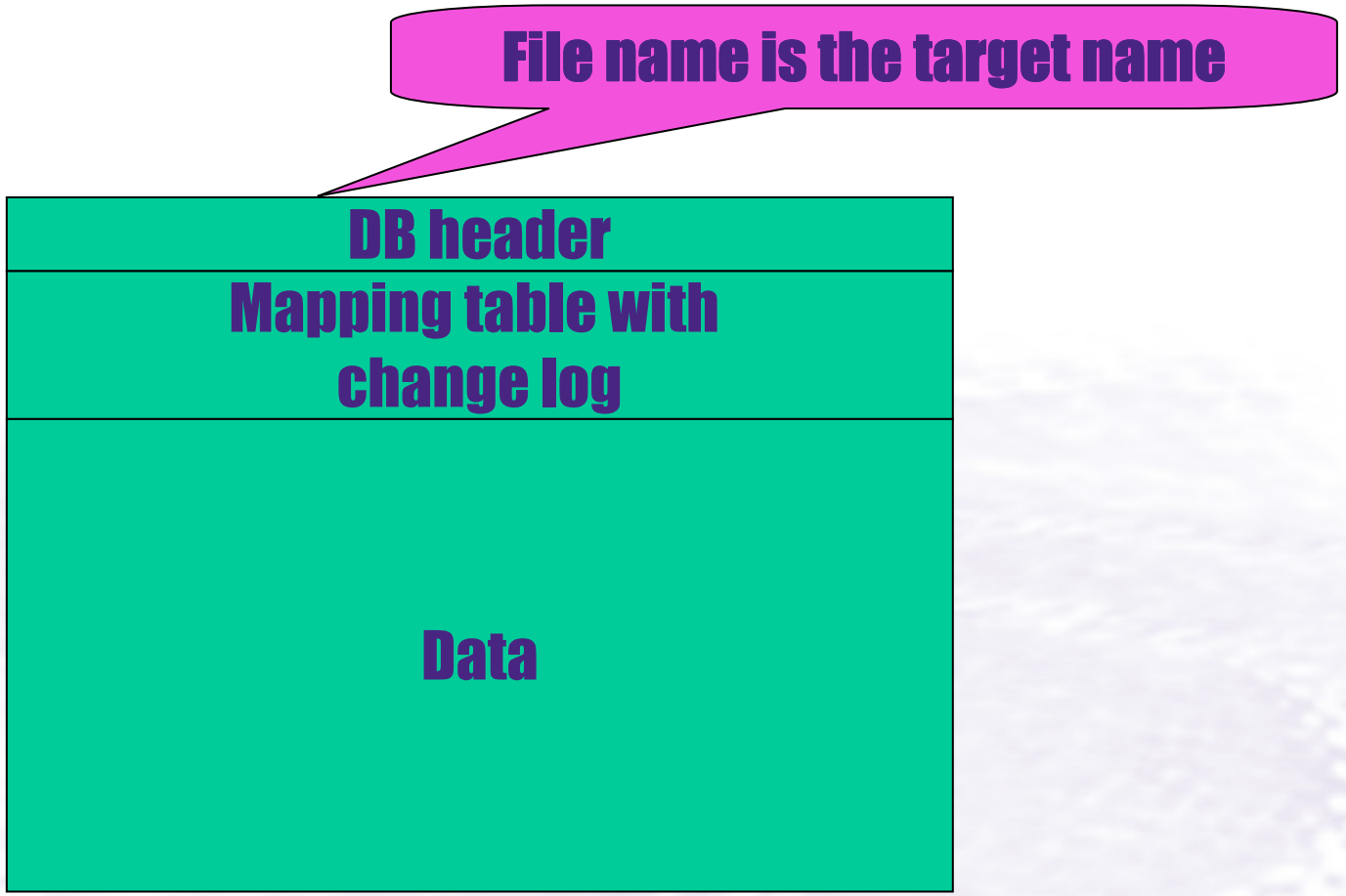
Server Mapping Table:

GUID	LUID
1010101	11
2121212	22
3232323	33
4343434	44

Merge change log and mapping table

Action	Change #	LUID
Mod	10	2
Mod	9	7456
Delete	8	34345
Mod	7	814
Mod	6	899
Mod	5	789
Mod	4	12345
Mod	3	34345
Mod	2	7456
Mod	1	567

Storage – File and file format



end