A generic database plugin for the c'man synchronization platform

Michael Kanonik

Supervisor: Dr. Ralf Schenkel

in cooperation with Consistec GmbH

31 October 2008
Outline

• Introduction
• SyncML Protocol
• Generic database plugin
• Conclusion
Introduction

• Data Synchronisation
  Process of establishing consistency among data on remote sources and harmonization of the data over time

• Reason for standard
  Existing synchronization solutions: vendor-, application- or operating system specific
Introduction

• **SyncML**: open platform-independent standard for data synchronization and device management.

• **SyncML Initiative** - a non-profit corporation formed by a group of major IT companies

• **c`man** (consistec mobile access node)
  synchronisation platform based on SyncML
SyncML Protocol

Synchronisation scenario (types) in c'man:

1) Two-way sync  
   Client and the Server exchange information about 
   modified data in these devices

2) One-way sync from Server only 
   Only the Client gets all modifications from the Server

SyncML have additionally:

3) Slow sync 
   All items are compared with each other
SyncML Protocol - Phases

• **Initialization**
  Interaction for transferring all modifications since a previous synchronization

• **Data Exchange**
  End a session properly
  Confirms that the synchronizing entities have received all the information

• **Completion**
  End a session properly
  Confirms that the synchronizing entities have received all the information
SyncML Protocol - Identifiers Mapping

Client devices: smaller capabilities

IDs to address data items:

- **Server**: globally unique identifiers **GUID**
  (lengths - typically in the range of 64–128 bytes)

- **Clients**: local unique identifier **LUID**
  (lengths <=16 bytes)
SyncML Protocol - ID Mapping

Client A

<table>
<thead>
<tr>
<th>LUID</th>
<th>DS obj</th>
</tr>
</thead>
</table>

Server

<table>
<thead>
<tr>
<th>GUID</th>
<th>DS obj</th>
</tr>
</thead>
<tbody>
<tr>
<td>10001</td>
<td>d o 1</td>
</tr>
<tr>
<td>10002</td>
<td>d o 2</td>
</tr>
</tbody>
</table>

- Pkg #0: Server Alert
- Pkg #1: Initialization package from Client
- Pkg #2: Initialization package from Server
- Pkg #3: Sync package from Client
SyncML Protocol - ID Mapping

Client A

<table>
<thead>
<tr>
<th>LUID</th>
<th>DS obj</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>do 1</td>
</tr>
<tr>
<td>2</td>
<td>do 2</td>
</tr>
</tbody>
</table>

Server

<table>
<thead>
<tr>
<th>GUID</th>
<th>DS obj</th>
</tr>
</thead>
<tbody>
<tr>
<td>10001</td>
<td>do 1</td>
</tr>
<tr>
<td>10002</td>
<td>do 2</td>
</tr>
</tbody>
</table>

Add do1, do2
SyncML Protocol - ID Mapping

<table>
<thead>
<tr>
<th>LUID</th>
<th>DS obj</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>d o 1</td>
</tr>
<tr>
<td>2</td>
<td>d o 2</td>
</tr>
</tbody>
</table>

Client A

Server

Status Add:OK
MAP(LUID,GUID)

GUID | DS obj |
-----|--------|
10001| d o 1  |
10002| d o 2  |
SyncML Protocol - ID Mapping

<table>
<thead>
<tr>
<th>LUID</th>
<th>DS obj</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>d o 1</td>
</tr>
<tr>
<td>2</td>
<td>d o 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GUID</th>
<th>DS obj</th>
</tr>
</thead>
<tbody>
<tr>
<td>10001</td>
<td>d o 1</td>
</tr>
<tr>
<td>10002</td>
<td>d o 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GUID</th>
<th>LUID of A</th>
</tr>
</thead>
<tbody>
<tr>
<td>10001</td>
<td>1</td>
</tr>
<tr>
<td>10002</td>
<td>2</td>
</tr>
</tbody>
</table>

Pkg #0: Server Alert
Pkg #1: Initialization package from Client
Pkg #2: Initialization package from Server
Pkg #3: Sync package from Client
Pkg #4: Sync package from Server
Pkg #5: Data mapping to Server
Pkg #6: Final Acknowledgment

Status MAP:OK
Goal of the thesis

• Extent of c'man synchronisation framework
• Compensate weakness of the SyncML protocol (inconsistent of DB)
• Implement configurable generic DB plugin
SyncML Protocol - ID Mapping Inconsistent

Client A

<table>
<thead>
<tr>
<th>LUID</th>
<th>DS obj</th>
</tr>
</thead>
<tbody>
<tr>
<td>1d01</td>
<td></td>
</tr>
<tr>
<td>2d02</td>
<td></td>
</tr>
</tbody>
</table>

Server

<table>
<thead>
<tr>
<th>GUID</th>
<th>DS obj</th>
</tr>
</thead>
<tbody>
<tr>
<td>10001</td>
<td>d01</td>
</tr>
<tr>
<td>10002</td>
<td>d02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GUID</th>
<th>LUID of A</th>
</tr>
</thead>
<tbody>
<tr>
<td>40004</td>
<td>4</td>
</tr>
<tr>
<td>40002</td>
<td>2</td>
</tr>
</tbody>
</table>

Pkg #0: Server Alert
Pkg #1: Initialization package from Client
Pkg #2: Initialization package from Server
Pkg #3: Sync package from Client
Pkg #4: Sync package from Server
Pkg #5: Data mapping to Server
Connection breaks
SyncML Protocol - ID Mapping Inconsistent

Client A

By next session
Client assigns new LUIDs

Problem: duplicates on the client
SyncML Protocol - ID Mapping Inconsistent

**Problem:** duplicates on the client

**Solution:**
1) Roll-back when no confirmation to Server
2) Use slow sync (is not implemented in c'man)
3) Drop Client DB and copy from Server
Outline

• Introduction
• SyncML Protocol
• Generic database plugin
• Conclusion
Generic Database plugin

Data Management part accessing and updating the actual DSs that are being synchronized with the Client.

**Datastore (DS) Adapter:**
1) converts a generic **sync object** into **datastore object** and vice versa
2) assists in ID Mapping
Generic Database plugin - Adaptor

Now:
Server adaptors are implemented manually for every application/DS

Problem:
For every DB scheme - need to add/change code manually to DS API, Marshaller and UnMarshaller
• a lot of code
• errors occur often
Generic Database plugin - Adaptor

Main class of the API:
abstract class ADatastore - encapsulates Datastore

• `modifyContent(AContentModification)`
  add, replace or delete the content according to Modification type

• `AContentModification getContentModification(String UNID)`
  Returns REPLACE modification object if there is an entry with the specified UNID; DELETE - otherwise

• `AContentModification[] getContentModificationsSince(Date date)`
  Returns all Items, that was modified since certain date
Generic Database plugin - Adaptor

**Goal:**
To generate a scheme independent adaptor per Datastore type

**Solution:**
Represent tables of relational DB in XML format
Conclusion

Problems:
• Weakness of the SyncML protocol (inconsistent of DB)
• Application-specific adapter

Goals of the thesis:
• Extension of c'man synchronisation framework
• Improving SyncML protocol (secure transaction)
• Configurable through XML data server plugin
End

Thank You!

Questions, remarks?
Generic Database plugin - SyncML Server

Generic SyncML Server implements

SyncML protocols:
• Representation
• Synchronisation
Appendix – SyncML Protocol
Appendix – SyncML Message

SyncML Message

SyncML Header

SyncML Body

SyncML Command

SyncML Command

SyncML Command
Appendix – Alert command in Initialization

```xml
<Alert>
  <CmdID>1</CmdID>

  Sync Type Definition
  <Data>200</Data>

  DB URI Definitions
  <Item>
    <Target><LocURI>./server_db</LocURI></Target>
    <Source><LocURI>./client_db</LocURI></Source>

    Meta
    <Anchor xmlns='syncml:metinf'>
    <Last>234</Last>
    <Next>276</Next>
    </Anchor>
  </Item>
</Alert>
```
Appendix – Levels of architecture

Client Application

Content (e.g., vCard)

SyncML Sync Protocol

HTTP

WSP

OBEX

Internet/Intranet

WAP/Wireless

Bluetooth IrDA

"App" Layer

Data Layer

Sync Layer

Transport Layer

"Physical" Layer

Corresponding Transport

Corresponding Physical Medium

Server App. / Sync Engine

Content

App DTDs

SyncML DTDs

Server App. / Sync Engine

Content

App DTDs