Open Data Sync: Open-Source Mobile Enterprise Data Synchronization

Hans Hrasna, Sr. Staff Engineer
Santiago Pericas-Geertsen, Staff Engineer
Ryan Shoemaker, Staff Engineer

Sun Microsystems, Inc.

TS-5947
Learn how to build a Mobile Enterprise Data Synchronization solution using Open Data Sync (ODS) and Mobile Enterprise Platform (MEP)
Agenda

- Introduction
- Deployment Scenarios
- Architecture
- Provisioning
- Security and Authentication
- Enterprise Connector Business Object API
- Mobile Client Business Object API
- NetBeans™ Tooling
- Live Demo
Introduction

**Needs**

- Move beyond simple email synchronization
- Key requirements
  - Prefer caching data on the device
  - Policy management
  - Device management
  - Security
- Control cost

**Challenges**

- Enterprise customers use multiple devices
- Enterprise solutions are proprietary ("locked in")
  - ERP must be delivered via proprietary methods
  - ERP applications/data are themselves also proprietary ("locked in")
Agenda

- Introduction
- Deployment Scenarios
- Architecture
- Provisioning
- Security and Authentication
- Enterprise Connector Business Object API
- Mobile Client Business Object API
- NetBeans Tooling
- Live Demo
Deployment – Mobile Provider Managed

CARRIER NETWORK

SyncML/HTTP(S)
Sun Fire T2000
Sun Java System MEP
Gateway Tier

Sun Fire T2000
Enterprise Information System

CORPORATE NETWORK

Sync DB
SOAP/HTTP(S)

Siebel
SAP
etc.

Sun Fire T2000
Sun Java System MEP
Enterprise Tier

Auth DB
Deployment – Enterprise Managed

CARRIER NETWORK

SyncML/HTTP(S)

Corporation

Sun Fire T2000
Enterprise Information System

Siebel SAP etc.

Sun Fire T2000
Sun Java System MEP Gateway Tier

Sync DB

PDA

Smart Phone

RIM Device

DB
Agenda

- Introduction
- Deployment Scenarios
- **Architecture**
- Provisioning
- Security and Authentication
- Enterprise Connector Business Object API
- Mobile Client Business Object API
- NetBeans Tooling
- Live Demo
Architecture – Design Goals

- Ease of use
- Use of standard technologies and protocols
- Rapid and flexible deployment
- Security suitable for enterprise deployment
- Modularity
- Easy to manage
Architecture – Components and Features
JavaME™ Client SDK

- Client Library that enables bi-directional synchronization of arbitrary data records based on OMA DS 1.1.2 and OMA DS 1.2 protocol
- Enables the development of device-agnostic clients
  - MIDP 2.0, CLDC 1.1, CDC 1.1.2
- Security
  - On-device data encryption
  - Authentication
  - TLS via HTTPS
  - Lock-out
  - Data destruction
Architecture – Components and Features

SyncML Engine

➢ Over-the-air data synchronization between client and server systems
  • Synchronize to any arbitrary enterprise backend data, e.g. Siebel, SAP, Oracle®

➢ Standard based – SyncML (OMA DS) protocol

➢ 100% Java implementation
Architecture – Components and Features
Universal Data Connector (UDC)

- Enables synchronization with proprietary server-side databases and applications
  - Synchronize to any arbitrary enterprise backend data, e.g. Siebel, SAP, Oracle

- Provides API for building JAX-WS based web service
- Based on JSR-170 Java Content Repository (JCR)
- Enterprise Connector Business Object API (ECBO) available for rapid connector development
Architecture

Mobile Device

Sync App

Sync Library

Client DB

SyncML/HTTP(S)

Server

Sync Engine

Sync DB

Connector

EIS

ODS & MEP Components
Application Components
Enterprise Data
APIs
Protocols

Proprietary

Client DB

Sync Library

Sync App

Mobile Device

SyncML/HTTP(S)

Server

Sync Engine

Sync DB

Connector

EIS

ODS & MEP Components
Application Components
Enterprise Data
APIs
Protocols

Proprietary
Architecture
Local Connectors Using JCR

- Sync DB
- JPA
- Sync Engine
- JCR
- Single Tier
- Siebel
- SAP
- Oracle
- EIS
- Proprietary
- Sun JCA Adapter
- Connector Library
- JCA
- ECBO
- JavaOne SM Conference | java.sun.com/javaone | 2008
Architecture
Remote Connectors Using SOAP

Gateway Tier
- JPA
- Sync DB
- Sync Engine
- JCR
- WS Connector
- SOAP/HTTP(S)

Enterprise Tier
- Connector
- Connector Library
- Sun JCA Adapter
- JCA
- ECBO
- Siebel
- SAP
- Oracle
- EIS
- Proprietary
Agenda

- Introduction
- Deployment Scenarios
- Architecture
- **Provisioning**
- Security and Authentication
- Enterprise Connector Business Object API
- Mobile Client Business Object API
- NetBeans Tooling
- Live Demo
MEP Provisioning Portal

- Based on JSR-124 Java EE Client Provisioning
- OTA discovery/provisioning
- Integrated content repository
- Controlled access to applications
- Provisioning based on device capability
- Content Management
- Content Versioning
Agenda

- Introduction
- Deployment Scenarios
- Architecture
- Provisioning
- Security and Authentication
- Enterprise Connector Business Object API
- Mobile Client Business Object API
- NetBeans Tooling
- Live Demo
Authentication

Enterprise Managed Deployment
- User/pass authentication directly with back-end
- No need to store passwords in Mobile Gateway

Mobile Provider Managed Deployment
- User/pass authentication with Mobile Gateway
- Mobile Gateway server authenticates with Web Service server
- Mobile user mapped to Enterprise user in Web Service server
- Two password domains: Gateway server and Web Service server
Mobile Client Security

➢ Client Application Authentication
  • `syncml:auth-basic` / `syncml:auth-md5` over `https`

➢ Transport Layer Security via HTTPS

➢ On-Device Data Encryption
  • Alphanumeric PIN based encryption scheme
  • MD5 digest of pin used as encryption key
  • data encrypted with triple DES
Agenda

- Introduction
- Deployment Scenarios
- Architecture
- Provisioning
- Security and Authentication
- Enterprise Connector Business Object API
- Mobile Client Business Object API
- NetBeans Tooling
- Live Demo
Enterprise Connector Business Object API

*Provide a framework to develop connectors that can perform CRUD operations on back-end systems*
Connector Architecture

- JCA API
- Other Java APIs
- Enterprise Connector BO API
- JCR API
- JavaEE
Enterprise Connector Business Object API

- Supports synchronization of arbitrary Java objects
- Each object instance is identified by a name
- Object serialization of your choice
  - Java serialization, XML, JavaScript™ Object Notation (JSON), etc.
  - Contract between mobile application and connector
- Hides JCR complexity under simple API
  - Typical connector is only 5 classes
- Can use JCA to access Sun JCA Adapters
Business Object Abstraction

- Business Objects are the entities being synchronized
- They are serialized into data records
  - Data records are encoded in base64 in SyncML
- Implement CUD operations with back-end systems
  - With Sun JCA Adapters: Siebel, SAP, Oracle Apps, etc.
  - Without Sun JCA Adapters: JDBC, web services, file system, etc.
BusinessObject Class

**BusinessObject**

(From api)

Attributes

- protected String name

Operations

- public String getName()
- public void setName(String name)
- public String getExtension()
- public byte[] serialize()
- public void deserialize(byte array[])
- public InsertCommand getInsertCommand()
- public UpdateCommand getUpdateCommand()
- public DeleteCommand getDeleteCommand()

**Command**

(From api)

Attributes

Operations

- public Command(BusinessObject bobject)
- public Object execute()

**UpdateCommand**

(From api)

Attributes

Operations

- public UpdateCommand(BusinessObject bobject)
- Operations Redefined From Command
- public Object execute()

**InsertCommand**

(From api)

Attributes

Operations

- public InsertCommand(BusinessObject bobject)
- Operations Redefined From Command
- public Object execute()
class MusicAlbum extends BusinessObject {
  String name;
  ...
  @Override
  public getDeleteCommand() {
    return
      new MusicAlbumDeleteCommand(this,
        getSQLConnection(),
        
                "DELETE FROM album WHERE " +
              "name = '" + name + "'");
  }
}
Business Object Provider Abstraction

- A connector's entry point
  - Life cycle methods to initialize and terminate
- Factory for Business Objects
- Provides access to session context
- Implements the R in CRUD
  - Single operation to retrieve list of Business Objects
BusinessObjectProvider Class

```java
BusinessObjectProvider
    { From api }
    
    Attributes
    
    Operations
    public void initialize( )
    public void terminate( )
    public SessionContext getSessionContext( )
    public void setSessionContext( SessionContext context )
    public String getRepositoryName( )
    public BusinessObject[0..*] getBusinessObjects( )
    public BusinessObject newBusinessObject( )
```

```java
SessionContext
    { From api }
    
    Attributes
    private String username
    private String password

    Operations
    public SessionContext( String username, String password )
    public Map<String, String> getParameters( )
    public void setParameter( String name, String value )
    public String getUsername( )
    public String getPassword( )
```

```java
BusinessObject
    { From api }
    
    Attributes
    protected String name

    Operations
    public String getName( )
    public void setName( String name )
    public String getExtension( )
    public byte[] serialize( )
    public void deserialize( byte array[0..*] )
    public InsertCommand getInsertCommand( )
    public UpdateCommand getUpdateCommand( )
    public DeleteCommand getDeleteCommand( )
```
MusicAlbumProvider Sample Class

class MusicAlbumProvider extends BusinessObjectProvider {
    Connection conn;
    ...
    @Override
    public List<? extends BusinessObject> getBusinessObjects()
    {
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT * from ALBUM");
        return resultSetToList(rs);
    }
}
Agenda

- Introduction
- Deployment Scenarios
- Architecture
- Provisioning
- Security and Authentication
- Enterprise Connector Business Object API
- Mobile Client Business Object API
- NetBeans Tooling
- Live Demo
Mobile Client Business Object API

Provide a framework for developing Java ME applications capable of synchronizing business data with back-end systems
Mobile Client Business Object API

- **Java ME library**
  - Allows bi-direction synchronization of arbitrary (user-defined) data types with corporate back-end systems:
    - Siebel CRM, SAP, Oracle Apps, JDBC, etc.
    - Application defines data model and serialized form
      - POJOs that extend `BusinessObject`

- **Implementation details**
  - Built on pure Java SyncML implementation (OMA DS)
  - Supports XML and WAP Binary XML (WBXML) encoding
  - SyncML protocol over http/s

- **Device requirements**
  - CDC 1.1.2 / CLDC 1.1 / MIDP 2.0
  - JSR-75 (for accessing mobile device filesystems)
Mobile Client Business Object API Cont.

- Six types of client initiated syncs
  - **Client ↔ Server**
    - Two way sync (aka fast sync): client initiated sync of modified data
    - Slow sync: full field-by-field sync with the server
  - **Client → Server**
    - One way sync from client: client modifications sent to server
    - Refresh sync from client: client data replaces server data
  - **Server → Client**
    - One way sync from server: server modifications sent to client
    - Refresh sync from server: server data replaces client data

- Server initiated syncs
  - Server can initiate syncs via SMS messages to the device
Architecture

JavaME Application

Mobile Client Business Object API

OpenDataSync (ODS)

SyncML (OMA DS)

MIDP 2.0

JSR-75 Filesystem

CLDC 1.1
Architecture Cont.

SyncType

Attributes
- private String syncType

Operations
- private SyncType(String type)
- public String getValue()

EncodingType

Attributes
- private String encodingType

Operations
- private EncodingType(String type)
- public String getValue()

SyncResults

Attributes
- private String applicationURI = "files"

Operations
- package SyncResults(EditItemUseCase eiuc)
- public int getClientAdditions()
- public int getClientDeletions()
- public int getServerAdditions()
- public int getServerUpdates()
- public int getServerDeletions()

SyncException

Attributes
- private Throwable linkedException

Operations
- public SyncException(String pMessage)
- private String formatMessage(String pMessage, Throwable pThru)
- public SyncException(Throwable pLinkedException)
- public SyncException(String pMessage, Throwable pLinkedException)
- public Throwable getLinkedException()
- public void setLinkedException(Throwable pLinkedException)
- public void printStackTrace()
Mobile Client Business Object API Overview

- **BusinessObject** defines your data model and the serialized form used to store the data on the device
  - Application developer must implement:
    - bean properties for your data model
    - `abstract void deserialize(byte[] data)`
    - `abstract byte[] serialize()`

- **BusinessObjectStorage** manages storing and retrieving instances of **BusinessObject** on the device

- **SyncManager** controls synchronization with gateway

- **SyncType** and **EncodingType** enumerate synchronization types and encoding types

- **SyncResults** provides coarse-grained stats after syncs
Client Application Code Sample

01: // JavaME UI code stripped away...
02:
03: SyncManager syncMgr = new SyncManager(“alb”);
04: BusinessObjectStorage boStore = new BusinessObjectStorage(“alb”);
05:
06: syncMgr.setCredentials(
07:     “username”, “password”,
08:     “https://syncserver.com/sync”);
09:
10: // Modify some local BusinessObjects...
11:
12: syncMgr.sync(SyncType.FAST_SYNC); // two way sync
13:
14: SyncResults results = syncMgr.getSyncResults();
Business Object Code Sample

public class Album extends BusinessObject {

    // bean properties
    public String getName() {...}
    public void setName(String name) {...}

    public String getArtist() {...}
    public void setArtist(String artist) {...}

    public int getRating() {...}
    public void setRating(int rating) {...}

    public Date getDatePublished() {...}
    public void setDatePublished(Date date) {...}
// serialize this BusinessObject to a byte array
public byte[] serialize() throws IOException {
    ByteArrayOutputStream out
        = new ByteArrayOutputStream();
    DataOutputStream dOut
        = new DataOutputStream(out);

    // some error checking removed for clarity
    dOut.writeUTF(getName());
    dOut.writeUTF(artist);
    dOut.writeUTF(datePublished);
    dOut.writeUTF(Integer.toString(rating));
    dOut.flush();

    return out.toByteArray();
}
/ deserialized the byte array into this BusinessObject
public void deserialize(byte[] data) throws IOException { 
    ByteArrayInputStream in
        = new ByteArrayInputStream(data);
    DataInputStream dIn
        = new DataInputStream(in);

    // error checking removed for clarity
    albumName = dIn.readUTF();
    artist = dIn.readUTF();
    String date = dIn.readUTF();
    datePublished = ...; // parse 'date' string
    rating = Integer.parseInt(dIn.readUTF());
}
Agenda

- Introduction
- Deployment Scenarios
- Architecture
- Provisioning
- Security and Authentication
- Mobile Client Business Object API
- Enterprise Connector Business Object API
- NetBeans Tooling
- Live Demo
Netbeans Tooling

- Enterprise connector development
  - Modules to generate Object Type Definitions (OTDs)
  - OTDs are object models specific to each back-end
    - E.g., OTDs for Siebel Account Objects
  - Enterprise Connector Business Object library

- JavaME client development
  - NetBeans 6.1 w/ Mobility Pack
  - Full integration with JavaME Wireless Toolkit
  - Choice of UI framework: (LWUIT, LCD UI, SVG, etc)
  - Mobile Business Object library
Agenda

- Introduction
- Deployment Scenarios
- Architecture
- Provisioning
- Security and Authentication
- Mobile Client Business Object API
- Enterprise Connector Business Object API
- NetBeans Tooling
- Live Demo
MusicDB
A simple album catalog app that syncs with a JDBC back-end....
About the demo...

➢ MusicDB Connector
  • Connector uses JDBC to connect to back-end DB
  • Only 5 Java classes:
    • MusicAlbum extends BusinessObject
    • MusicAlbumProvider extends BusinessObjectProvider
    • One sub-class for each command

➢ MusicDB Client
  • Only 1 data model class:
    • Album extends BusinessObject
  • Uses Triple DES for on-device data encryption
  • TLS provided by HTTPS
  • Application code + MCBO lib + encryption lib = Only 190K after obfuscation
Summary

- ODS - a platform for building mobile enterprise solutions
- Based on open standards
- Business Object APIs hide all complexity
- Open source at https://opendatasync.dev.java.net
- Available soon
THANK YOU

Hans,Hrasna@Sun.com
Santiago.PericasGeertsen@Sun.com
Ryan.Shoemaker@Sun.com

TS-5947