Modern web-based IMS application architectures

Robert Recknagel

robert.recknagel@outlook.de

March 18, 2015

IMS Technical Symposium 2015

Agenda

Introduction

The sample application

IMS Catalog

IMS Open Database

The 3-tier ATM application

Java in IMS dependent regions

IMS Enterprise Suite Connect API

The 4-tier ATM application

Live Demo

Summary

Questions?

Introduction

Importance of IMS modernization

Agenda

Introduction

The sample application

IMS Catalog

IMS Open Database

The 3-tier ATM application

Java in IMS dependent regions

IMS Enterprise Suite Connect API

The 4-tier ATM application

Live Demo

Summary

Questions?

Future trends in application design

- Current situation:
  - Access through 3270 is slowly dying out
  - Decreasing number of new fat client applications
  - More and more web-based applications
  - Growing demand for mobile accessibility
- Most future applications will have a front-end based on web technologies (HTML 5, CSS, JavaScript, …)
  - This makes the front-end platform independent
  - No deployment to workstations is needed for those applications
  - Many mobile apps do also use these technologies (even if they behave like native apps)

Future trends in application design

- Behind the front-end an application server will execute major parts of the application’s functionality
  - The functionality will be provided by small software components (Java Servlets, EJBs, …)
  - The implementation of these components is transparent to the front-end
  - More and more different back-end systems have to be accessed through these components
- Back-end systems will run on different hardware under different operating systems (z/OS, UNIX, Linux, Windows Server, …)
  - Easy access to these back-end systems is needed
  - This includes not only the data stored under control of these back-end systems but also existing applications running under these back-end systems

➔ So there is a demand for an easy integration of both IMS databases and IMS transactions into new application architectures

Modernizing IMS applications

- In the future it will also become more and more important to modernize existing IMS applications because
  - The programmers of those applications will be retired
  - Many of the older applications are not documented, so their exact functionality is hard to understand for others
  - Young computer scientists do not have Assembler, COBOL or PL/I skills, but they will have Java skills
  - It is still the same with DL/I and SQL

Modernizing IMS applications

- You all should be open-minded for new technologies because
  - The mainframe is no stand-alone system anymore
  - Using similar technologies from front-end to back-end enables the possibility to understand the whole application functionality
  - New technologies also have important pros (for instance: SQL is set-oriented, so less coding is needed for the same functionality)
  - These pros may compensate the higher resource usage of these technologies
  - Using new technologies may change the view of managers on IMS and the mainframe

The sample application

Two examples of modern multi-tier IMS application architectures

Agenda

Introduction

The sample application

IMS Catalog

IMS Open Database

The 3-tier ATM application

Java in IMS dependent regions

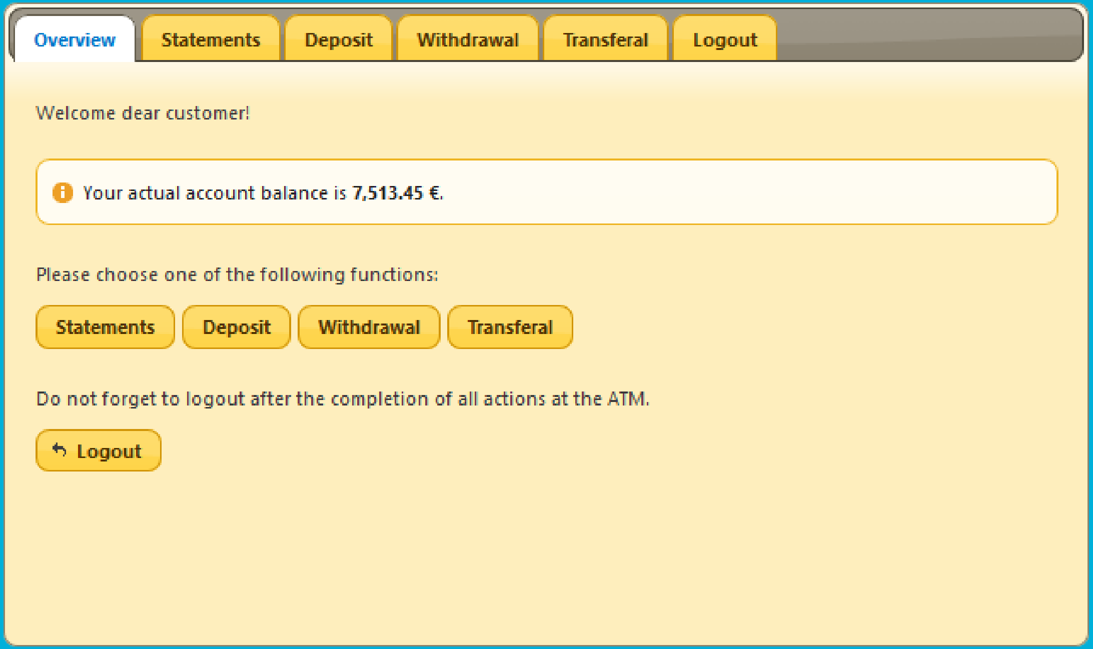
IMS Enterprise Suite Connect API

The 4-tier ATM application

Live Demo

Summary

Questions?



Welcome dear customer!

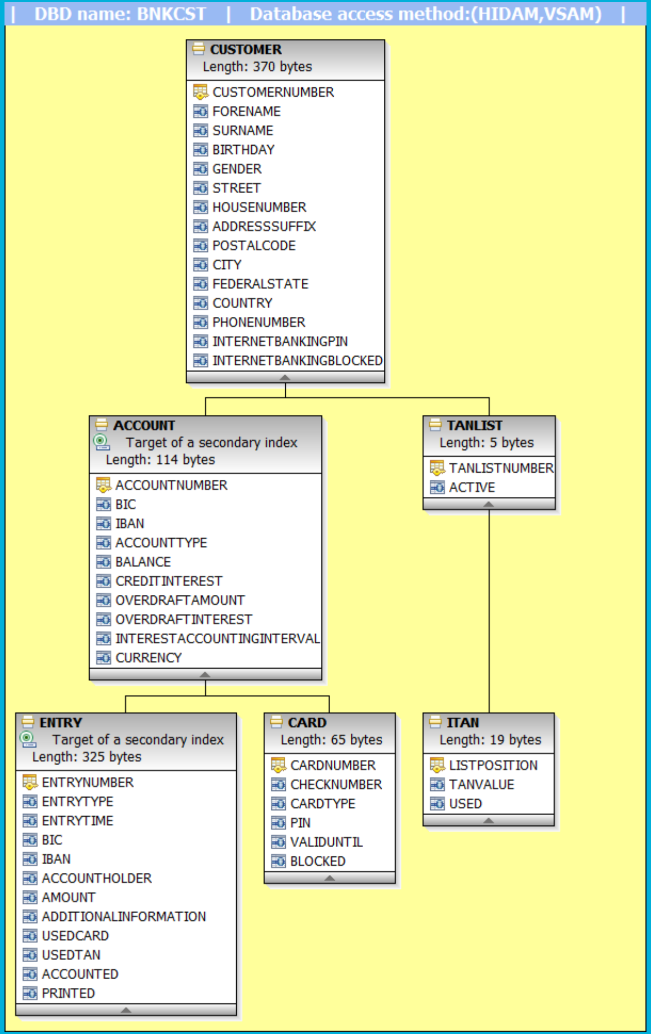
Your actual account balance is **7,513.45 €.**

Please choose one of the following functions:

Do not forget to logout after the completion of all actions at the ATM.

The sample application simulates the functionality of an **ATM**.

Both implementations have the same web-based front-end and the same IMS database as back-end. The layers between are different to each other.



| DBD name: BNKCST | Database access method:(HIDAM,VSAM) |

CUSTOMER
Length: 370 bytes

- CUSTOMERNUMBER
- FORENAME
- SURNAME
- BIRTHDAY
- GENDER
- STREET
- HOUSENUMBER
- ADDRESSSUFFIX
- POSTALCODE
- CITY
- FEDERALSTATE
- COUNTRY
- PHONENUMBER
- INTERNETBANKINGPIN
- INTERNETBANKINGBLOCKED

ACCOUNT
Target of a secondary index
Length: 114 bytes

- ACCOUNTNUMBER
- BIC
- IBAN
- ACCOUNTTYPE
- BALANCE
- CREDITINTEREST
- OVERDRAFTAMOUNT
- OVERDRAFTINTEREST
- INTERESTACCOUNTINGINTERVAL
- CURRENCY

TANLIST
Length: 5 bytes

- TANLISTNUMBER
- ACTIVE

ENTRY
Target of a secondary index
Length: 325 bytes

- ENTRYNUMBER
- ENTRYTYPE
- ENTRYTIME
- BIC
- IBAN
- ACCOUNTHOLDER
- AMOUNT
- ADDITIONALINFORMATION
- USEDCARD
- USEDTAN
- ACCOUNTED
- PRINTED

CARD
Length: 65 bytes

- CARDNUMBER
- CHECKNUMBER
- CARDTYPE
- PIN
- VALIDUNTIL
- BLOCKED

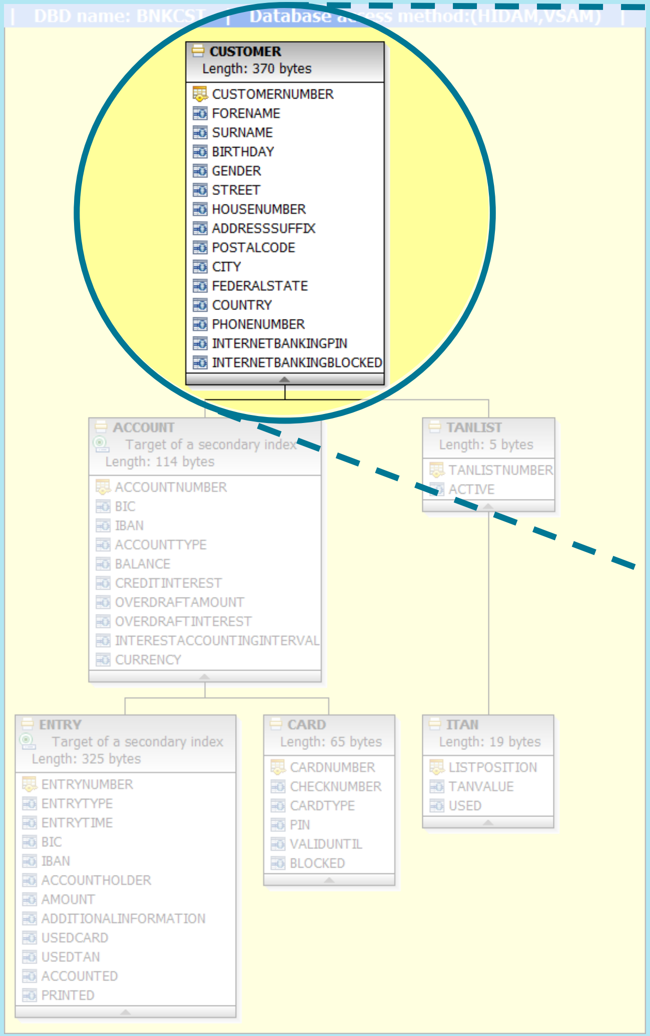
ITAN
Length: 19 bytes

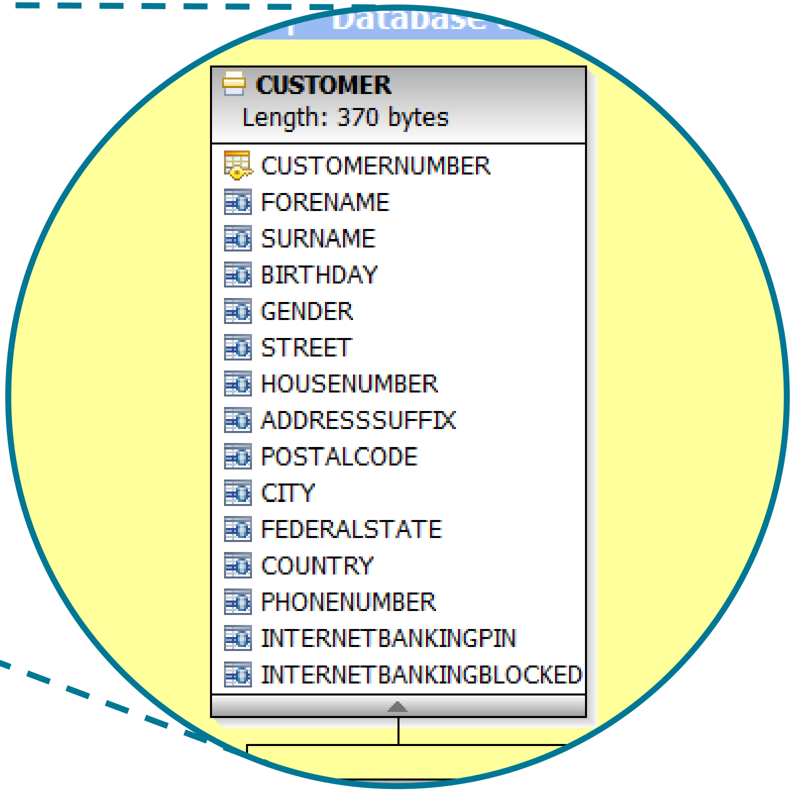
- LISTPOSITION
- TANVALUE
- USED

The data the ATM application accesses is stored in the **bank customer database**.

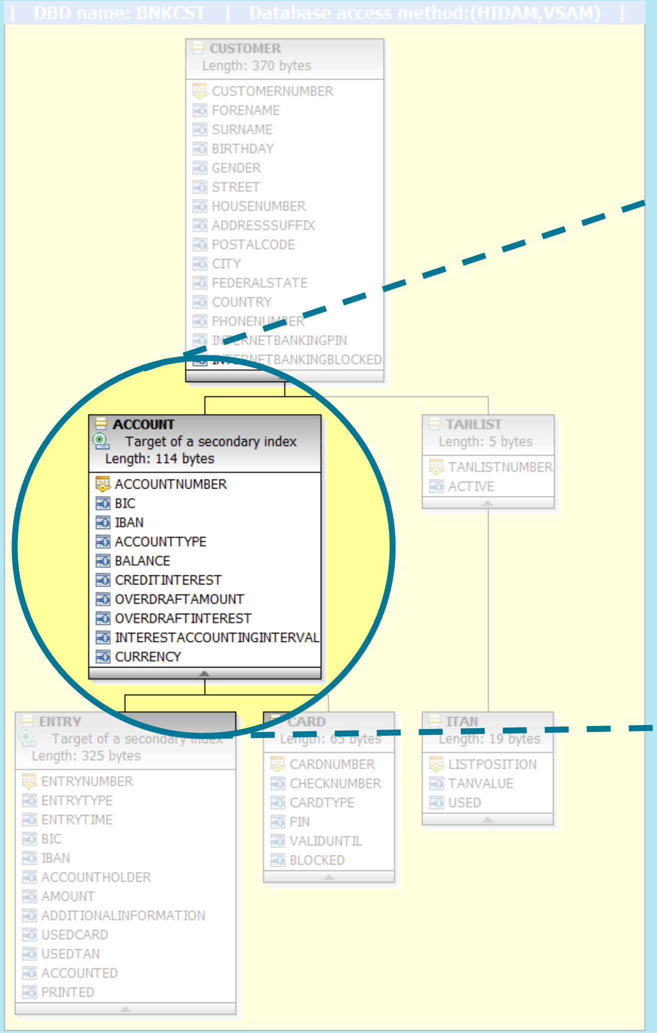
- HIDAM database with three secondary indices for faster SQL access
- No undefined space in the database segments
- Usage of meaningful segment and field names defined by the EXTERNALNAME attribute in the DBD

The ATM application uses only parts of the database and accesses the database always through a secondary index.





The root segment stores key information about the **bank customer** like name, birthday, address and phone number.



| DBD name: BNKCST | Database access method:(HIDAM,VSAM) |

CUSTOMER
Length: 370 bytes

- CUSTOMERNUMBER
- FORENAME
- SURNAME
- BIRTHDAY
- GENDER
- STREET
- HOUSENUMBER
- ADDRESSSUFFIX
- POSTALCODE
- CITY
- FEDERALSTATE
- COUNTRY
- PHONENUMBER
- INTERNETBANKINGPIN
- INTERNETBANKINGBLOCKED

ACCOUNT
Target of a secondary index
Length: 114 bytes

- ACCOUNTNUMBER
- BIC
- IBAN
- ACCOUNTTYPE
- BALANCE
- CREDITINTEREST
- OVERDRAFTAMOUNT
- OVERDRAFTINTEREST
- INTERESTACCOUNTINGINTERVAL
- CURRENCY

TANLIST
Length: 5 bytes

- TANLISTNUMBER
- ACTIVE

ENTRY
Target of a secondary index
Length: 325 bytes

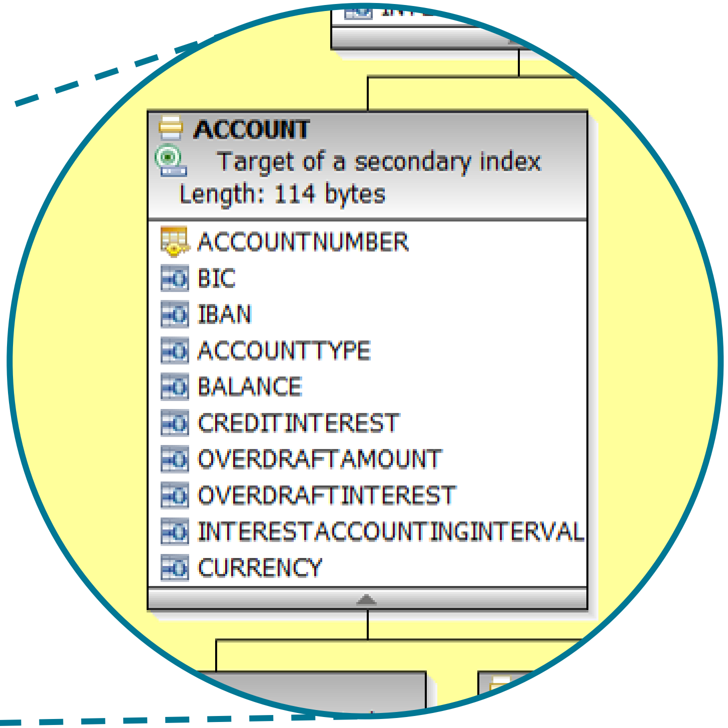
- ENTRYNUMBER
- ENTRYTYPE
- ENTRYTIME
- BIC
- IBAN
- ACCOUNTHOLDER
- AMOUNT
- ADDITIONALINFORMATION
- USEDCARD
- USEDTAN
- ACCOUNTED
- PRINTED

CARD
Length: 65 bytes

- CARDNUMBER
- CHECKNUMBER
- CARDTYPE
- PIN
- VALIDUNTIL
- BLOCKED

ITAN
Length: 19 bytes

- LISTPOSITION
- TANVALUE
- USED

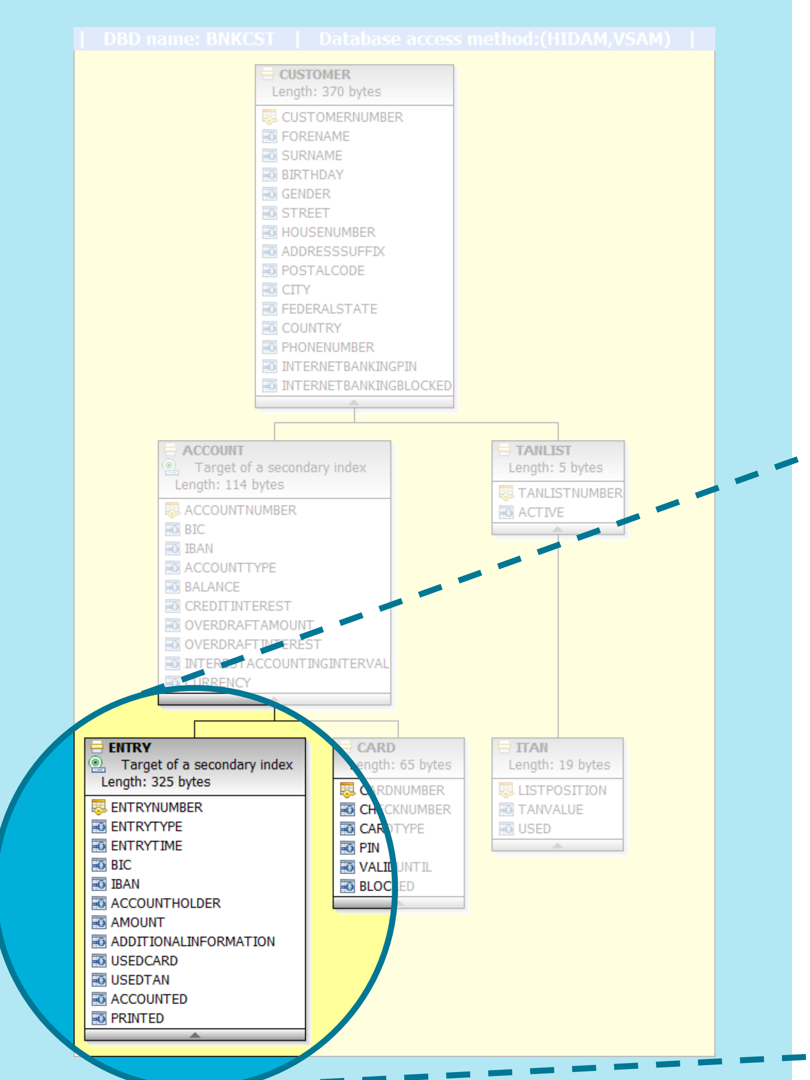


ACCOUNT
Target of a secondary index
Length: 114 bytes

- ACCOUNTNUMBER
- BIC
- IBAN
- ACCOUNTTYPE
- BALANCE
- CREDITINTEREST
- OVERDRAFTAMOUNT
- OVERDRAFTINTEREST
- INTERESTACCOUNTINGINTERVAL
- CURRENCY

The ATM application gets in the database at the **account** segment using a secondary index on the account number field. This segment mainly stores balance and interest information.

The **entry** segments contain information about the transactions of an account. Another secondary index is defined on this segment.



| DBD name: BNKCST | Database access method:(HIDAM,VSAM) |

CUSTOMER
Length: 370 bytes
- CUSTOMERNUMBER
- FORENAME
- SURNAME
- BIRTHDAY
- GENDER
- STREET
- HOUSENUMBER
- ADDRESSSUFFIX
- POSTALCODE
- CITY
- FEDERALSTATE
- COUNTRY
- PHONENUMBER
- INTERNETBANKINGPIN
- INTERNETBANKINGBLOCKED

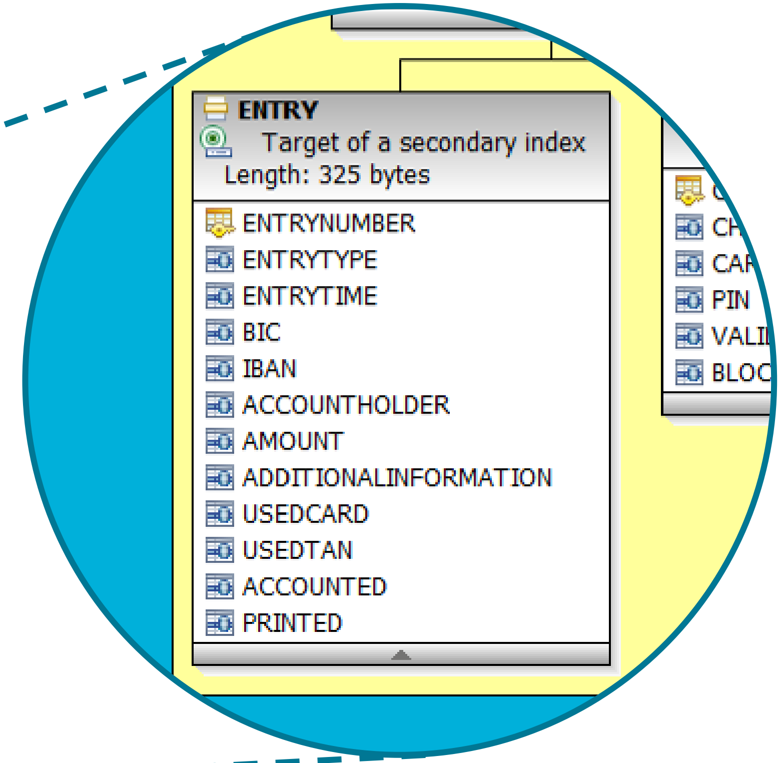
ACCOUNT
Target of a secondary index
Length: 114 bytes
- ACCOUNTNUMBER
- BIC
- IBAN
- ACCOUNTTYPE
- BALANCE
- CREDITINTEREST
- OVERDRAFTAMOUNT
- OVERDRAFTINTEREST
- INTERESTACCOUNTINGINTERVAL
- CURRENCY

TANLIST
Length: 5 bytes
- TANLISTNUMBER
- ACTIVE

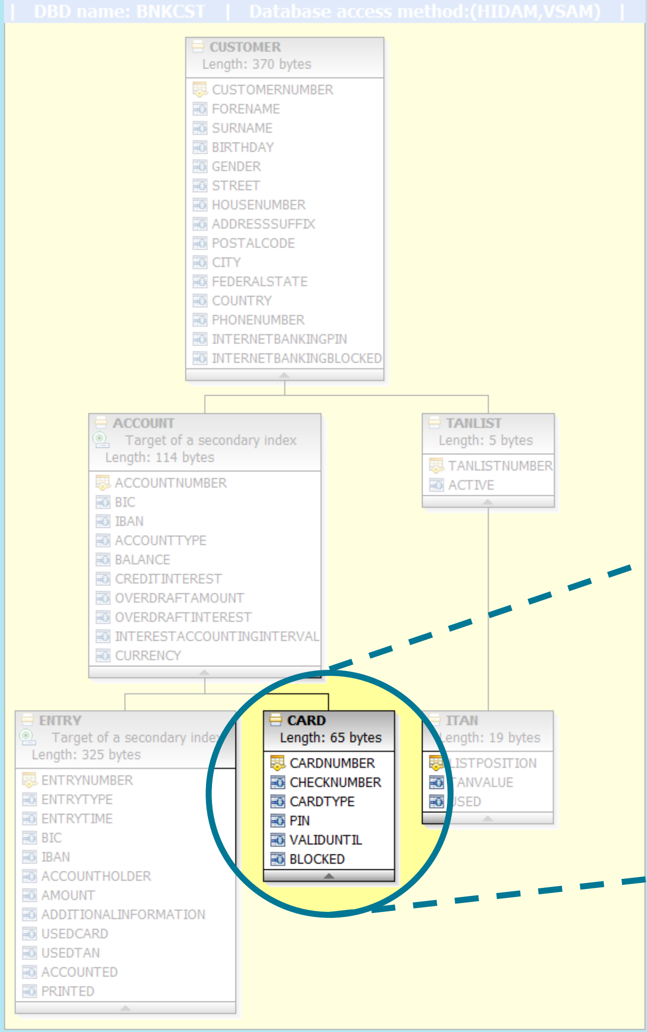
ENTRY
Target of a secondary index
Length: 325 bytes
- ENTRYNUMBER
- ENTRYTYPE
- ENTRYTIME
- BIC
- IBAN
- ACCOUNTHOLDER
- AMOUNT
- ADDITIONALINFORMATION
- USEDCARD
- USEDTAN
- ACCOUNTED
- PRINTED

CARD
Length: 65 bytes
- CARDNUMBER
- CHECKNUMBER
- CARDTYPE
- PIN
- VALIDUNTIL
- BLOCKED

ITAN
Length: 19 bytes
- LISTPOSITION
- TANVALUE
- USED



ENTRY
Target of a secondary index
Length: 325 bytes
- ENTRYNUMBER
- ENTRYTYPE
- ENTRYTIME
- BIC
- IBAN
- ACCOUNTHOLDER
- AMOUNT
- ADDITIONALINFORMATION
- USEDCARD
- USEDTAN
- ACCOUNTED
- PRINTED



| DBD name: BNKCST | Database access method:(HIDAM,VSAM) |

CUSTOMER
Length: 370 bytes
CUSTOMERNUMBER
FORENAME
SURNAME
BIRTHDAY
GENDER
STREET
HOUSENUMBER
ADDRESSSUFFIX
POSTALCODE
CITY
FEDERALSTATE
COUNTRY
PHONENUMBER
INTERNETBANKINGPIN
INTERNETBANKINGBLOCKED

ACCOUNT
Target of a secondary index
Length: 114 bytes
ACCOUNTNUMBER
BIC
IBAN
ACCOUNTTYPE
BALANCE
CREDITINTEREST
OVERDRAFTAMOUNT
OVERDRAFTINTEREST
INTERESTACCOUNTINGINTERVAL
CURRENCY

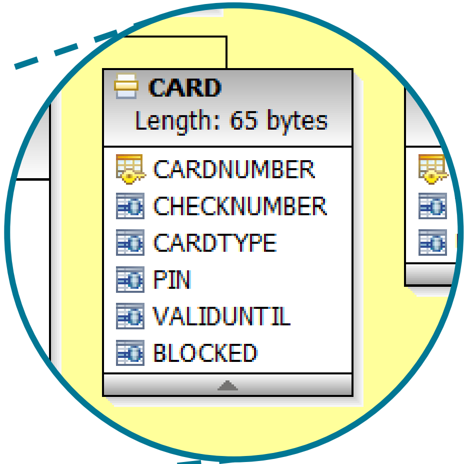
TANLIST
Length: 5 bytes
TANLISTNUMBER
ACTIVE

ENTRY
Target of a secondary inde
Length: 325 bytes
ENTRYNUMBER
ENTRYTYPE
ENTRYTIME
BIC
IBAN
ACCOUNTHOLDER
AMOUNT
ADDITIONALINFORMATION
USEDCARD
USEDTAN
ACCOUNTED
PRINTED

CARD
Length: 65 bytes
CARDNUMBER
CHECKNUMBER
CARDTYPE
PIN
VALIDUNTIL
BLOCKED

ITAN
ength: 19 bytes
ISTPOSITION
ANVALUE
USED

For the authentication process at the ATM the information stored in the **card** segment is needed. Besides the card number this segment contains the encrypted PIN as well as the validity date and the blocking state.



CARD
Length: 65 bytes
CARDNUMBER
CHECKNUMBER
CARDTYPE
PIN
VALIDUNTIL
BLOCKED

The other segments
contain TAN data for internet banking.

Functional parts of the ATM application

- The functionality of the ATM application is split into eight parts:
  1. Authentication by credit or cash cards,
  2. Card blocking after failing three times the authentication,
  3. Balance inquiries,
  4. Deposits,
  5. Withdrawals,
  6. Transferals,
  7. Querying transactions for printing the account statements and
  8. Marking them as printed in the database
- This splitting allows a granular access right definition in the PSB for
  - More security,
  - Less locking,
  - …

IMS Catalog

Role of the IMS Catalog in Java applications

IMS Technical Symposium 2015

Agenda

Introduction

The sample application

IMS Catalog

IMS Open Database

The 3-tier ATM application

Java in IMS dependent regions

IMS Enterprise Suite Connect API

The 4-tier ATM application

Live Demo

Summary

Questions?

Java metadata classes vs. IMS Catalog

- Java applications do need a description of the database layout and the program view to the database
- IMS supports two solutions for providing the metadata needed by Java applications:

Java metadata classes	**IMS Catalog** (V12+)
Local storage, may be distributed over different systems	Central storage, IMS has full control on the metadata
May be obsolete in comparison with the active ACBLIB	Up-to-date in comparison with the ACBLIB

Java metadata classes vs. IMS Catalog

Java metadata classes	**IMS Catalog** (V12+)
Less initial effort, more continuous effort caused by recurrent metadata class generation (done by IMS Enterprise Suite Explorer) and their deployment	More initial effort caused by IMS Catalog setup, less continuous effort
Faster connection setup because of local metadata	Slower connection setup because of metadata querying during the connection setup

Java metadata classes vs. IMS Catalog

Java metadata classes	**IMS Catalog** (V12+)
Changes for meaningful names instead of short IMS segment or field names as well as changes for data conversion from IMS datatypes to SQL datatypes can be done in the metadata class, no change of the DBD source necessary	DBD changes needed for both meaningful names instead of short IMS segment or field names and data conversion from IMS datatypes to SQL datatypes
No database versioning possible	IMS Catalog allows database versioning, but the handling is complex

IMS Catalog metadata management

- IMS Catalog is the recommended metadata source
- The Catalog is built from the ACBLIB by running the Catalog Populate Utility DFS3PU00 (or the combined ACB Generation and Catalog Populate Utility DFS3UACB)
- The metadata can be accessed by issuing a GUR call through DFSDDLT0 or through a REXXTDLI application
- The getCatalogMetadataAsXML() function of the IMS Universal DL/I driver provides a similar function for Java applications
- This function is also used by the Universal drivers during the connection setup (if the Catalog is used as metadata source) and by the IMS Enterprise Suite Explorer
- Old metadata can be deleted by running the Catalog Record Purge Utility DFS3PU10

IMS Catalog structure

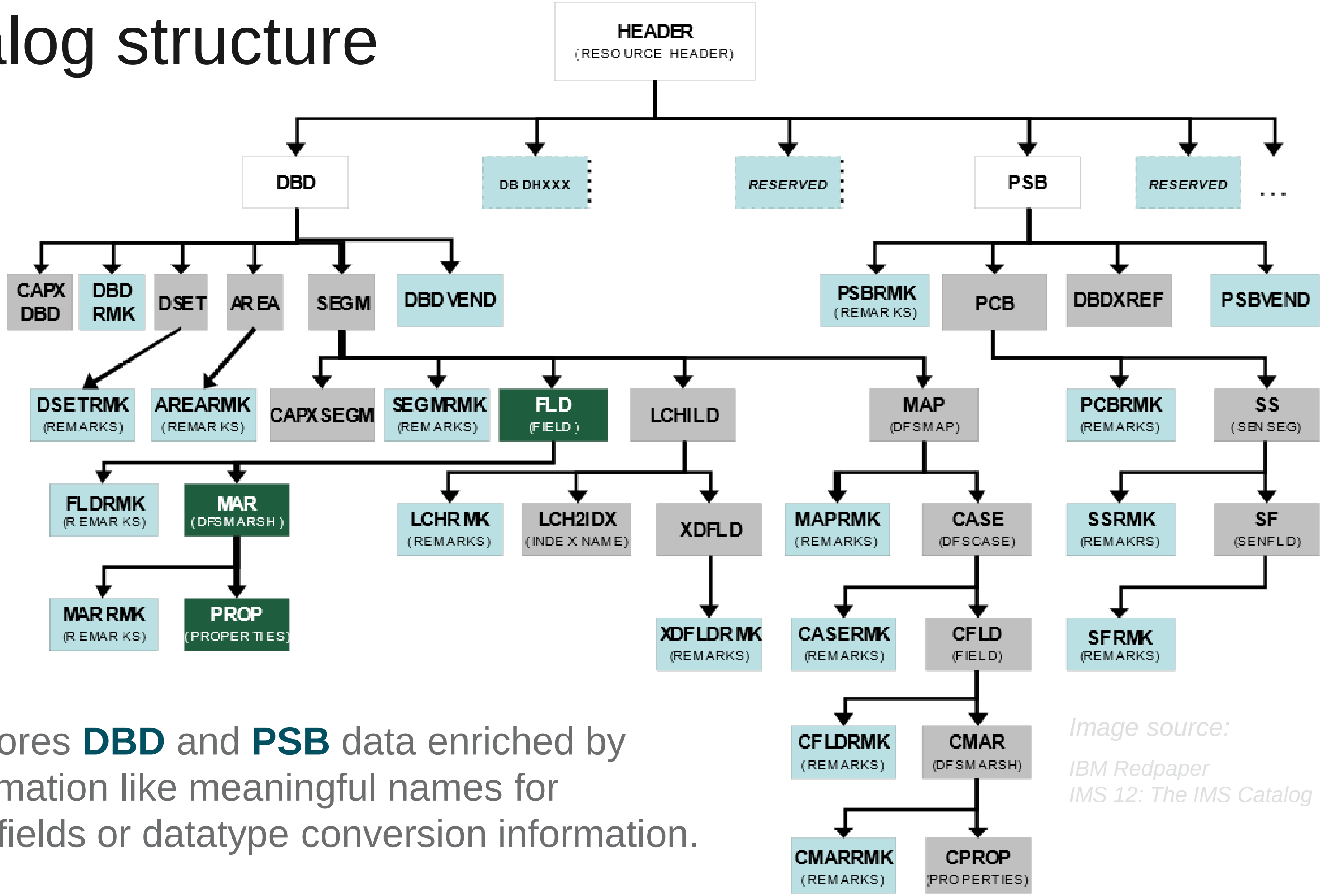


Image source:

IBM Redpaper
IMS 12: The IMS Catalog

IMS Catalog stores **DBD** and **PSB** data enriched by additional information like meaningful names for segments and fields or datatype conversion information.

DBD and PSB metadata enrichment

- DBD metadata enrichment:
  - Use EXTERNALNAME on SEGM and FIELD statement to specify meaningful names for SQL access
  - Also needed on XDFLD statement for SQL access through secondary index (PTF installation necessary)
  - Use DATATYPE on FIELD statement to specify the corresponding SQL datatype (for instance BIT, BYTE, INT, LONG, FLOAT, DOUBLE, DECIMAL, BINARY, CHAR, …)
  - Use DFSMARSH statement after FIELD statement for additional datatype conversion definitions like patterns for DATE, TIME and TIMESTAMP (for instance 'yyyy.MM.dd')
- PSB metadata enrichment:
  - Use EXTERNALNAME on PCB statements to specify meaningful database names for SQL access

IMS Open Database

Direct access to IMS data from distributed systems

Agenda

Introduction

The sample application

IMS Catalog

IMS Open Database

The 3-tier ATM application

Java in IMS dependent regions

IMS Enterprise Suite Connect API

The 4-tier ATM application

Live Demo

Summary

Questions?

IMS Open Database overview

- The IMS Open Database was introduced with version 11
- Allows direct access to IMS data from distributed environments
- Standardizes the ways of access for all Java applications
- Both DL/I and SQL can be used to access the data
- Distributed access is routed over IMS Connect
- Therefore a new IMS Connect port must be configured
- IMS Connect passes incoming connections to the Open Database Manager
- The ODBM is a new IMS component and part of the Common Service Layer

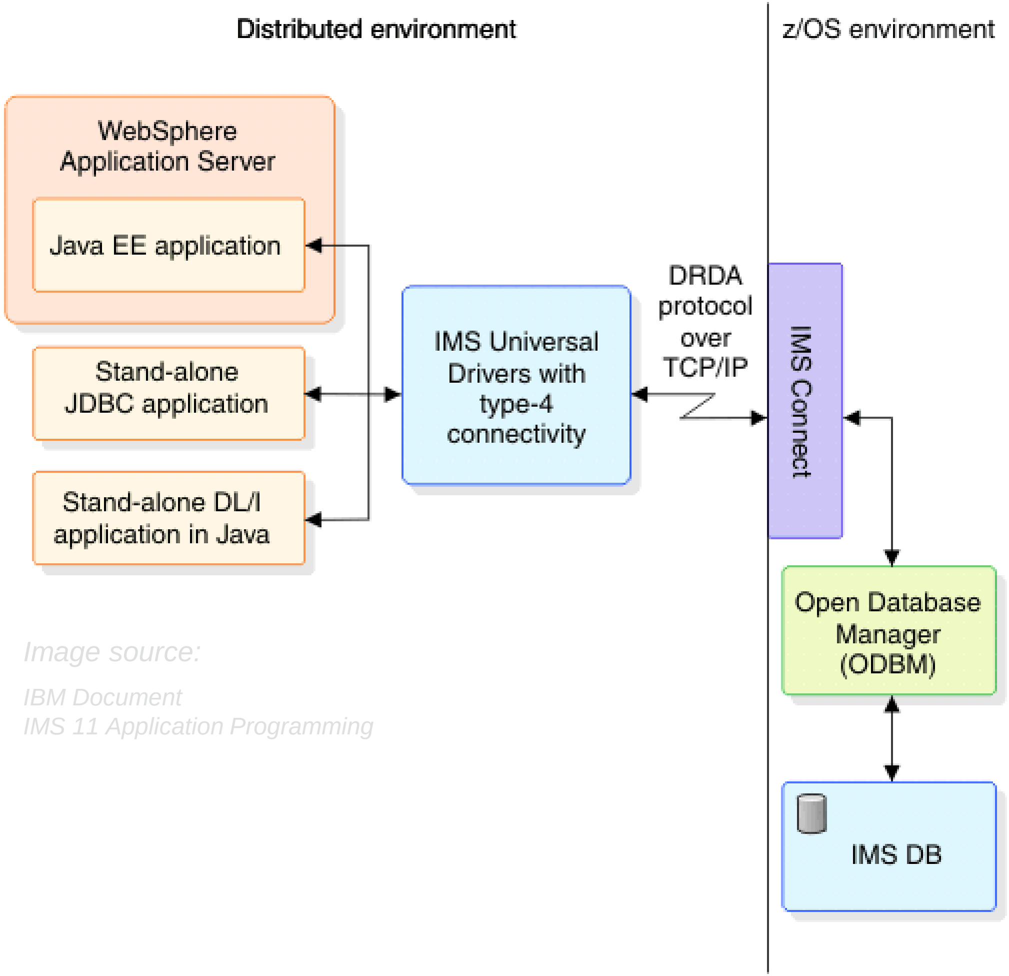

Distributed environment
z/OS environment
WebSphere Application Server
Java EE application
Stand-alone JDBC application
Stand-alone DL/I application in Java
IMS Universal Drivers with type-4 connectivity
DRDA protocol over TCP/IP
IMS Connect
Open Database Manager (ODBM)
IMS DB

Image source:

IBM Document
IMS 11 Application Programming

Java applications running on distributed systems have to establish a **type-4 connection** to ODBM.

- Stand-alone applications have to use the IMS Universal JDBC or DL/I driver provided by the ZFS file imsudb.jar
- Java EE applications running under WAS or other application servers are able to use the IMS Universal DB Resource Adapters instead (provided by several different ZFS files)

The 3-tier ATM application

A lightweight architecture for medium access rates

IMS Technical Symposium 2015

Agenda

Introduction

The sample application

IMS Catalog

IMS Open Database

The 3-tier ATM application

Java in IMS dependent regions

IMS Enterprise Suite Connect API

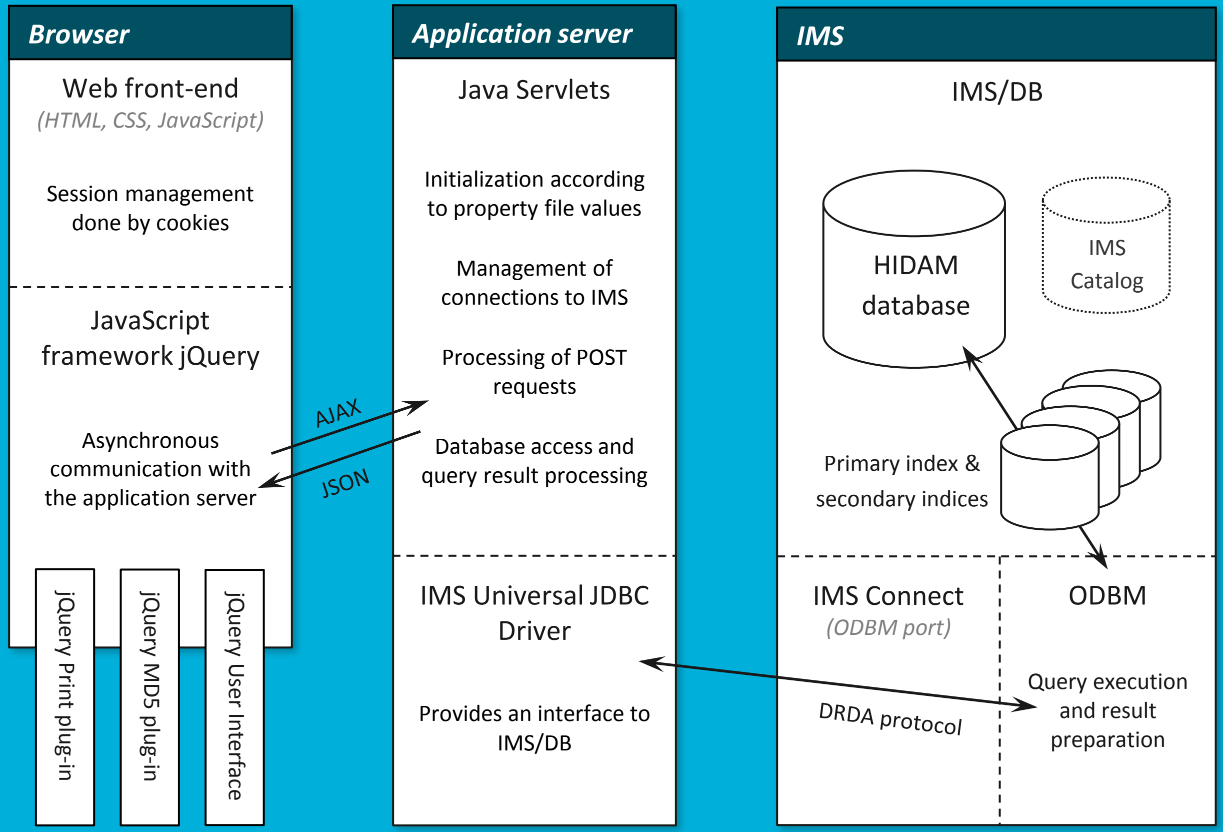
The 4-tier ATM application

Live Demo

Summary

Questions?

Architectural overview



Browser

Web front-end
(HTML, CSS, JavaScript)

Session management done by cookies

JavaScript framework jQuery

Asynchronous communication with the application server

jQuery Print plug-in

jQuery MD5 plug-in

jQuery User Interface

AJAX

JSON

Application server

Java Servlets

Initialization according to property file values

Management of connections to IMS

Processing of POST requests

Database access and query result processing

IMS Universal JDBC Driver

Provides an interface to IMS/DB

DRDA protocol

IMS

IMS/DB

HIDAM database

IMS Catalog

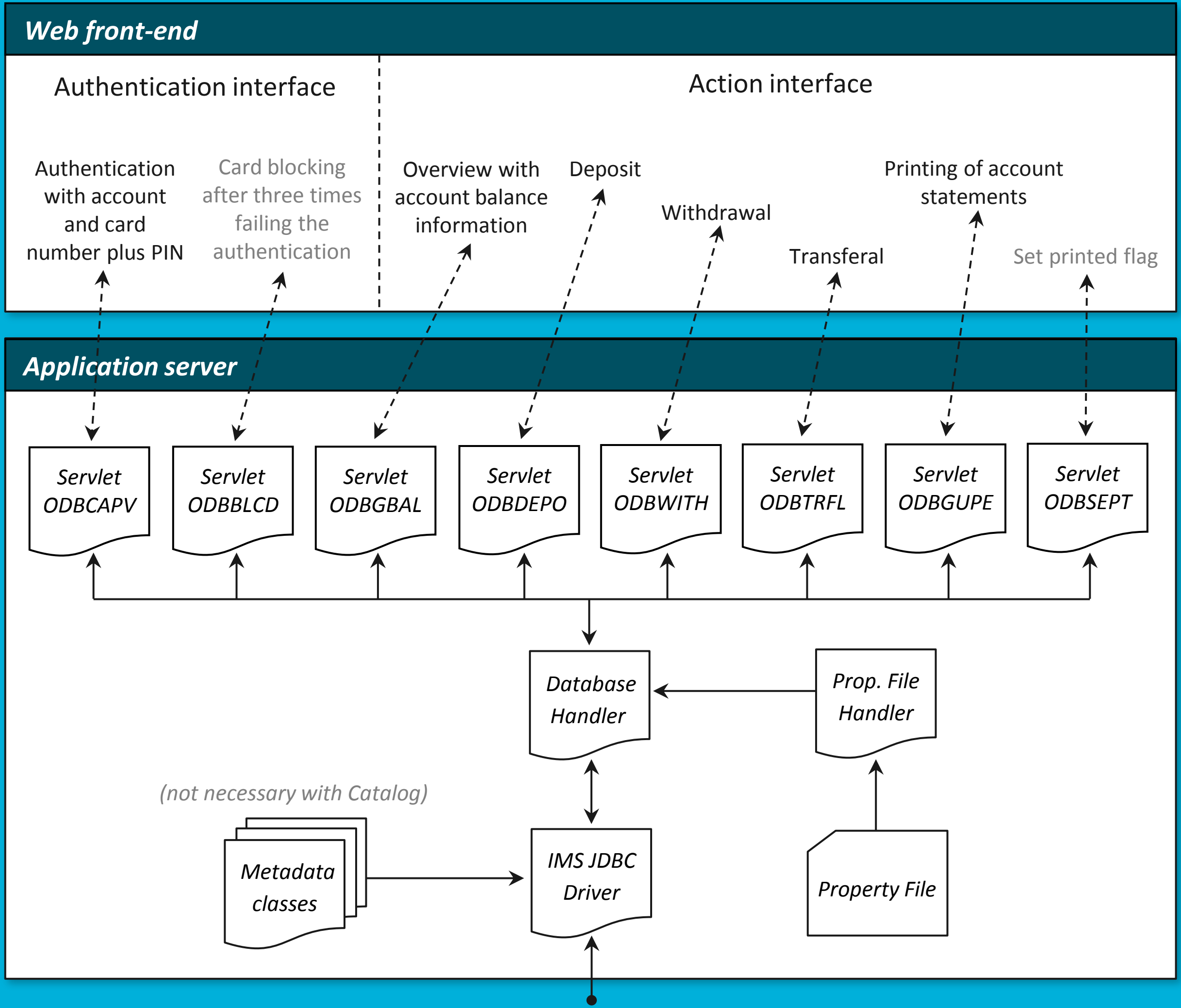
Primary index & secondary indices

IMS Connect
(ODBM port)

ODBM

Query execution and result preparation

Functional overview



Different approaches of Servlet connection handling

There are two different ways to handle connections to IMS/DB in a Java Servlet:

1. Keeping connections alive the whole Servlet lifecycle or
2. Connection setup and teardown per request

Keeping alive	Setup and teardown per request
Perfect for high access rates, maybe not ideal for occasional access	Unsuitable for high access rates, maybe better for occasional access
Shorter response times	Higher response times through connection setup (especially if the IMS Catalog is the metadata source)

Different approaches of Servlet connection handling

Keeping alive	Setup and teardown per request
Timeout management must be mostly done by the Servlet, several IMS Connect timeouts must be deactivated	Bigger parts of the timeout handling can be done by IMS Connect, no IMS Connect timeouts must be deactivated
Less CPU usage, higher real memory usage caused by permanent connections	Higher CPU usage caused by repetitive connection setup and teardown, less real memory usage

During the implementation of the ATM application did appear some actually unsettled problems with IRLM lock cycles and/or hanging ODBM threads when using permanent connections.

Java in IMS dependent regions

Modernization of IMS core applications

IMS Technical Symposium 2015

Agenda

Introduction

The sample application

IMS Catalog

IMS Open Database

The 3-tier ATM application

Java in IMS dependent regions

IMS Enterprise Suite Connect API

The 4-tier ATM application

Live Demo

Summary

Questions?

Preconditions of running Java applications under IMS

- Java applications can run under IMS in Java Message Processing (JMP) and Java Batch Processing (JBP) regions as well as in MPP regions for better interoperability with existing COBOL applications
- There are several preconditions for running Java applications in IMS dependent regions:
  - There must be a 31Bit JDK (V6+) installed and referenced in the DFSJVMEV member
  - The IMS JDR Resource Adapter as well as the IMS Universal Drivers must be mounted in a USS directory
  - The application itself must be located in a .jar file under USS
  - Resource adapter and drivers as well as the application itself must be referenced in the DFSJVMMS member
  - The IMS program name must be mapped to the application‘s Java class name in the DFSJVMAP member

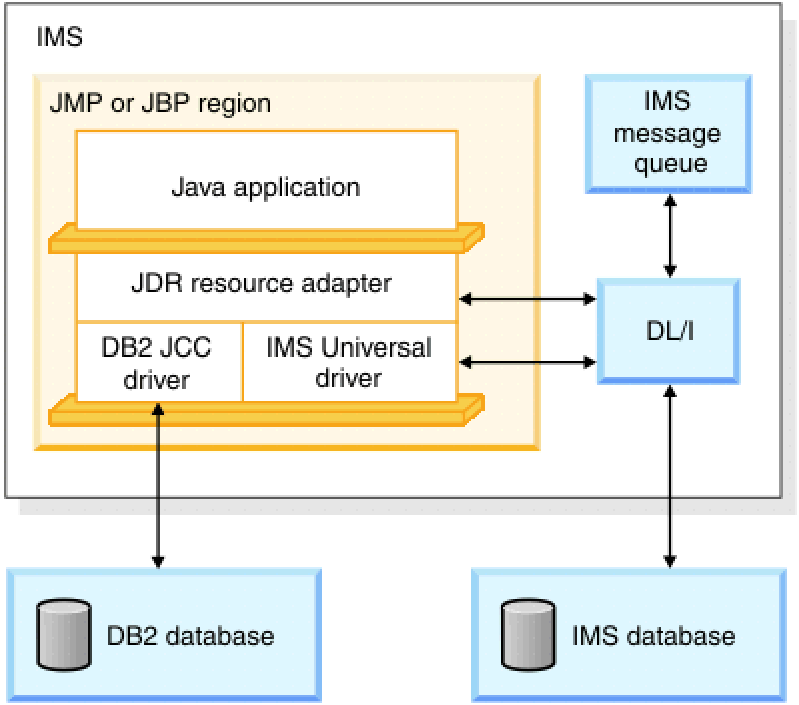


Image source:

IBM Document
IMS Version 11 Universal Drivers und JDR Resource Adapter Type-2 Support Programming Guide

JMP transaction programs access the IMS message queues through the IMS JDR Resource Adapter.

All Java applications running in IMS Java dependent regions are able to establish **type-2 connections** to IMS/DB.

They are also able to access DB2 through the DB2 JCC Driver, which has to be referenced in the DFSJVMMS member.

DB2 has to be attached to the region through RRSAF in IMS versions previous to V13.

IMS Enterprise Suite Connect API

Accessing IMS transactions from distributed environments

Agenda

Introduction

The sample application

IMS Catalog

IMS Open Database

The 3-tier ATM application

Java in IMS dependent regions

IMS Enterprise Suite Connect API

The 4-tier ATM application

Live Demo

Summary

Questions?

IMS Enterprise Suite Connect API overview

- The IMS Enterprise Suite Connect API is a free downloadable API for IMS access through IMS Connect
- There is a Java and a C version of the API available
- The IMS Enterprise Suite Connect API is an alternative to the IMS TM Resource Adapter (previously named IMS Connector for Java)
- It provides all types of interaction with IMS transactions
- The API also provides an IMS command interface

Usage in the 4-tier ATM application

- ATM application uses the Java version of the API
- The API is used by Java Servlets to interact with JMP transactions
- Even if the ATM application only accesses JMP transactions, of course there is no limitation on the API to JMP transactions
- So the API also allows an easy integration of existing IMS transactions into modern application architectures

The 4-tier ATM application

A better scaling and more secure architecture

Agenda

Introduction

The sample application

IMS Catalog

IMS Open Database

The 3-tier ATM application

Java in IMS dependent regions

IMS Enterprise Suite Connect API

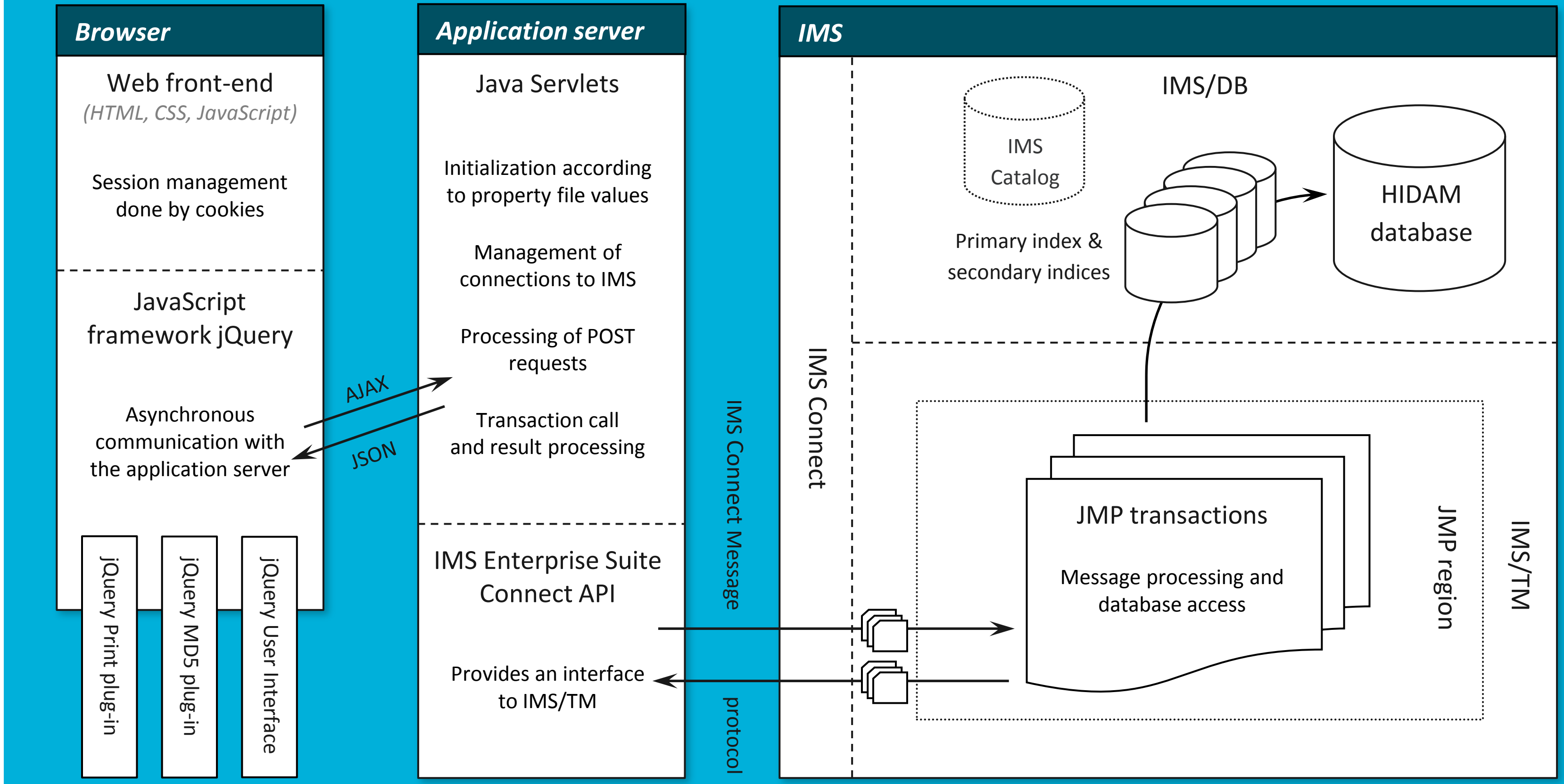
The 4-tier ATM application

Live Demo

Summary

Questions?

Architectural overview



Browser

Web front-end
(HTML, CSS, JavaScript)

Session management done by cookies

JavaScript framework jQuery

Asynchronous communication with the application server

jQuery Print plug-in

jQuery MD5 plug-in

jQuery User Interface

AJAX

JSON

Application server

Java Servlets

Initialization according to property file values

Management of connections to IMS

Processing of POST requests

Transaction call and result processing

IMS Enterprise Suite Connect API

Provides an interface to IMS/TM

IMS Connect Message protocol

IMS

IMS Connect

IMS/DB

IMS Catalog

Primary index & secondary indices

HIDAM database

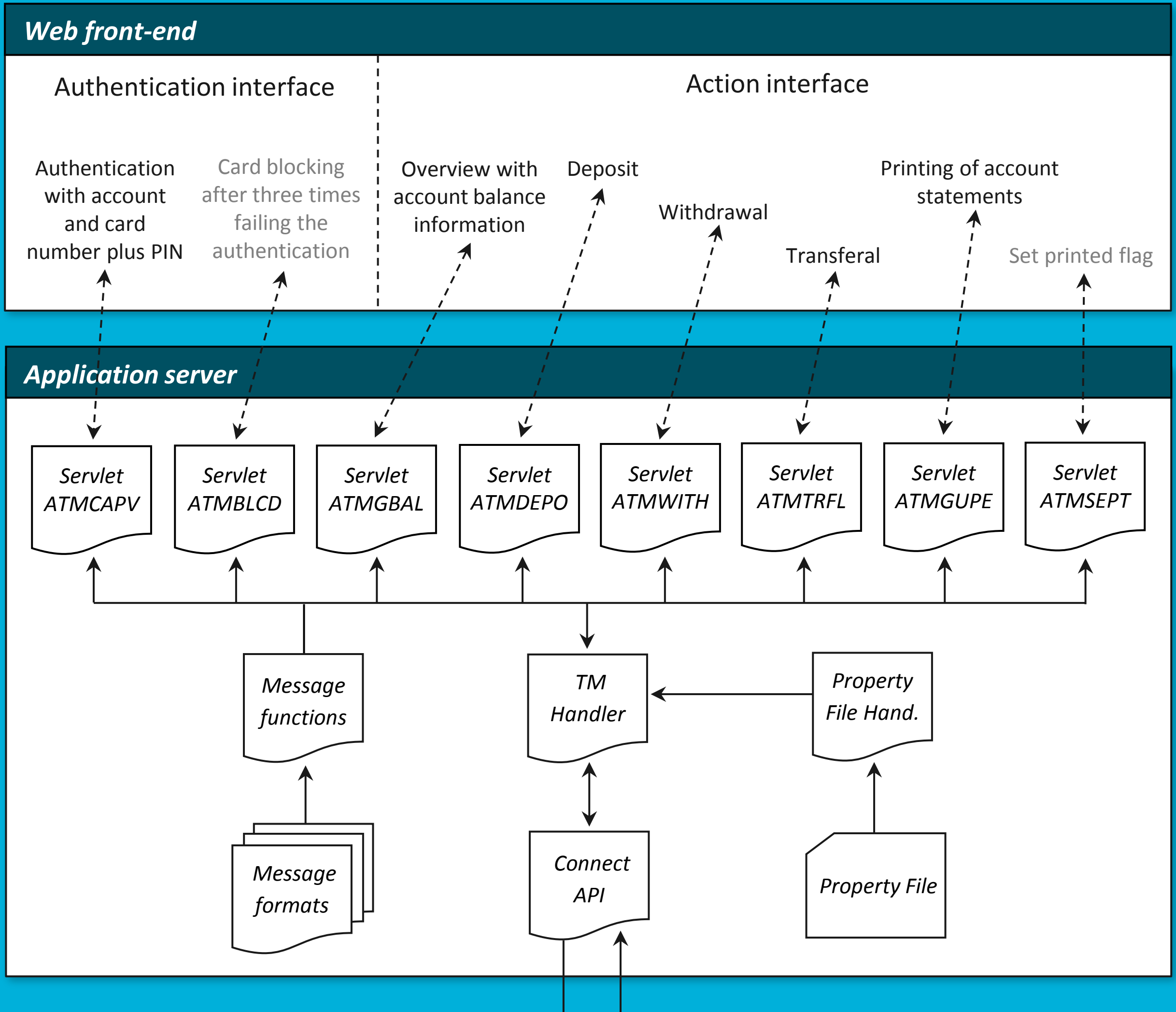
JMP transactions

Message processing and database access

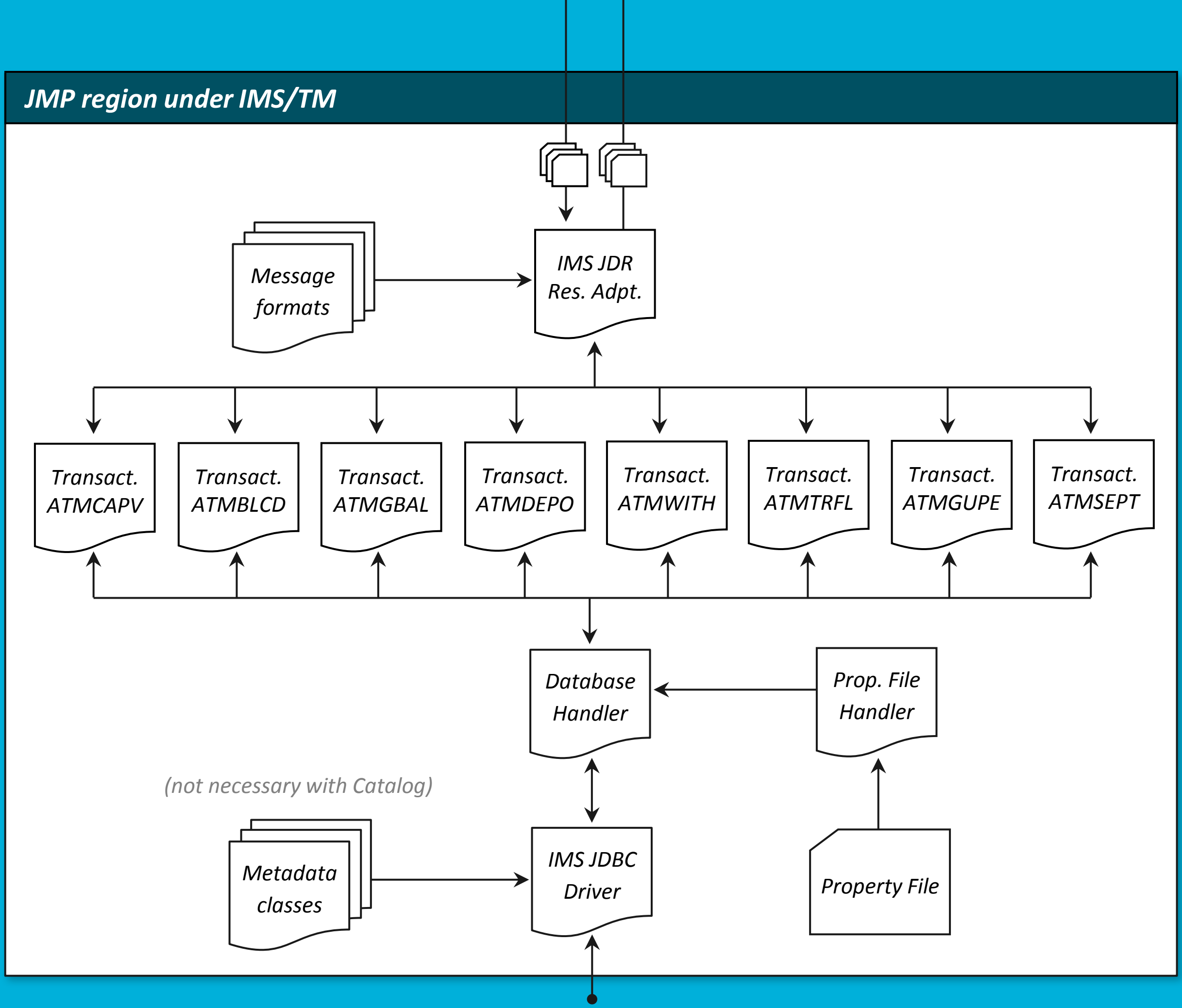
JMP region

IMS/TM

Functional overview



Functional overview



Live demo

Having a look into the ATM application

Agenda

Introduction

The sample application

IMS Catalog

IMS Open Database

The 3-tier ATM application

Java in IMS dependent regions

IMS Enterprise Suite Connect API

The 4-tier ATM application

Live Demo

Summary

Questions?

Summary

What is the best solution for your business case?

Agenda

Introduction

The sample application

IMS Catalog

IMS Open Database

The 3-tier ATM application

Java in IMS dependent regions

IMS Enterprise Suite Connect API

The 4-tier ATM application

Live Demo

Summary

Questions?

Comparison of both architectures

3-tier architecture	4-tier architecture
Light-weight architecture for medium workloads with less parallel access	More complex but better scaling solution (especially if JMP transaction programs are optimized for re-entrant usage)
A bit higher response times	Less response times (excluding the first access after the start of the JMP region)
Difficult to guarantee a high security level	More secure because database access only by IMS controlled programs

Comparison of both architectures

3-tier architecture	4-tier architecture
Longer locks if DB2 should also be accessed because the application server must be the commit coordinator	Shorter locks because IMS is the commit coordinator
Potentially less costs for medium access rates	Potentially less costs for high access rates
Less implementation effort	More implementation effort, good knowledge of IMS and z/OS JDK needed for implementation of JMP transaction programs

Comparison of both architectures

<table>
<tr><th>3-tier architecture</th><th>4-tier architecture</th></tr>
<tr><td>ODBM needed</td><td>No ODBM needed</td></tr>
<tr><td>RRS usage necessary for parallel DB2 access</td><td>No RRS usage necessary for parallel DB2 access since V13</td></tr>
<tr><td>Difficult to monitor</td><td>Better monitoring possibilities</td></tr>
<tr><td colspan="2">Potentially long search for errors</td></tr>
</table>

It is hard to compare the resource usage because this depends on the business case and the access rate.

Alternative solutions

- Alternative application server components:
  - Java Connector Archticture (JCA)-compliant database attachment using the IMS Universal DB Resource Adapters (enables two-phase commit for access to multiple back-end systems)
  - Enterprise Java Beans providing the core mid-tier functionality like session management and database access behind the Servlet layer (better reuseability of software components in other business cases)
- Alternative architecture options:
  - IMS Enterprise Suite SOAP Gateway
  - z/OS Connect

➔ Visit session B16 for more details

Security considerations

- In the sample environment IMS Connect does an RACF authorization
  - RACF passwords are stored encrypted in the property files
  - For the encryption the light version of the jasypt framework is used
- There are additional security checks in IMS:
  - Is the user allowed to access the transaction?
  - Is the user allowed to allocate the PSB?
- The sample application is not that secure as a real application should be:
  - The session management is only done at the client (that is insecure even if there is an invisible session timeout handling)
  - Most value checks are only done at the front-end
  - Scripting and SQL injection prevention is only implemented rudimentarily
  - The communication between the layers is insecure

Security considerations

- How to provide more security?
  - Implement the session management not only on client side
  - Do not use HTTP GET requests with clear-text parameterization
  - Better only use POST requests (like the ATM application does)
  - Use the HTTPS instead of the HTTP protocol for the communication between front-end and application server
  - Do additional value checks through the application server components
  - Use SSL connections between Servlets and IMS Connect (both the IMS Universal drivers and the IMS Enterprise Suite Connect API support SSL)
  - Allow only access to IMS Connect from defined IP addresses

Questions?

Thank you for your attention!