EBX5 Technical Architecture & Integration

June 2015
Architecture overview

Client
- 100% Browser-based (HTTP/HTTPS)
- Supported browsers: IE, Firefox, Chrome, Safari
- Web Services client: SOAP over HTTP/HTTPS or JMS

Server
- Servlet container in JRE 1.5+ (Servlet 2.4)
- Example: IBM WebSphere, Oracle WebLogic, Apache Tomcat, RedHat JBoss
- Deployment: EAR or WAR/JAR

Database
- Oracle Database 10gR2 or higher
- PostgreSQL 8.4 or higher
- Microsoft SQL Server 2008 or higher
- IBM DB2 UDB v8.2 or higher
- H2 v1.3.170 or higher (dev)
Hardware / Service Components

Java Application Server

Web Applications
- Core EBX5 Web Apps
  - Core EBX
  - EBX ROOT
  - Manager
- Optional EBX5 Web Apps
  - DMA
  - Data Services
  - iGov
  - Matching
  - Insight
  - EPM

Modules (project web apps)
- Project 1
  - Data Model
  - Resources
  - Code
- Project 2
  - Data Model
  - Resources
  - Code

Libraries
- Core EBX5 Libraries
  - ebx.jar
  - ebx-addons.jar
- Optional 3rd Party Libs
  - JDBC driver
  - JavaBean
  - JavaMail
  - SSL

RDMBS
- Persistence Tables
- History Tables
- Other (e.g. KPI values)
Logical Architecture

Mail Server (optional)

Web Server

Apps. Server

WebServices

LDAP

Directory/LDAP (optional)

RDBMS Server

Logical Architecture

SMTP - 25

LDAP - 89

HTTP/S - 8080

HTTP/S - 8080

HTTP / SOAP

JDBC - 1521

Orchestra NETWORKS
Interfaces & Deployment
Interfaces

EBX5

Data Services
Web Services (SOAP/WSDL) generated from data models

Files
File import/export in XML, CSV, XLS, XLSX

RDMBS
SQL
Native SQL access
Web Services (SOAP/WSDL) dynamically generated from data models. Over HTTP/S or JMS

**Product Data Model (XML Schema doc)**
- Product
  - Product-ID
  - Name
  - Family
  - Pricing 0 - ∞
  - Price
  - Discount

**Product Data Service (WSDL)**
- Operations
  - select_Product
  - insert_Product
  - update_Product
  - delete_Product
  - count_Product
  - getChanges_Product

**MDM Features Data Service (WSDL)**
- Operations
  - create_DataSpace
  - create_Snapshot
  - close_DataSpace
  - close_Snapshot
  - merge_DataSpace
  - validate_DataSet
  - validate_DataSpace
  - start_workflow
  - end_workflow

Note: rich types (here multi-value complex type) supported in both data model and data services

Dynamically generated Data Services from a Data Model
Prebuilt Data Services on MDM features
Data Services

Objectives
• Provide a standards-based service layer on MDM
• Expose all operations as a service
  o CRUD on data sets
  o MDM features (workflow, version control...)
• Decouple MDM from applications

How it works
• Dynamic generation of Web Services from the data model
• WSDL mapped on XML Schema data model (same standard)
• Absorbs data model changes on the fly
  o No code generation required
  o No redeployment cycle required

Key features
• CRUD web service on any Data Set (based on Data Model)
  o Insert / Update work on both bulk and per record modes
• MDM features on Data Spaces, Workflow...
• Security: HTTP Authentication or WSSE
• SOAP-fault for validation errors
Data Propagation

- EBX provides a canonical format in XML, XLS or CSV (the one created during the data model design time)
- A middleware (e.g ODI) is in charge to transform this format to a suitable format for the target system and integrate the transformed data using a dedicated connector.

<table>
<thead>
<tr>
<th>ETL</th>
<th>ESB</th>
<th>Autre (DQ, BPM, BRMS)</th>
</tr>
</thead>
</table>
| Informatica Powercenter  
IBM Datastage  
Oracle ODI  
Talend ETL  
Microsoft SSI | WebMethods ESB  
JBoss ESB  
Oracle OSB  
Tibco ESB  
IBM ESB  
IBM MQ  
Microsoft BizTalk | IBM Quality Stage  
Tibco BusinessWorks  
WebMethods BPM  
Oracle BPM  
IBM JRules  
Informatica Data Quality  
MS SharePoint  
MS Excel |
Data Propagation

Synchronous/Asynchronous mode:
• EBX provides both modes
  • Best practice for real time is to sync at merge time
  • Batchs can be triggered using any third party software or using the natively provided scheduler

Delta/Full propagation:
• EBX enables delta and full stock mechanisms, down to attribute level
• Delta is easily identified using dataspaces
• Native web service to get delta between dataspaces/versions
Data Propagation

Target systems/environments:

- EBX provides native export/import archive mechanism in order to propagate data between environments (also applies to data model, workflow models, configurations, etc).
- Leverages dataspace native merging mechanism to put together data from both source and target systems.
- Can be automated using native scheduler (or third party tool), or in sync with dataspace merge event.
SQL Interfaces for data access

Using SQL Connect, Master Data are automatically replicated in native SQL tables.

Record level history is stored in native SQL tables.

Data Quality indicators generate historical data in native SQL tables.
Deployment architectures

Multiple deployment options can be combined

- Active/Passive Failover
- DB clustering
- EBX clustering (using D3)

Multiple deployment options can be combined.
Integration scenario
Examples
Example: Authoring and export via an ETL
Example: Authoring and export via an ESB

- UI
- Main Data Space
  - Insert a new record
  - Merge Data Space
- Notify middleware
- Get changes or select
- Push updates to apps
- EBX5
- ESB
- APPS
Example: xRef look-up in an integration process

- Integration process
- Transform
- Push updates to apps
- Look-up xRef
Example: Import data in staging areas

- **Main Data Space**
  - Staging 1
  - Staging 2
- **Merge Data Space**
  - Import data from App 1
  - Import data from App 2
Example: Launch approval workflow

- **Start Workflow**
  - Main Data Space
  - Update
  - Import data from App
  - Launch approval workflow
  - Approve in workflow
  - Merge Data Space
D3: Distributed Data Delivery
**Distributed architecture**

**Master EBX**
- Data Governance Time

**Slave EBX**
- Data Consumption Time

**EBX**
- D3 Slave

**D3 Master**
- Defines a data space as deliverable
- Associates it to delivery profiles
- Control & monitor profiles subscribers
- Broadcasts data updates
- 2-phase commit

**D3 Slaves**
- Automatic registering
- Receive updates
- Commit changes
- Read-only on synchronized data spaces
- Write on local data spaces

1. Data is updated in EBX (via UI or Services)
2. Update is pushed to D3 Slaves
3. Update is received by D3 Slaves
4. Acknowledgment by D3 Slaves to D3 Master
5. D3 Master sends commit instruction to D3 Slaves
6. Update is committed to EBX Slaves

Messages: SOAP on HTTP/HTTPS, Data updates on TCP-IP
Example: Geographical Federation

Master MDM

NA Cluster
  Slave 5
  Slave 6

EU Cluster
  Slave 1
  Slave 2

APAC Cluster
  Slave 3
  Slave 4
Example: Realtime Data Cluster

- **Update**
- **Load Balancer**
- **EBX Master**
  - App Server
  - JVM
  - EBX
  - RDBMS
- **EBX Slave 1**
  - App Server
  - JVM
  - EBX
  - RDBMS
- **EBX Slave 2**
  - App Server
  - JVM
  - EBX
  - RDBMS

- SOAP HTTP Synchronisation
- Data Spaces Broadcast
- TCP/IP Socket
- Read-only access
Managing master data at different levels

Product data is managed centrally in the master MDM instance.

Product data is synchronized to the slave MDM instance.

Local suppliers are managed in the slave MDM instance and linked to global products.
Integration with Enterprise Security Systems
Custom Authentication

➢ Requirement
  o Enterprise directory (for instance LDAP, Active Directory) responsible for holding the users' authentication information (login/password).
    • Passwords are not stored in EBX5.
    • Whenever opening a session (log on through the GUI, SOAP request or access through the Java API), authentication is performed against the enterprise directory.

➢ Implementation
  o Users declared in EBX5 directory only with their login (no password), to be able to assign them roles.
  o Develop a custom Java class (using EBX5 API) responsible for delegating the authentication to the enterprise directory (usually using the LDAP protocol).
Fully externalized users and roles

- Requirement (in addition to the "custom authentication" use case)
  - User - role associations declared and maintained in the enterprise directory.

- Implementation
  - EBX5 directory not used.
  - Authentication delegated to the enterprise directory.
  - Custom Java component responsible for retrieving the user's roles from the enterprise directory whenever needed by EBX5 (in order to avoid overloading the directory, this component would typically maintain a cache of users and roles with an appropriate synchronization policy).
Custom internal directory in EBX5

➢ Requirement
  o Periodic password expiration.
  o Account locking when typing a wrong password three times in a row.
  o Specific pattern for passwords:
    • Eight characters length.
    • Must contain letters and digits.

➢ Implementation
  o EBX5 built-in directory not used.
  o Custom directory inside EBX, as a data set in a dedicated Data Space
    • Based on a custom schema (built-in directory's schema can be re-used and enriched).
    • Specific rules (password expiration + account locking) in a custom Java class
      overriding built-in authentication.
  o Define role-based permissions on these Data Space and data set:
    • Only authorized users (directory's administrators) can access and update it.
    • The Data Space hosting the directory is hidden for business users.
Single sign-on (1/2)

EBX5 can be integrated into a SSO context, in order to share user authentication with other applications (Once authenticated, the user can access any application part of the SSO context).

EBX5 is compatible with the different kinds of SSO Systems:
- NTLM (Integrated Windows authentication)
- Kerberos
- IBM Tivoli Directory Server
- Lightweight Third Party Authentication (IBM WebSphere)
- Central Authentication Service (CAS)
EBX5 does not handle authentication anymore.
Override the built-in authentication mechanism. Implementation will depend on the target SSO System (EBX5 API gives access to the HTTP request, this is useful for ticket-based SSO Systems).

- When a user without any active EBX5 session tries to access the GUI:
  - Check if the user is already authenticated in the SSO context.
  - If not, redirect him to the SSO authentication page.
  - If yes, open an EBX5 session for this user and redirect him to the GUI home page.

- EBX5 GUI login policy can be customized if necessary:
  - Configuration done in the deployment descriptor of ebx.war (web.xml).
  - Choose the appropriate authentication method (Basic, Digest or Form-based, as described in Servlet 2.3 specification).

- When using Form-based authentication method: Customize the default login page in order to submit login/password to the SSO authentication system.
Thank you.

www.orchestranetworks.com

twitter.com/orchestramdm

www.linkedin.com/company/orchestra-networks

www.facebook.com/OrchestraNetworks