Oracle Messaging: AQ vs. WebLogic JMS

Bertrand Caradec
Logica Deutschland GmbH & Co KG, Now Part of CGI
Sulzbach (Taunus)

Schlüsselworte

Messaging, MOM, AQ (Advanced Queuing), JMS (Java Message Service), WebLogic, MDB (Message-driven Beans), XMLType, Oracle Objekten

Einleitung

MOM (Messaging-Oriented Middleware) ist der allgemeine Begriff für die Hardware/Software die Nachrichtenaustausch zwischen distribuierte Anwendungen bietet. Anders als bei einer RPC (Remote Procedure Call) Middleware, wird beim MOM der Nachrichtenversand asynchron vorgenommen. Die Elemente eines MOM Systems sind Clients, Nachrichten und der MOM Provider, der eine API und administrative Tools zur Verfügung stellt.

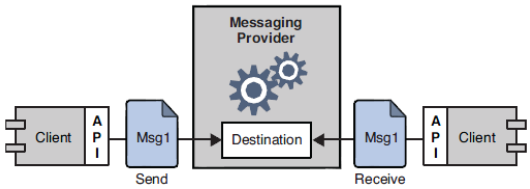


Abb. 1: MOM

Oracle bietet verschiedene MOM Providers:

- AQ (Oracle Advanced Queuing)
- WebLogic JMS
- GlassFish Message Queue
- Tuxedo Message Queue

In dem Vortrag werden die Möglichkeiten von Oracle AQ und WebLogic JMS verglichen. Neben den Grundkonzepten der Oracle AQ und WebLogic JMS werden die Ergebnisse einer Technologie Evaluierung im Rahmen eines anspruchsvollen Messaging Projekts präsentiert. Behandelt werden u.a.: Messaging Architektur, Persistierung, Skalierung, Message Types (XML vs Oracle User-Defined Type), Message Propagation, Nachrichtenverarbeitung durch Message-driven Beans vs PL/SQL Routine.

Überblick über die betrachten Messaging Technologien

JMS

JMS (Java Message Service) ist ein fester Bestandteil von JEE und wurde 1998 von Sun Spezifiziert (aktuelle Version ist 1.1 von 2002, JMS 2.0 erwartet für 2013). JMS definiert eine sehr allgemeine API für die Ansteuerung einer MOM zum Senden/Empfangen von Nachrichten aus einem Java Client heraus. Die JEE Anbieter müssen die JMS API umsetzen so dass der JMS Dienst bereitgestellt wird. JMS unterstützt zwei unterschiedliche Übermittlungsarten von Nachrichten: Point-to-point (Queue) Verbindungen und Publish/Subscribe (Topic) Kommunikation.

WebLogic JMS als MOM

Seit der Akquisition von BEA in 2008, ist der Java EE Applikation Server WebLogic das zentrale Produkt der Oracle Middleware. WebLogic JMS, die im WebLogic Server integrierte JMS Provider Implementierung, geht über den Umfang der JMS 1.1 Spezifikation hinaus mit folgenden Zusatzfunktionen: „Store and Forward“, „Distributed Destinations“, „JMS Clustering“, „Unit-of-unit“, „Unit-of-work“. Die wichtigsten Elemente von WebLogic JMS sind:

- JMS Server: Management Container für Queues und Topic
- JMS Module: Konfiguration von JMS Ressourcen (Queue, Topic, Connection Factory)
- JMS Client: produziert oder konsumiert Nachrichten
- Persistierung: File Storage oder JDBC Storage

Oracle AQ als MOM

Oracle AQ, Bestandteil des Oracle RDBMS, erschien mit der Version 8.0.3 als erstes integriertes Messagingsystem in einer Datenbank. Oracle AQ wird bei jeder Oracle Edition (XE, SE, EE) kostenlos mitgeliefert. Seit der Version 10.1 als „Oracle Streams AQ“ neu genannt, ist Oracle AQ der intern MOM Provider für den OESB (Oracle Enterprise Service Bus) und wird u.a. bei den Oracle Streams und Oracle Scheduler benutzt.

Oracle AQ besteht hauptsächlich aus zwei PL/SQL Packages:

- DBMS_AQADM (Rolle AQ_ADMINISTRATOR_ROLE): Administration von Queues (single or multi consumer), Queues starten/stoppen, Definition von Subscriber und Propagation
- DBMS_AQ (Rolle AQ_USER_ROLE): enqueue, dequeue

Da AQ eine reine Oracle Datenbank Lösung ist, gelten die Features eines Oracle Servers (Skalierbarkeit, Hochverfügbarkeit, Ausfallsicherheit) auch für die Oracle Queues. Oracle liefert dazu eine JMS Schnittstelle zu AQ (aqapi.jar, Java Package oracle.jms), so dass AQ auch als JMS Provider benutzt werden kann. Die JMS Clients greifen per JDBC an die Queues zu.

AQ Integration mit Weblogic JMS

Seit WebLogic Server 11g lässt Oracle AQ sich im WebLogic Server leicht integrieren. Dafür kommt der JMS Foreign Server im Einsatz. Der Foreign Server referenziert third-party JMS Providers (wie Oracle JMS) in einer lokalen WebLogic Server JNDI Struktur.

Das Messaging Projekt

Systemlandschaft

Das Passagierinfosystem (PasInSys) ist das zentrale Datenverteilsystem eines großen deutschen Verkehrsunternehmens. Wichtige Geschäftsvorfälle sind dabei Ist-Meldungen und Prognosen, Umleitungen, Ausfälle, Freie Informationen, Anschlüsse und Störungen. Der PasInSys -Server bekommt diese Informationen aus verschiedenen Systemen, aktualisiert entsprechend seinen

Tagesfahrplan und leitet die Informationen an verschiedenen Abnehmer weiter.

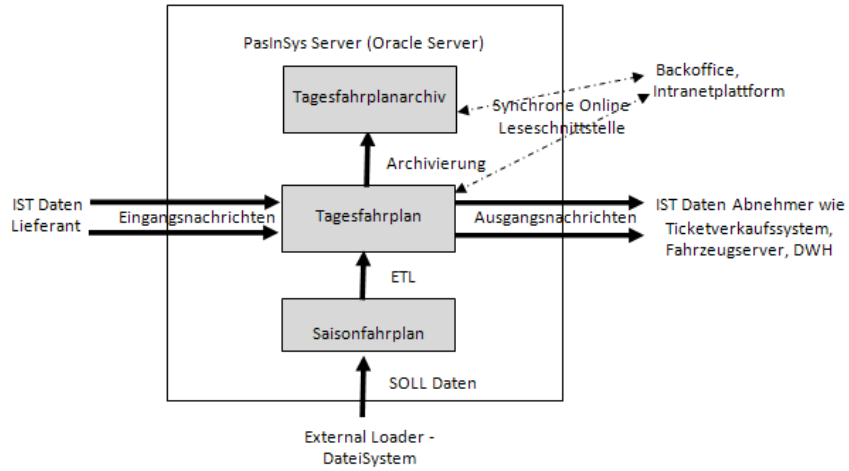


Abb. 2: Überblick des PasInSys Servers

Aktuelles Messagingsystem

Das aktuelle Messagingsystem ist auf Oracle AQ basiert. Auf dem PasInSys Oracle Server laufen parallele Oracle Scheduler Jobs die:

- XML Nachrichten aus den Eingangsqueues lesen
- den Fahrplandatenbestand aktualisieren
- entsprechenden gefilterten XML Nachrichten für die Abnehmer erzeugen und in den Ausgangsqueues bereitstellen

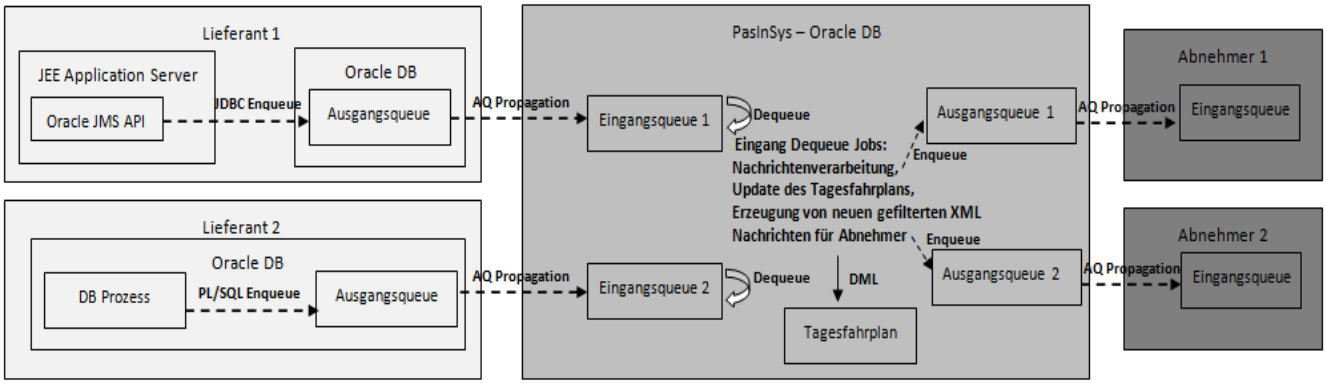


Abb. 3: aktuelles Messagingsystem

Technische Details:

- XMLTYPE ist der Payload Type der Nachrichten
- die XML Nachrichten werden zwischen die Lieferant, PasInSys und die Abnehmer durch AQ Propagation ausgetauscht. Die Ausgangsqueues sind als Multiconsumer Queue definiert, während die Eingangsqueues als Subscriber einer Ausgangsqueue eingetragen sind
- Storage Parameter bei Anlegen von Queues Tabellen (mit Advanced Compression Option):
  `xmltype user_data store as securefile clob LOB_segment_name (tablespace tablespace_name compress) tablespace tablespace_name`
- Bei der Nachrichtenverarbeitung werden die XMLTYPE Nachrichten Anhand des Packages DBMS_XMLDOM zuerst in eine DOM Struktur umgewandelt und sequentiell mit DOM Elementen gelesen.

- Die Exception Queues, die nach einem bestimmten Dequeue-Timeout oder nach x Rollback Dequeue der normalen Queue gefüllt werden, werden nach X Tagen geleert

AQ und WebLogic Evaluierung

Aufgrund neuer Anforderungen kommt es zu einer erheblichen Erweiterung der Soll- und der Ist-Daten im PasInSys Server, welche mit der aktuellen technischen Architektur und Hardware nicht erfüllt werden kann. Dafür wird ein Redesign des PasInSys Servers gebraucht. Eine Evaluierung der Messagingsystemen Oracle AQ / WebLogic JMS und der Technologie für den internen / externen Datenaustausch mit den neuen Anforderungen (2000 Nachrichten/s) sollte durchgeführt werden.

Evaluierung des Nachrichtenformats
Diese Evaluierung soll auch das Nachrichtenformat (XML oder Objektstruktur) auswerten.
Die benutzte Nachricht für den Test ist die am meistens verschickte XML Prognose Meldung. Das entsprechende Objekt ist als User Defined Type (UDT) in der Oracle Datenbank definiert und als Java Class mit der Hilfe des JPublisher generiert worden.
Um die Reaktion des Messagingsystems auf die Vergrößerung der Nachricht zu kontrollieren, wird jede Testreihe mit BIG Nachrichten zusätzlich ausgeführt. Eine BIG Nachricht ist 5 Mal so groß wie die normale Prognose Meldung.

Oracle Streams AQ 11gR2

a) 1 Queue ->1 Queue

In diesem Test werden nur 1 Eingangsqueue und die entsprechende Ausgangsqueue benutzt. Die Enqueuing und Dequeuing können parallelisiert werden aber die Propagation kann nur durch einen einzigen Job realisiert werden (es ist nicht möglich mehrere Propagation Jobs auf die gleiche Queue zu definieren).

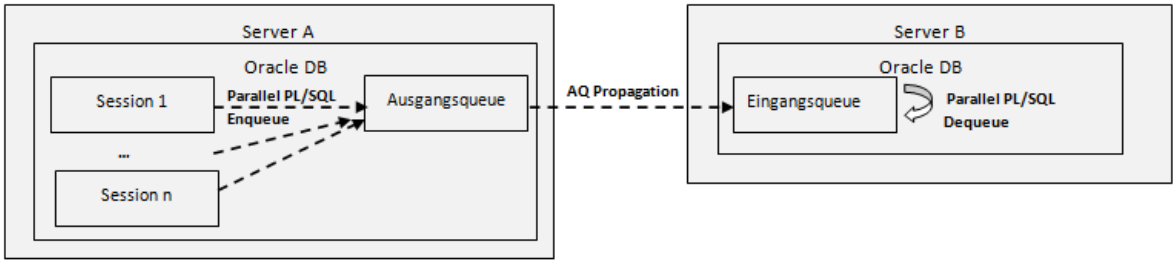


Abb. 4: 1 Queue ->1 Queue

Ergebnisse:

- die lokalen Enqueuing und Dequeuing lassen sich gut parallelisieren
- die Propagation wird aber nicht verbessert durch die Parallelisierung des Enqueuings
- das UDT Standard Format ist mit Abstand das effizientestes Format vor allem für die Propagation (Faktor 3 im Vergleich mit dem XML Standard)
- Ein BIG UDT Objekt lässt sich schlechter enqueued/dequeued als das entsprechende BIG XML Objekt
- Die Dequeuing Performance mit XMLType ist unabhängig von der XML Größe

b) n Queues ->n Queues

In diesem Test wird die Skalierbarkeit des AQ durch die Anzahl von Eingangs und Ausgangsqueue getestet. Jede Eingangsqueue wird zu einer bestimmten Ausgangsqueue propagiert.

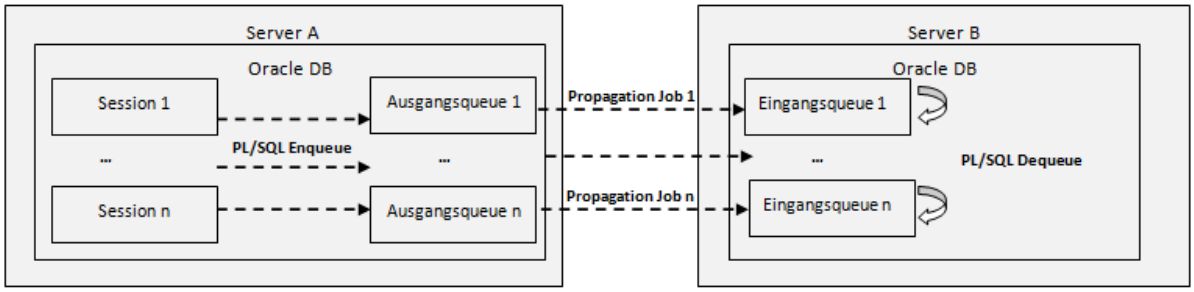


Abb. 5: n Queues ->n Queues

Ergebnisse:

- die Propagierung lässt sich durch mehreren Eingang und Ausgangsqueue skalieren
- UDT ist das leistungsfähigstes Format für Enqueuing, Propagierung und Dequeuing
- aber Enqueueing/Dequeueing verschlechtert sich deutlicher für einen BIG UDT im Vergleich zu einem BIG XML

<u>Oracle WebLogic Server 10.3.5 – JMS</u>

a) Vergleich zwischen JDBC und File Persistenz

Zu einem WebLogic JMS Server können zwei Arten von Persistenz zuordnet werden: File Storage oder Datenbank Storage per JDBC. Bei einem JDBC Storage erzeugt WebLogic die Tabelle WLSTORE in die entsprechende JDBC Data Source, wo die Nachrichten gespeichert werden. Struktur der Tabelle in einer Oracle Datenbank:

Feld	**Typ**
ID	NUMBER
TYPE	NUMBER
HANDLE	NUMBER
RECORD	LONG RAW

Defaultmäßig ist das Feld Record als LONG RAW (obsolet Typ von Oracle) definiert. Es ist aber möglich das Feld als BLOB zu deklarieren.

Bei einem File Storage muss ein WebLogic FileStore angelegt werden, was einen entsprechenden File Ordner in `MW_HOME\user_projects\domains\`*`domain-name`*`\servers\`*`server-name`*`\data\store\` erzeugt.

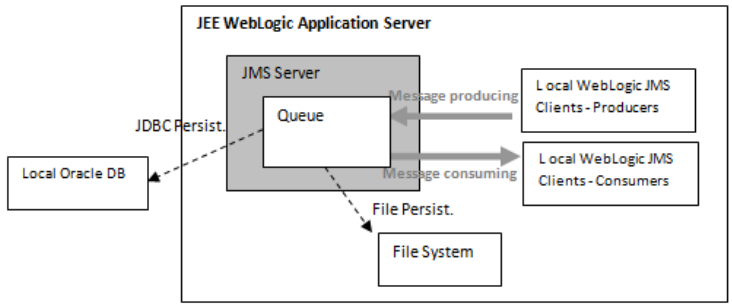


Abb. 6: Testarchitektur für die Evaluierung der WebLogic JMS Persistenz

Ergebnis: File Storage ist deutlich leistungsfähiger als JDBC Storage auf einer lokalen Datenbank. Mit einer remote Datenbank sollte der Unterschied noch deutlicher sein.

b) Vergleich von JMS Nachrichtentypen

JMS definiert 5 verschiedenen Typen von Nachrichtobjekten: `TextMessage`, `BytesMessage`, `ObjectMessage`, `StreamMessage` und `MapMessage` die die Interface `javax.jms.Message` implementieren. WebLogic JMS bietet dazu einen `XMLMessage` (`extends javax.jms.TextMessage`) Typ, der einen built-in Support für XPath hat.

Ergebnis: `ObjectMessage` lassen sich am bestens mit WebLogic JMS verarbeiten, vor allem für lokales Dequeing. Sie lassen auch sich viel besser skalieren als `XMLMessage`

c) Message Consuming mit Message Driven Beans

Ein Message-driven Bean (MDB) ist ein Entreprise Bean, das JMS Nachrichten asynchron prozessiert. Wenn eine Nachricht in eine Queue/Topic ankommt, ruft der EJB Container die `onMessage()` Methode des entsprechenden MDB (Implementierung von `javax.ejb.MessageDrivenBean`), der auf diese Queue wartet.

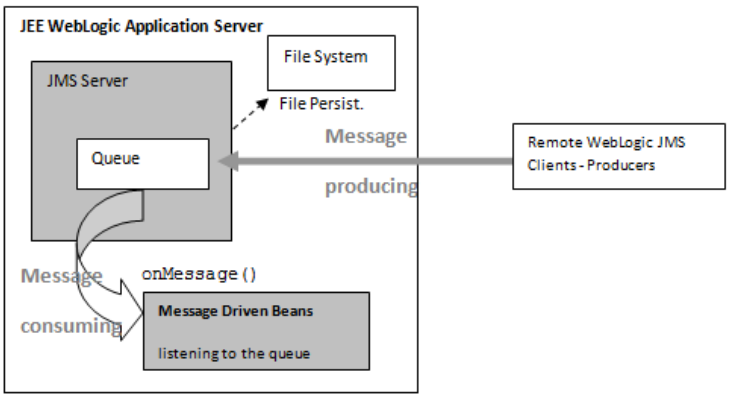
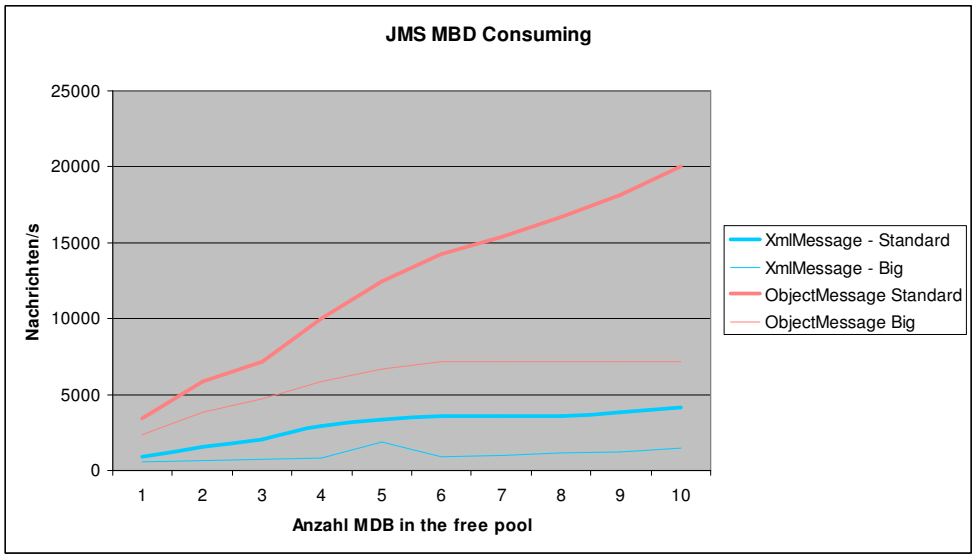


Abb. 7: Testarchitektur für die Evaluierung der WebLogic JMS Message-driven Beans

Die MDB sind in den WebLogic XML Files ejb-jar.xml und weblogic-ejb-jar.xml zu konfigurieren: Queue Name, Max Messages in Transaktion, Max Beans in Free Pool.

Der WebLogic Server EJB Container poolt MDB Instanzen im Speicher. Die Größe des Pools ist konfigurierbar mit den Parametern `<initial-beans-in-free-pool>` und `<max-beans-in-free-pool>` in weblogic-ejb-jar.xml. Wenn eine Nachricht in eine Queue ankommt, versucht der EJB Container einen freien MDB im Pool. Wenn keine MDB Instanz verfügbar ist, erzeugt er eine neue Instanz falls die aktuelle Pool Größe kleiner als `<max-beans-in-free-pool>` ist. Wenn der Pool schon seine maximale Größe erreicht hat, bleibt die Nachricht in der Queue bis ein MDB im Pool wieder verfügbar wird.



Ergebnis: Message Consuming von `ObjectMessage` lässt sich sehr gut durch MDB skalieren lassen

Vergleichsmatrix

a) Messaging

	WebLogic Server JMS	**Oracle Streams AQ 11g**
Storage des Queues	File System des Applikation Servers	Oracle AQ Tabelle
Remote Message Producing	Java clients	Oracle AQ Propagation mit mehreren Eingangsqueue
Local Message Consuming	Message-Driven Beans im Applikation Server	Dequeuing mit Oracle PL/SQL API
Optimal Nachrichtformat	JMS ObjectMessage	Oracle User Defined Type (UDT)
Backup	Bei Pufferung der Nachrichten im Dateisystem ist das Backup-Problem ungelöst	Standard Backup / Recovery einer Oracle Datenbank

b) Technische Nachrichtenverarbeitung

	WebLogic Server JMS	**Oracle AQ**
Ort der Nachrichtenverarbeitung	JEE Application Server	Oracle Datenbank

Umgebung der Verarbeitung	Message-driven Beans gesteuert bei dem EJB Container	Oracle Jobs (Oracle Session)
Programmiersprache	Java 6	PL/SQL 11g
Aspekte der Programmiersprache	Moderne Sprache: objekt-orientiert, Vererbung, Abstraktion, Polymorphismus, große API	Prozedurale Sprache, eher benutzt für ETL Ladenoperation als für komplexe Logik, objekt orientiert möglich mit Aufwand
Technologie Community **(open source, Frameworks)**	Sehr groß	Begrenzt
Netzwerklast für DB Read / Write	Hoch (JDBC)	Optimal (Verarbeitung läuft in der DB)
Skalierbarkeit	Hoch durch multi-threading mit Enterprise Java Beans und Clustering des Application Servers	Möglich durch Parallelisierung (parallel Job Sessions) des Dequeuings aber begrenzt da 1) der PL/SQL Engine ist nur ein von den vielen anderen Komponenten der Oracle Datenbank 2) die Oracle Datenbank läuft auf einen einzigen Server (kein Cluster)

Fazit und Entscheidungen für das Projekt

Grundsätzlich wären beide Technologien einsetzbar und würden den erforderlichen Durchsatz gewährleisten. Mit dem Einsatz von WebLogic Server JMS würde die Nachrichtenverarbeitung im Applikation Server stattfinden und der PasInSys Server würde eine mehr „State of the Art“ Architektur bekommen:

- klare Trennung zwischen Business Logik (im JEE WebLogic Server) und Datenpersistenz (in der Oracle Datenbank)
- erhöhte Skalierbarkeit durch multi-threading von Message-Driven Beans und Clustering
- Nachrichtenverarbeitung in Java statt PL/SQL: objekt-orientierte Programmierung, große API, open source Frameworks …
- weniger Last in der Datenbank

Da AQ jedoch Standardbestandteil des Oracle-Servers (kein Kosten) ist und die Auslagerung der Nachrichtenkomponente in einen Applikationserver eine komplexere Architektur mit eigener Sicherungsproblematik und erhöhten Netzwerklast zwischen WebLogic und die Datenbank zur Folge hätte, kommt Oracle AQ im Einsatz.

Der Datenaustausch zwischen dem PasInSys Server und eng benachbarten Schnittstellenpartnern geschieht im Altsystem unter Verwendung von XML, was wegen der erforderlichen XML-Synthese und -Analyse unter Berücksichtigung der stark gestiegenen Durchsatzanforderungen des neuen Systems zu aufwändig wäre. Deshalb wird der Einsatz von Oracle-PL/SQL-Objekten unter Verwendung des JPublishers vorgesehen.

Diese vorgesehene Technologie bewirkt eine deutliche Effizienzsteigerung, aber auch eine enge Koppelung der Projekte. Über Schnittstellen, für die eine solche enge Koppelung nicht akzeptabel ist, sind Daten per XML auszutauschen. Dafür kommen die Oracle XDB Features im Einsatz um Oracle Objekten in XMLType oder umgekehrt umzuwandeln: Mapping zwischen die UDTs und XML Strukturen in XML Schemata definieren, Benutzung der Member Funktionen von XMLType (toobject, createxml), Definition von XMLType Views basiert auf Objekt Views die die Daten aus den relationalen Tabellen selektieren.

Kontaktadresse:

Bertrand Caradec
Logica Deutschland GmbH & Co. KG | Now part of CGI
Am Limespark 2
D- 65843 Sulzbach (Taunus)

Telefon: +49 (0) 151 16360433
E-Mail: bertrand.caradec@logica.com | bertrand.caradec@cgi.com
Internet: www.logica.de | www.cgi.com