Integration Patterns mittels OSB - Beispiele aus der Praxis

Ulrich Haug
CGI (Germany) GmbH & Co. KG
Am Limespark 2 | 65843 Sulzbach (Taunus) | Germany

Schlüsselworte
Oracle Service Bus, Integration, JMS, Claim Check, Message Exchange Patterns

IHR ABSTRACT:
Dieser Vortrag zeigt an Beispielen wie unterschiedliche Integrationsszenarien mit dem Oracle Service Bus umgesetzt werden können. Verwendete Patterns sind u.a. Request-Response, Umsetzung von Oneway mit JMS Puffer, Request-Callback Claim-Check, Shared-Database Pattern.

Der Vortrag richtet sich an Interessierte mit Grundkenntnissen im Bereich Oracle Service Bus. Grundsätzliche Artefakte im OSB mittels Proxy und Business Service, Soap/http, JMS, WSDL, XML, XSD, XSLT, xQuery sollten bekannt sein.

Einleitung

Dieser Vortrag zeigt an Beispielen wie unterschiedliche Integrationsszenarien mit dem Oracle Service Bus umgesetzt werden können. Verwendete Patterns sind u.a. Request-Response, Umsetzung von Oneway mit JMS Puffer, Request-Callback Claim-Check, Shared-Database Pattern.

Der Vortrag richtet sich an Interessierte mit Grundkenntnissen im Bereich Oracle Service Bus. Grundsätzliche Artefakte im OSB mittels Proxy und Business Service, Soap/http, JMS, WSDL, XML, XSD, XSLT, xQuery sollten bekannt sein.

Integration Patterns mittels OSB - Beispiele aus der Praxis

Der Oracle Service Bus erlaubt es, eine Vielzahl an Nachrichten-Austausch Patterns (Message Exchange Patterns) in einfacher Weise umzusetzen. Gerade wenn die anzubindenden Systeme unterschiedliche Transport und Nachrichtenprotokolle unterstützen, zeigt der Enterprise Service Bus seine Stärken.
Für die Kommunikation zwischen unterschiedlichen Systemen kennt man Message Exchange Patterns wie

- Request-Response (synchron),
- Oneway (aka Fire&Forget),
- Request-Callback (asynchroner Request-Response),
- Publish & Subscribe.

Auf einfachste Weise lassen sich Integrationen mit dem Oracle Service Bus über das synchrone ***Request-Response Pattern*** abbilden. Die Vorteile liegen klarerweise auf der Einfachheit der Anbindung, gerade bei Transportprotokoll soap/http.
Innerhalb des Oracle Service Bus sollte der Messageflow dem ***VETRO Pattern*** genügen: Das Pattern steht für **V**alidate – **E**nrich – **T**ransform – **R**oute und schließlich **O**perate (Ausführung des Service).
Gerade die Validierung der eingehenden Nachrichten bei der Integration sollte berücksichtigt werden. Auch eine Response ist ein eingehende Nachricht für den Oracle Service Bus und für den Service Provider! Es macht keinen Sinn eine invalide Nachricht weiterzureichen. Abgesehen von unverständlichen Nachrichten könnten diese auch maligne Intensionen verfolgen.

- Ein invalider Request wird direkt zurückgewiesen.
- Eine invalide Response wird nicht weitergeleitet.

In beiden Fällen empfiehlt es sich bei der Integration über den Oracle Service Bus, einen entsprechenden Errorhandler zu definieren, den Fehler zu fangen und in geordneter Weise eine Fehlermeldung zurückzugeben (selbstverständlich verlangt dies auch entsprechende Soap Faults, die für die Kommunikation dieser Fehler genutzt werden können). Aus diesem Grund werden die Validierungen in eigenes Stages mit zugehörigen Errorhandlern umgesetzt. Eine Abweichung von dieser Regel sollte nur in begründeten Fällen geschehen. Gründe hierfür können zu komplexe Validierungen (rechenintensiv) sein.
Unabhängig vom Stil des Message Exchange Patterns empfiehlt sich eine grundsätzliche obige geschilderte Validierung der Nachrichten. Der Umgang mit invaliden Nachrichten unterscheidet sich jedoch von Pattern zu Pattern.
Das synchrone Request-Response Pattern verlangt auf der anderen Seite, dass beim Aufruf des Providers diese auch zur Verfügung steht. Diese zeitliche Kopplung kann über weitere Message Exchange Patterns aufgelöst werden.
Manchmal muss mittels des Request-Response Patterns (aber auch bei den folgenden Austauschs-Szenarien) eine sehr große Datenmenge übertragen werden. Dazu bietet der Oracle Service Bus Mechanismen an, um diese effizient zu behandeln. Spaltet sich die Nachricht in einen fachlichen Anteil (soap Body) und ein Attachment auf, das für die Bearbeitung innerhalb des Oracle Service Bus nicht berücksichtig werden muss (also nicht in-memory benötigt wird), so kann man für diese Behandlung das ***Claim Check Pattern*** verwenden. Dabei wird beim Eingang der Nachricht die große Datenmenge z.B. auf einem Filesystem abgelegt, dieser ein Identifier zugewiesen und beim Verlassen des Service Bus anhand des Identifiers die große Datenmenge geholt und dem Service Provider zur Verfügung gestellt.

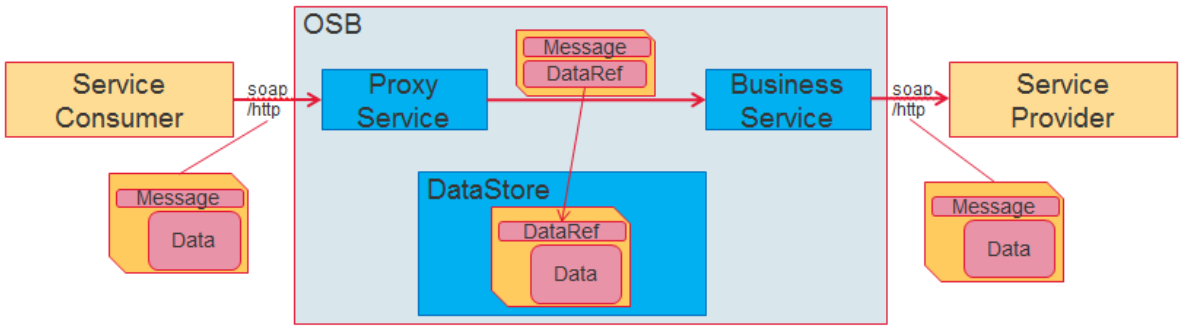

Abbildung 1 Claim Check Pattern

Oftmals möchte man aber nur eine Nachricht, ein Ereignis an einen Provider schicken. Dazu bedient man sich dem ***Oneway Pattern***. Beispiele hierfür sind Statusmeldungen oder Events (wobei Events durchaus auch ein Publish & Subscribe Pattern bedingen). Dieses Pattern kann aber auch genutzt werden, um die Kopplung zwischen einer Message Oriented Middleware wie IBM MQ oder JMS mit einer soap/http Schnittstelle zu realisieren.
Wird über soap/http eine Oneway Nachricht an den Oracle Service Bus geschickt, so ist bei einer Oneway Kommunikation dem Nachrichten Sender nicht klar, dass die Nachricht wirklich beim Message Receiver angekommen ist. Insbesondere besteht die Gefahr, dass der Message Receiver gar nicht zum Zeitpunkt der Nachrichtenversendung zur Verfügung steht und deshalb die Information der Nachricht verloren geht. Gleiches passiert, sofern der Oracle Service Bus während der Verarbeitung aus welchen Gründen auch immer abstürzt.

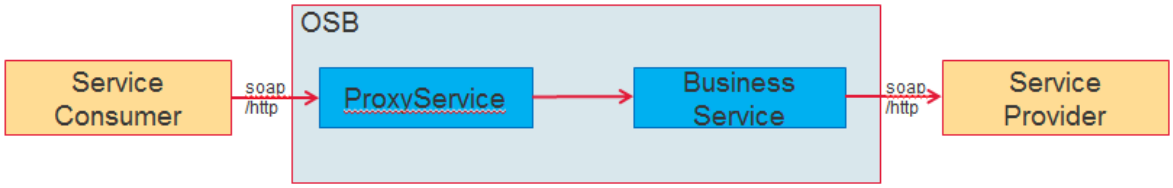

Abbildung 2 Oneway ohne JMS Pufferung

Aus diesem Grund empfiehlt es sich (sofern man Einfluss auf den Webservice nehmen kann), die Schnittstelle so zu gestalten, dass der Proxy Service

- zuerst die Nachricht in eine persistente JMS Queue einträgt und danach
- eine Quittierung an den Service Consumer schickt.

Die Quittung attestiert dem Konsumer, dass die Nachricht erfolgreich an den OSB übermittelt wurde und nun in der Folge die Auslieferung der Nachricht über den OSB gesichert wird. Schlägt die Auslieferung an den OSB fehl, so wird eine negative Quittung zurückgegeben. Weiterhin ist bei soap zu beachten, dass eine Oneway Kommunikation keinen Soap fault erlaubt. Die Umgestaltung der Schnittstelle in eine Request/Response Schnittstelle eröffnet hierbei wieder die Möglichkeit auch Faults zu senden.
Die Nachricht wird intern in eine JMS Queue persistent eingestellt und erst dann die erfolgreiche Entgegennahme der Nachricht durch den Oracle Service Bus mittels der Soap Response quittiert wird. In der Folge wird die Nachricht mittels eines Proxy Service aus der Nachrichten Queue geholt und an den Nachrichten Konsumer geschickt. Mittels dieses Mechanismus erhöht sich der Quality of Service, da man sich die Funktionalitäten von JMS zu nutzen macht.

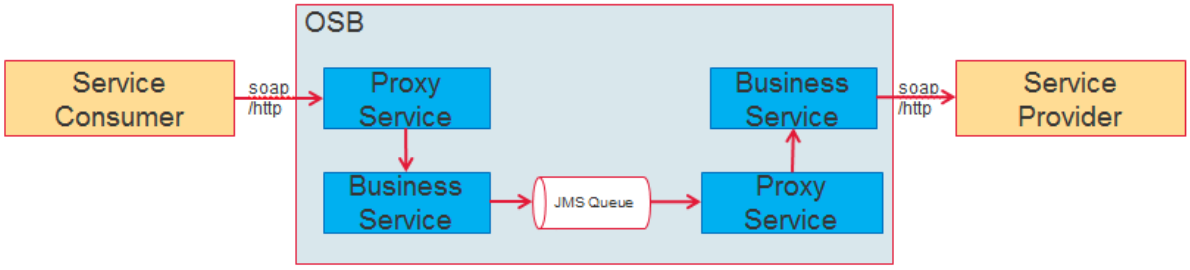

Abbildung 3 Oneway mit JMS Pufferung und Quittierung

Erwartet jedoch der Service Konsumer eine fachliche Antwort (also nicht nur obige Quittierung bei Oneway) und dauert deren Berechnung zu lange oder ist die Verfügbarkeit des Service Providers und Consumers zeitlich nicht gewährleistet, so bietet sich das ***Request-Callback Pattern*** (bekannt auch als asynchroner Request-Reply) an. Solche Integrationsszenarien, die asynchrone Kommunikation verlangen, lassen sich sehr gut mittels JMS über den Weblogic und Oracle Service Bus realisieren. Gerade im Fall von nicht vorhandenen Adaptern und bei fehlender Unterstützung soap/http oder „Abneigung“ gegen diese Technologie kann eine JMS Integration bei existierender Unterstützung durch das Backend System eingesetzt werden (Beispiele für Backendsysteme: Application Server mit Message Driven Beans, etc.).
Aber auch bei der internen Nachrichtenverarbeitung im Oracle Service Bus lässt sich die einfache JMS Integration nutzen, um weitere Integrations-Szenarien umzusetzen. Den Einsatz von JMS kann man im OSB für gesteigerte Quality of Service Funktionalität bei Service Aufrufen nutzen. Hierzu zählt unter anderem Etablierung von Retry Mechanismen und Reliable Messaging (siehe oben z.B. Oneway mit JMS Puffer). Bevor nun mögliche Umsetzungen von Request-Callback Pattern besprochen werden, werden grundsätzliche Patterns bzgl. der Nutzung von Messaging angerissen.

Ein Nachrichten Sender schickt über eine JMS Queue eine Nachricht an einen Empfänger. Mittels dem OSB kann ein Service Provider nach außen JMS als Transport Protokoll nutzen, obwohl er selbst ein anderes Transportprotokoll spricht (natürlich gilt auch gleiches für den Service Consumer).

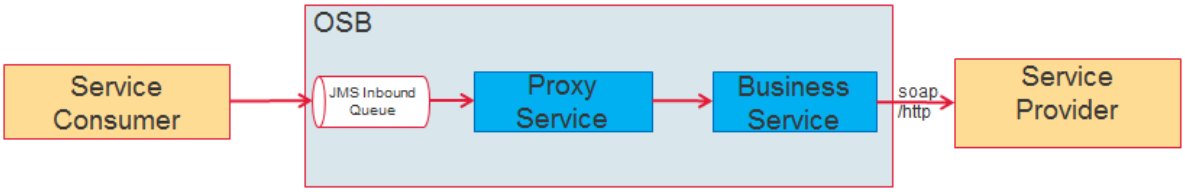

Abbildung 4 JMS Anbindung

Grundsätzlich empfiehlt es sich, dass alle Nachrichten, die über Messaging versendet werden, auch einem bestimmten Datenformat genügen. Die Nachrichten müssen als XML beschrieben werden und für jeden Nachrichtentyp muss ein zugehöriges XML Schema definiert werden.
Diese Forderung an die Nachrichten Struktur führt uns zum ***Datatype Channel Pattern***: Für die unterschiedlichen Nachrichten Datentypen werden dann zugehörige Queues / Topics angelegt. Jeder Empfänger kann sich genau auf den erwartenden Nachrichten Inhalt verlassen. Weiterhin kann man für die unterschiedlichen Nachrichten Typen damit auch unterschiedliche Qualitätsstufen beim Messaging definieren. Beispiele hierfür sind, persistente Queues/Topics versus non-persistente, Throttling von Nachrichten, etc.
Obige Forderung bzgl. XML Struktur der Nachricht samt XML Schema ermöglicht es, invalide Nachrichten zu identifizieren. So werden die Nachrichten Inhalte auf Wohldefiniertheit und validen Inhalt überprüfen. Verlangen die Nachrichten Empfänger gar bestimmte Nachrichten Header Attributwerte wie Nachrichten Correlation ID, so müssen diese auch überprüft werden. Wird zum Beispiel in einem *Request-Callback Pattern* die Return Adresse vergessen, so ist die Nachricht zwar bearbeitbar, aber der Aufrufer bekommt keine Antwort. Diese Art von fehlerhaften Nachrichten müssen aussortiert werden. Auch der über JMS und OSB einfach etablierbare Retry-Mechanismus macht für solche Nachrichten keinen Sinn. Stattdessen kommen die Nachrichten direkt in eine Invalid Message Queue (***Invalid Message Channel Pattern***) und belasten so nicht weiter die JMS Infrastruktur.

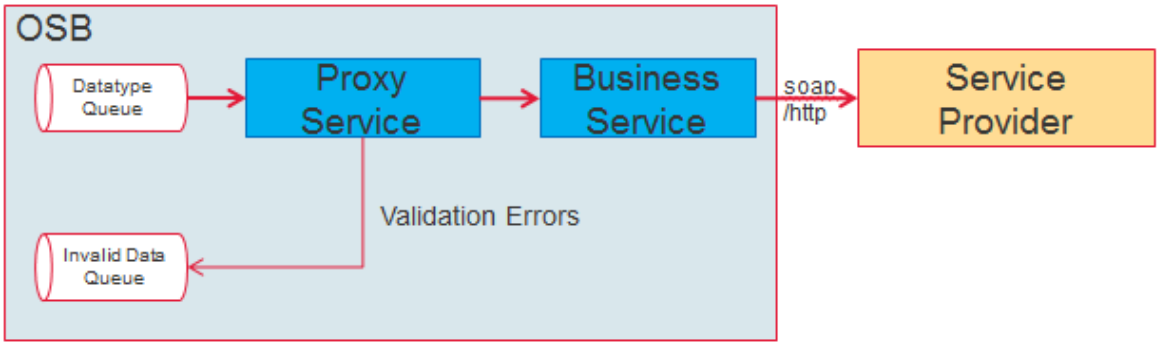

Abbildung 5 Invalid Messsage Channel

Die *Invalid Message Channel* Queue sollte ein Nachrichtenformat erlauben, dass neben der reinen Payload auch Metadaten beinhaltet, um den Kontext der Nachricht zu transportieren.
Auf die *Invalid Message Queue* kann ein weitere Proxy Service lauschen, der die Daten in eine dedizierte Datenbank einträgt. Dies kann über den JCA Datenbank Adapter des OSB realisiert werden.

Erhält der auf die Queue lauschende Proxy Services beim Aufruf seiner zu integrierenden Applikation einen Fehler aus der Applikation, so kann man im Allgemeinen zwischen zwei Strategien wählen:

1. Die Nachricht wird auch als ungültig angesehen und wird in die *Invalid Message Queue* eingetragen. (damit werden messaging errors mit application errors vermengt)
2. Da die Nachricht der Applikation bekannt ist, wird sie nicht in die Invalid Message Queue eingetragen und wird komplett verworfen (oder in eine *Application Error Queue* eingetragen)

Ist die zu integrierende Applikation nicht erreichbar, so kann über den JMS Retry Mechanismus versucht werden, die Nachricht mehrfach auszuliefern. Hierbei kann man auch die Strategie fahren, unterschiedliche Wiederholungsintervalle zuzulassen. Im ersten Wiederholungsintervall wird z.B. mit kurzen wiederholungspausen gearbeitet. Sind alle Wiederholungen aufgebraucht, so wird über den auf der JMS Destination definierten Redirect die Nachricht in eine weitere Queue eingestellt. Auf dieser ist ein Retry definiert, der längere Pausen zwischen den Wiederholungen definiert. Dieses Vorgehen lässt sich wiederholen.

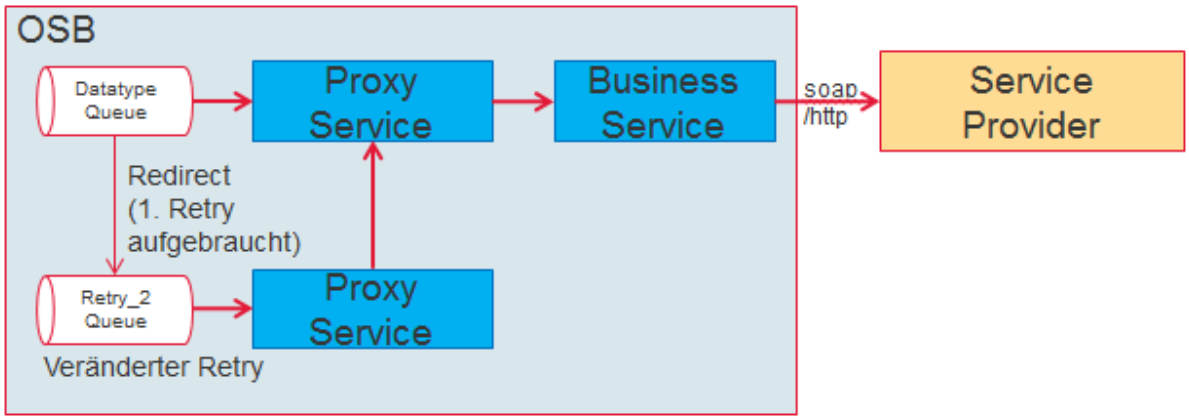

Abbildung 6 Retry Kaskade

Zu berücksichtigen sind hierbei unter anderem:

- Wann verliert eine Nachricht seine fachliche Gültigkeit und macht so eine verspätete Auslieferung sinnlos?
- Kann das empfangende System mit Duplikation umgehen bzw. ist der Empfänger idem potent?

Ist eine Nachricht in diesem Sinne technisch final **nicht** auslieferbar (Empfänger nicht erreichbar), so wird die Nachricht in eine *Dead Letter Queue* eingetragen (***Dead Letter Channel Pattern***). Wie oben ist diese Queue unbedingt zu überwachen.

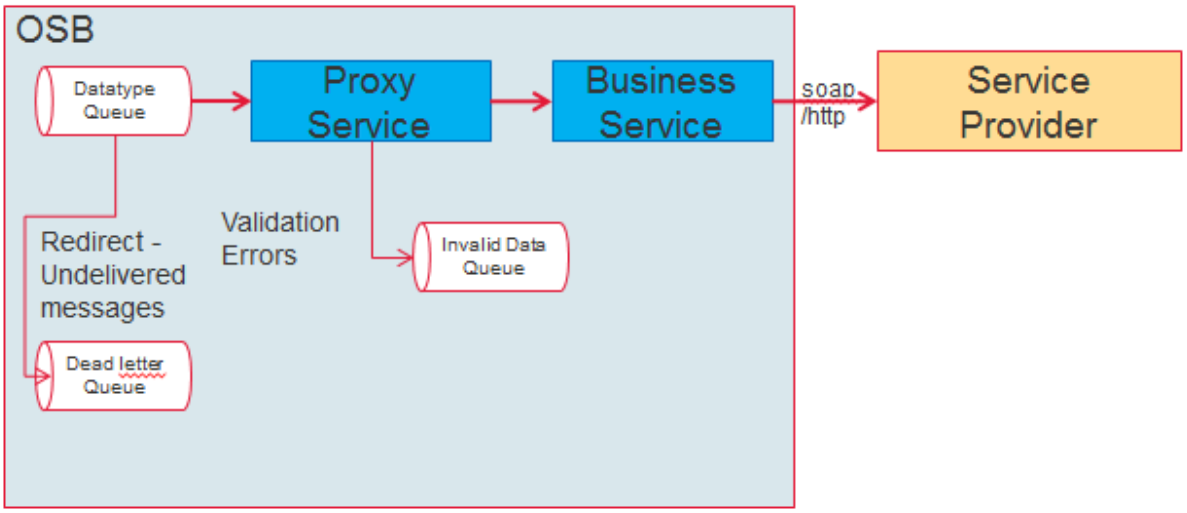

Abbildung 7 Dead Letter Channel

Das obig genannte *Datatype Channel Pattern* führt aber dazu, dass man ggf. eine sehr hohe Anzahl an Queues bzw. Topics erhält. Um diese Anzahl zu reduzieren, kann man ein generisches Nachrichten-Datenformat für bestimmte Nachrichtentypen definieren. Jede spezialisierte Nachricht wird mittels des ***Message Router Pattern*** an einen bestimmten Empfänger geschickt. Eine Nachricht im Format des generischen Datentyps wird in eine Eingangs-Queue eingestellt und von dort vom Message Router abgeholt. Der Message Router analysiert die Nachricht und stellt sie in eine Zielqueue ein oder schickt sie direkt an den entsprechenden Service Provider.

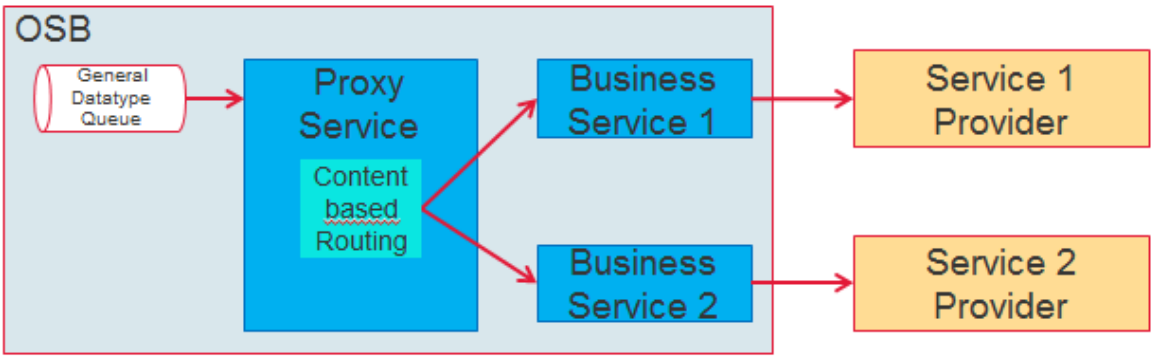

Abbildung 8 Message Router

Zu Bedenken sind aber dabei folgende Punkte:

- Dieses Pattern sollte nur bei einer feststehenden Anzahl an Empfänger Systemen umgesetzt werden.
- Die Nachrichten müssen analysiert werden und werden in eine weitere Queue eingestellt. Dies hat sicherlich Auswirkung auf das Laufzeitverhalten. Je nach Forderung der Reaktionszeiten kann dies ein zu großer Overhead sein.

- Bei komplexen Nachrichten Verarbeitungen im Proxy Service und dem Wunsch Retry zu nutzen (Nachricht wird erneut aus der General Datatype Queue geholt) empfiehlt es sich, das Routing zu den Service Providern über weitere Queues zu entkoppeln. Damit wird das

Errorhandling für die unterschiedlichen Service Providern voneinander entkoppelt (Bem.: Konterkariert natürlich in gewissen Maßen die Motivation einer General Datatype Queue):

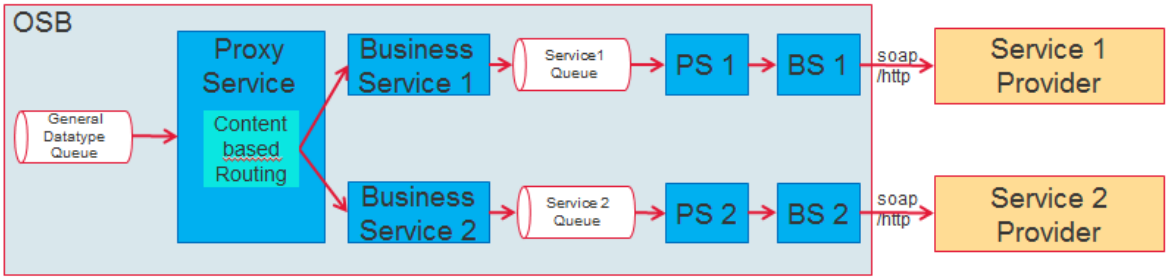

Abbildung 9 Message Router mit Queues

Nach obiger Darstellung der unterschiedlichen Aspekte, die grundsätzlich bei JMS Kommunikation beachtet werden sollten, kann das *Request-Callback Pattern* mittels JMS umgesetzt werden (obige Details sind zur Vereinfachung weggelassen. Der Service Consumer schickt eine CorrelationsID mit, um die Antwort der Anfrage zuzuordnen. Ebenso wird bei dynamischer Zuordnung des Consumers die entsprechende JMS Destination URI mitgeschickt. Eine dynamische Zuordnung funktioniert nur, sofern die Zugriffsrechte für jede Response-Queue die gleichen sind. Enthält der Request nicht Correlation ID und JMS Destination, so ist die Nachricht falsch und wird in die *Invalid Message Queue* verschoben.

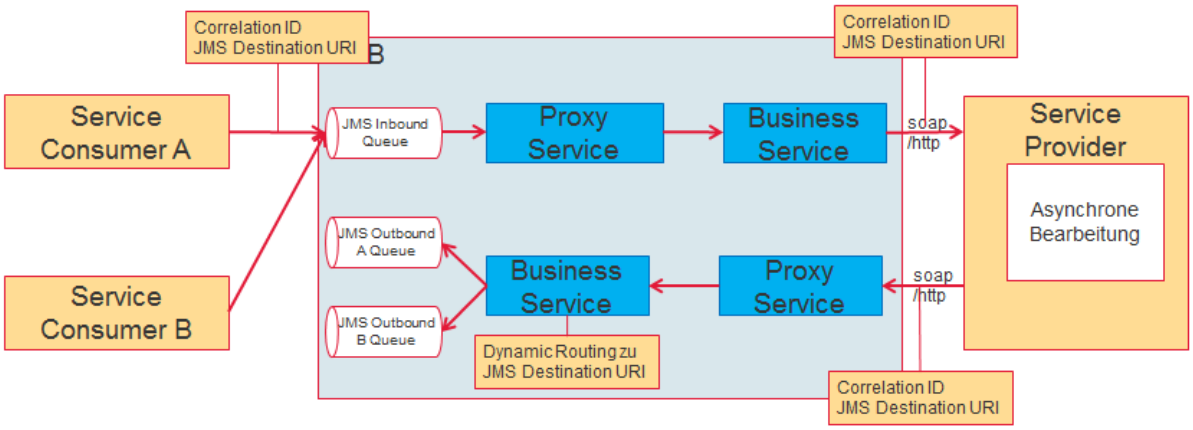

Abbildung 10 Request-Callback Pattern mit JMS mit dynamischer Destination Zuordnung

Werden die Response Nachrichten zu groß, so kann die Response unterteilt werden. Dies bedeutet, dass man den Nachrichten noch weitere Metadaten zuordnen muß:

- SequenzID: Kenner der aktuellen Nachrichten Nummer
- MaxSequenzNumber: Maximale Anzahl der Nachrichten, so dass die Vollständigkeit der Gesamt-Nachricht nachvollzogen werden kann.

Mittels dieser Daten kann die Payload aufgespalten und Vollständigkeit beim Empfang überprüft werden. Fehlende Nachrichten führen zur Aussortierung. Hier stellt sich die Frage, wie lange auf eine ausstehende Nachricht aus der Sequenz gewartet werden kann.
Die Sequenzierung der Nachrichten führt zu einem weiteren Problem bei Integration, nämlich der Einhaltung der Nachrichten Reihenfolge. Aus Skalierungsgründen sollte eine mögliche Einhaltung der Nachrichtenreihenfolge nicht im Oracle Service Bus umgesetzt werden.

Bieten die zu integrierenden Systeme keine Schnittstellen wie soap/ http oder JMS an, so kann eine Integration auch über Datenbank geschehen. Dieses Pattern nennt man ***Shared Database Pattern***. Hierzu kann man direkt den JCA DB Adapter des OSB nutzen.

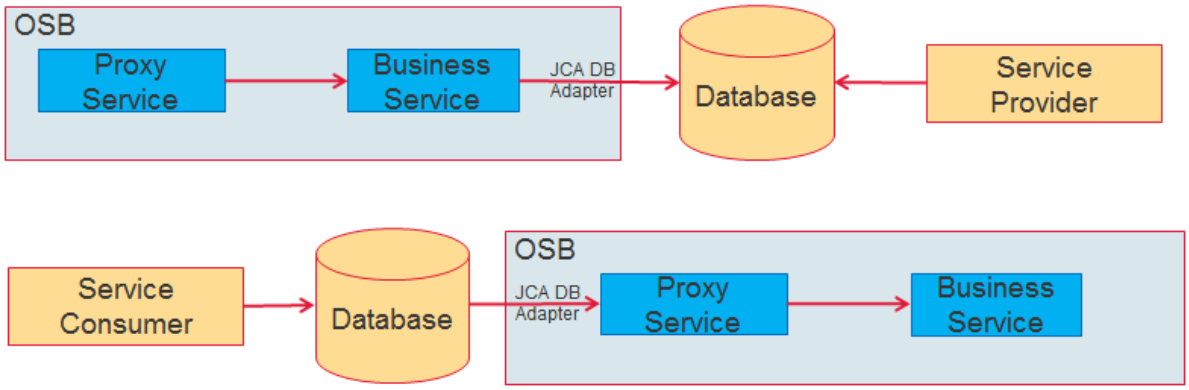

Abbildung 11 Shared Database Pattern

Lauscht ein Proxy Service mittels JCA Adapter auf eine Datenbank Tabelle, so werden oft die empfangen Daten markiert oder gar gelöscht, um beim nächsten Lesen nicht diese erneut zu lesen. Aus diesem Grund empfiehlt es sich, die gelesenen Daten in eine persistente JMS Queue einzutragen. Dies garantiert, dass unabhängig vom Transport Protokoll des Zielsystems erstmal die Daten sicher dem OSB übergeben wurden.

Obige Pattern haben einen Einblick gegeben, wie man mittels des Oracle Service Bus unterschiedliche Integrations Patterns umsetzen kann. Die Mechanismen des Oracle Service Bus erlauben uns durchaus komplizierte Anforderungen an die Kommunikation rasch und ohne zu großen Aufwand umzusetzen. Selbstverständlich gibt es bei Integration noch weitere, viel komplizierte Anforderungen, die jedoch nicht in diesem Rahmen betrachtet werden können.

Kontaktadresse:
Ulrich Haug
CGI (Germany) GmbH & Co. KG
Am Limespark 2
65843 Sulzbach (Taunus)

Telefon: +49 (0) 170 5791 256
Fax:
E-Mail ulrich.haug@cgi.com
Internet: www.cgi.com