Integration von Telefonie in Oracle Fusion Middleware Applikationen

Lyubomir Yordanov - Yordanov Consulting
Ralf Ernst, Bundesagentur für Arbeit, Nürnberg

Schlüsselworte
ADF, Active Data Service, OSB, WLS, Telefonie, Systemarchitektur

Einleitung
Die Integration eines Client-Computers mit einer unternehmensweiten Telefonanlage ist eine für die Mitarbeiter oft sehr wichtige Funktionalität, für die bereits seit vielen Jahren etablierte Lösungen existieren. Die bestehenden Lösungen basieren hierbei meist auf Client-APIs der jeweiligen Telefonie-Hersteller.
Eine immer wichtigere Bedeutung für die Unternehmenswelt gewinnen allerdings WebAnwendungen, die Desktop Anwendungen und s.g. Fat-Clients immer mehr verdrängen. Aus Sicht eines Mitarbeiters muss eine WebAnwendung allerdings dieselben Use-Cases ermöglichen wie eine traditionelle Desktop Anwendung.
Die z. B. aus Outlook oder Lotus Notes gut bekannte Integration des Telefons gehört auch dazu. Aus der Tatsache, dass Web-Anwendungen in großen Unternehmen in hochkomplexen, verteilten Systemarchitekturen deployed und betrieben werden entstehen weitere Herausforderungen. In diesem Manuskript wird ein Lösungsansatz für eine Integration einer heterogenen Telefonie-Infrastruktur in bestehende Web-Anwendungen basierend auf den Produkten Oracle Service Bus, Weblogic Server und Oracle ADF vorgestellt. Dabei wird auch die Nutzung der sogenannnten „server-push“ Technologie vorgestellt, die der Oracle Active Data Service zur Verfügung stellt.

Beweggründe / Ausgangslage
Das IT-Systemhaus der Bundesgentur für Arbeit (BA) betreibt eine der größten IT-Systemlandschaften Deutschlands. Neben Standardsoftware für client-zentrierte Office Funktionalitäten, einem ERP System für Auszahlungen und Verwaltung und einer Business Intelligence Plattform für statistische Auswertungen zählen hierzu rund 100 zentralisierte Applikationen mit mehr als 10000 Masken, die die spezielle Geschäftslogik der BA implementieren.

Die mehr als 150.000 Benutzer dieser Applikationen müssen zur Erledigung Ihrer Geschäftsprozesse eine Vielzahl von Anwendungen nutzen, die eine Integration mit der Telefonie-Infrastruktur benötigen z.B. zwecks automatisiertes Anzeigen von Kundendaten aus der Telefonnummer eines eingehenden Anrufs oder Initiieren von Anrufen direkt aus WebAnwendungen heraus.
Diese Komfort-Funktionalitäten können einem Mitarbeiter durchschnittlich 10 Sekunden für das manuelle Anwählen am Apparat oder Suchen von Benutzerdaten in der WebAnwendung ersparen. Bei einer durchschnittlichen Anzahl von 10 Anrufen am Tag, an denen die 150.000 Mitarbeitern teilnehmen, hat die Organisation um die 1 Million Stunden und dadurch 30 Millionen Euro Einsparpotential.

Nicht-funktionale Anforderungen
Aus Sicht der Benutzeroberfläche müssen die graphischen Komponenten, die die Integration der Telefonanlagen mit den WebAnwendungen ermöglichen, mit der generellen Strategie der unternehmensweiten Standardisierung und Vereinheitlichung der graphischen Benutzeroberflächen im Einklang stehen.

Aus Sicht der Software Architektur muss die gewünschte GUI Komponente als in unterschiedlichen WebAnwendungen wiederverwendbare, generische Komponente bereitgestellt werden, um die einfache Integration der gewünschten Funktionalität in einer Vielfalt von mit Oracle-ADF erstellten Web-Anwendungen zu ermöglichen.
Zum Anderen soll bei der zu erwartenden Datenmenge, die durch die Gesamtlösung übermittelt werden, eine effiziente Integration in die vorhandene Systemarchitektur der Middleware und die vorhandene Telefonie-Infrastruktur erreicht werden.
Die bereits vorhandenen Konzepte für Hochverfügbarkeit, Lastverteilung usw. sollen die produktive Betreibbarkeit unterstützen.
Organisatorisch gesehen sind die Betreiber der Telefonanlagen und der IT getrennt. Zudem betreibt die BA aktuell eine Telefonie-Infrastruktur von zwei verschiedenen Providern. Die Lösung soll eine möglichst große separation-of-concerns beider Parteien ermöglichen, d.h. Lifecyclemaßnahmen oder Infrastrukturänderungen der Telefonie-Welt darf keinerlei Einfluss auf die Middleware-Welt haben und umgekehrt.
Das Gesamtsystem soll dabei eine beinahe Echtzeit Performance gewährleisten – sobald das Telefon des Mitarbeiters klingelt muss die Information, dass ein Anruf eingeht, am Bildschirm angezeigt werden (<50-75ms). Eine gesicherte Zustellung ist dabei nicht notwendig – 15 Sekunden nach einem eingehenden Anruf ist die visuelle Information dafür bereits wertlos.
Aus Security Sicht sind die Anforderungen relativ klein – beide teilnehmende Systeme (Telefonieinfrastuktur und IT-Middleware) befinden sich in der Organisation selbst. Eine Authentifizierung beider Komponenten mittels Basic Authentication ist ausreichend.

Infrastruktur
Die Telefonieinfrastuktur der BA ist heterogen und verteilt. Es existieren bundesweit z.Zt. 11 Cluster von Siemens Enterprise Networks (SEN) und mehrere Avaya Cluster. Diese Cluster sind ständigen Änderungen unterworfen. Hierzu zählen z.B. Lifecycle, Verteilung, Organisationsänderungen, gegebenenfalls sogar Providerwechsel usw..

Die IT Systemarchitektur (BA Middleware), in der die WebAnwendungen betrieben werden, basiert auf den Produkten des Oracle Fusion Middleware Stacks. Backbones sind Oracle WebLogic Server (WLS) und Oracle Service Bus (OSB). In der Systemarchitektur ist vorgesehen, dass die Anwendungen in s.g. „Middleware Zellen“ betrieben werden. Jede Middleware Zelle besteht aus physikalisch und software-technisch identischen – Enclosures mit mehreren, mit Infiniband verbundenen x86-Bladeservern. Eine Zelle beherbergt also alle BA-Anwendungen und SOA Services, die auf eigene WLS-Cluster deployed sind.
Eine besondere Domäne in der Zelle stellt der Enterprise-Service-Bus (Oracle OSB) dar, der als zentrale Integrations- und Kommunikationsplattform zwischen den einzelnen Anwendungen und den SOA-Services dient. Das bedeutet insbesondere, dass es keine direkte Kommunikation zwischen Benutzer und OSB gibt – das Routing der Clients und die dabei notwendige Session-Affinität für WebAnwendungen wird über Hardware-Loadbalancer der Fa. Citrix gewährleistet, die die http-Requests zu den jeweiligen Zellen routen.
Die Nutzung von SOA-Services erfolgt dabei über den Enterprise-Service Bus der jeweiligen Middleware-Zelle.

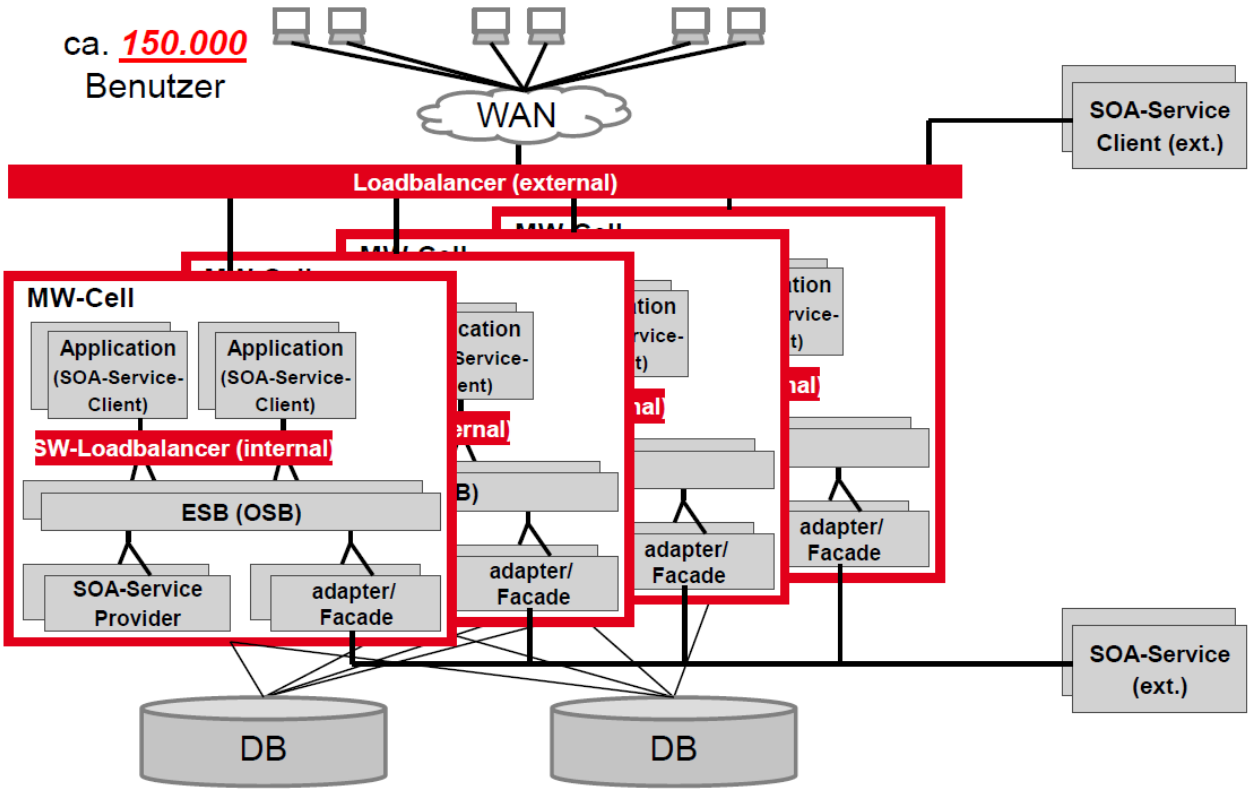


Abb. 1: BA Middleware - Systemarchitektur

Die gesamte Netzwerkkommunikation findet damit innerhalb der Zellen über Infiniband statt. Der Ansatz ähnelt damit ein wenig der Exalogic Server Familie von Oracle.

Lösungsarchitektur
In der Lösungsarchitektur werden zwei Use Cases betrachtet:

- **Eingehender Anruf** – Bei einem eingehenden Anruf wird der Benutzer in seiner aktuellen WebAnwendung über ein Pop-up benachrichtigt. Ein Klick auf das Pop-up ermöglicht es eine Aktion in der jeweiligen Anwendung auszuführen also z.B. die Benutzerdaten des Anrufers auf dem Bildschirm anzuzeigen.
- **Ausgehender Anruf** – Der Benutzer ist in der Lage, einen Anruf aus seiner aktuellen WebAnwendung heraus zu initiieren und muss die Nummer nicht manuell auf seinem Telefon anwählen.

Use Case 1: Eingehender Anruf
TAPI oder CSTA sind eine Art State-Machine und erfordern eine zusandsbehaftete Verbindung zwischen der Middleware und der Telefoninfrastruktur. Die Anforderungen dieses Use Cases sind allerdings über einen publish-subscribe Pattern am besten zu erreichen – die Telefonieinfrastruktur „publisht“ eine Nachricht und die WebAnwendungen „subscriben“ sich auf diese über die Middleware:

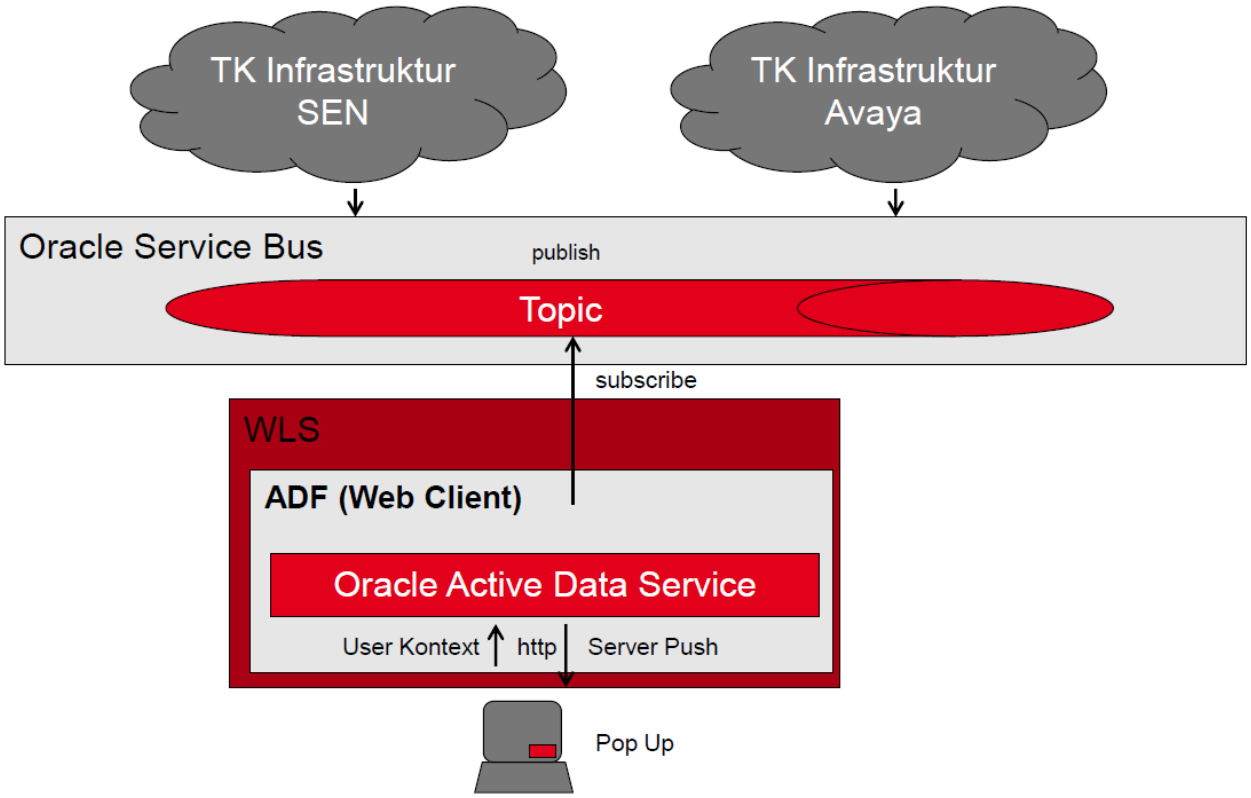


Abb. 2: Use Case 1 - eingehender Anruf

Dabei kommuniziert die TK-Infrastruktur über eine SOAP/http Schnittstelle am OSB mit der Middleware. Eine verteilte Message Queue (JMS Topic) ist jeweils identisch in jeder OSB Domäne aller Middleware Zellen konfiguriert. Subscriber auf die JMS Topics sind die WebAnwendungen in den unterschiedlichen WLS Domänen. Auf jeder der vielen, redundanten WLS Domänen sind also Teilmengen der Benutzer der WebApplikationen angemeldet. Da die Middleware-Zellen voneinander unabhängig quasi autark konzipiert sind, muss also ein Mechanismus implementiert werden, der die Nachricht in alle Zellen weiterleitet, da man ja nicht weiß in welcher Zelle sich der jeweilige auf die Nachricht lauschende Benutzer (subscriber) befindet.
Die Schnittstelle ist ein „Proxy Service“ am OSB, der über einen zellenexternen Hardware-Loadbalander adressiert wird. Da so die Nachricht nur in eine der Middleware Zellen landet, muss der OSB mittels seiner Split-Join Funktionalität und entsprechender Message Flows einen Fan-Out zu den OSBs aller anderen Middleware Zellen realisieren und so die Nachricht parallel in alle anderen Zellen weiterleiten.
Eine Nachricht aus der Telefonieanlage ist sehr einfach aufgebaut – in ihrem SOAP-Body hat sie nur zwei Daten – die Nummer des Anrufers und die Nummer des Angerufenen. Empfangen wird diese weitergeleitete Nachricht von s.g. „private“ Proxy Services und dann zu „OSB Business Services“, die diese zu JMS umwandeln. Zusätzlich werden im Message Flow zwischen den beiden – alle benötigten JMS Metadaten gesetzt – z.B. wird die Nummer des Angerufenen aus dem Payload der Nachricht ausgelesen und als JMS Property gesetzt.
Um die GUI Komponente im Browser eines Benutzers einer WebAnwendung über den Anruf zu informieren wird ein server-push über den Oracle Active Data Service gemacht. Da die Nummer des Angerufenen als JMS Property im OSB gesetzt wurde kann dediziert nur der betroffene Benutzer informiert werden und es fließen keine unnötigen Notifications durch die Middleware:

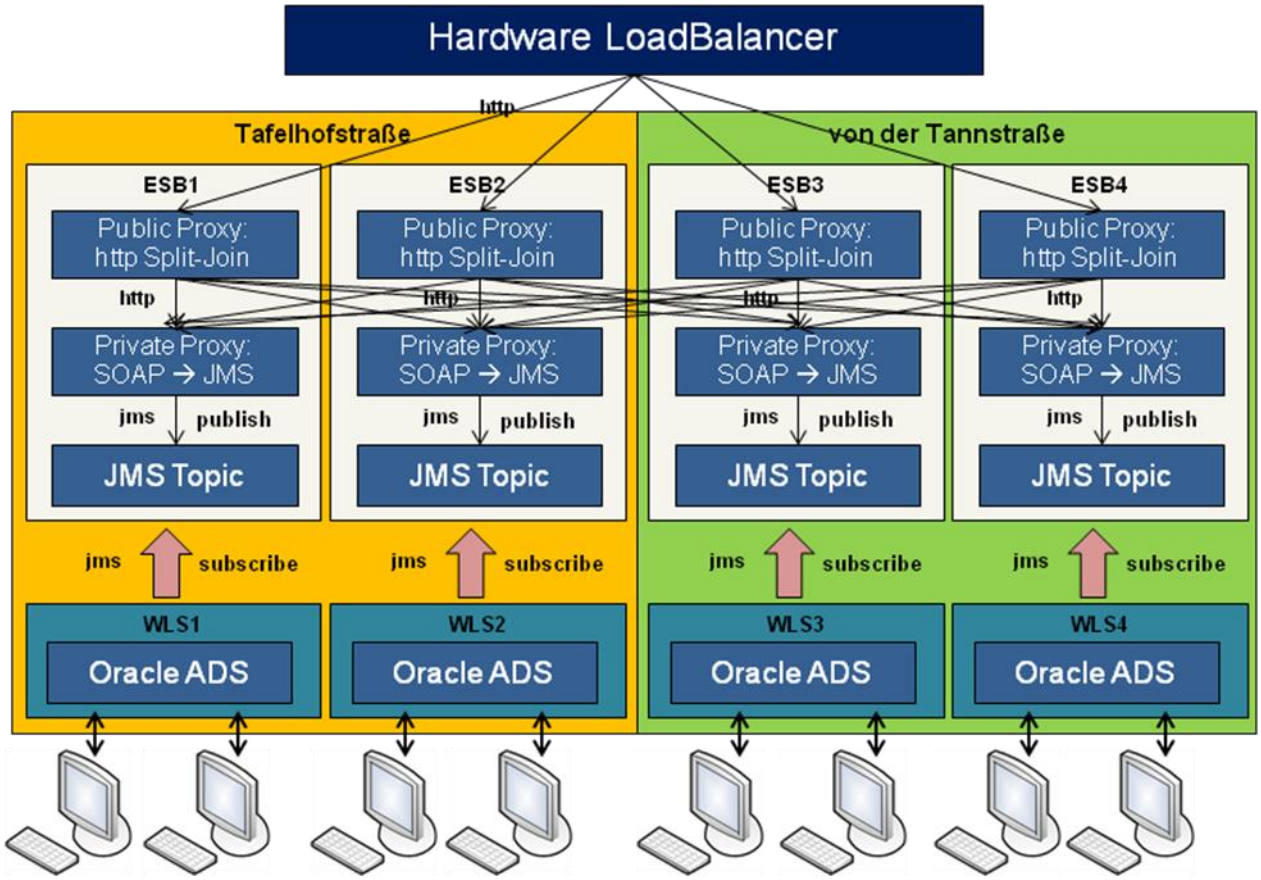


Abb. 3: Systemarchitektur - eingehender Anruf

Die OSB Komponenten und die „public“ und „private“ Proxy Services des OSB realisieren den in der BA-Systemarchitektur notwendigen fan-out Mechanismus: Der eigentliche Server-Push der JMS Nachricht zu der ADF-basierten WebAnwendung wird durch den Oracle Active Data Service realisiert:

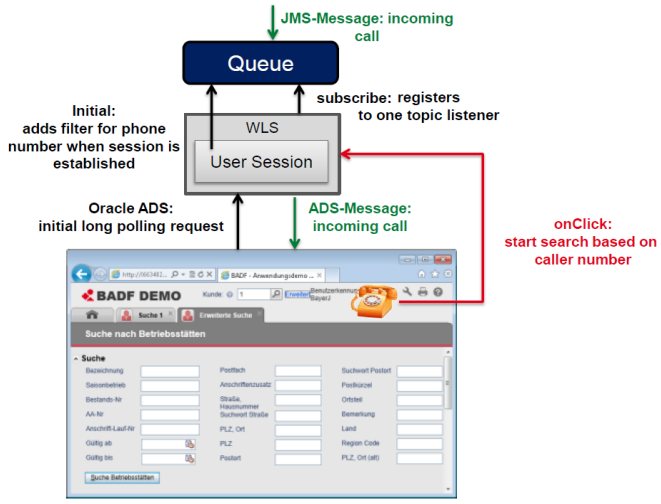


Abb.4 Server Push mit Oracle ADS

Wenn der Benutzer auf das Pop-up klickt wird z.B. eine Suche basierend auf der Nummer des Anrufes angetriggert, um die Daten des Anrufers auf dem Bildschirm anzuzeigen. Um die GUI-Komponente generisch zu halten ist hier aber nur ein Hook implementiert, den die nutzende Applikation ausimplementieren muss.

Use Case 2: Ausgehender Anruf
Für den zweiten Use Case – Mitarbeiter initiiert einen Anruf aus seiner WebAnwendung heraus - wird das Fire-and-forget Patter eingesetzt. Dabei wird aus der BA-Middleware ein SOAP Service der jeweiligen Telefonieinfrastruktur aufgerufen. Um die zwei Systemen voneinander zu entkoppeln wird auch an dieser Stelle einen Loadbalancer vor der Telefonieinfrastruktur eingesetzt. Der Service Contract (WSDL) beider Telefonieinfrastrukturen ist gleich. Die BA-Middleware schickt die Daten des Anrufs parallel zu beiden, ohne Rücksicht zu welcher Anlage genau die Nummer gehört – Avaya oder SEN. Es ist Aufgabe des Telefonieproviders eine Nachricht zu ignorieren oder zu verarbeiten (separation of concerns):

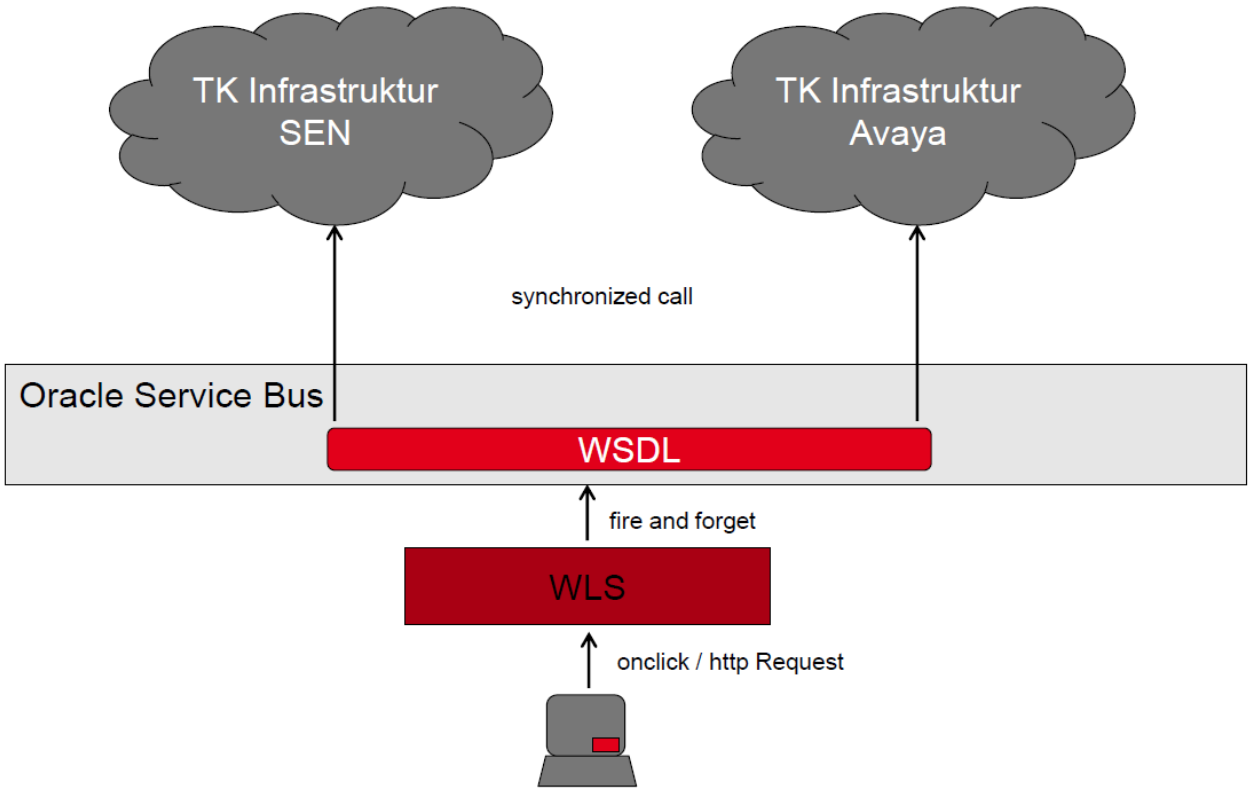


Abb. 4: Use Case 2 - ausgehender Anruf

Zu diesem Zweck sendet eine Anwendung (Server-Seite) einen SOAP Request an den OSB. Im Business Service wird dann die Nachricht zu allen angeschlossenen Telefonie-Providern weitergeleitet. Daraufhin wird die Telefonverbindung auf dem Endgerät des Anwenders (Telefon, Headset) hergestellt. Es wird kein Return-Code ausgewertet (Fire-and-forget), da dies die Middleware- und Telefoniesysteme wiederum koppeln würde und eine Rückmeldung ja bereits über das Endgerät erfolgt.

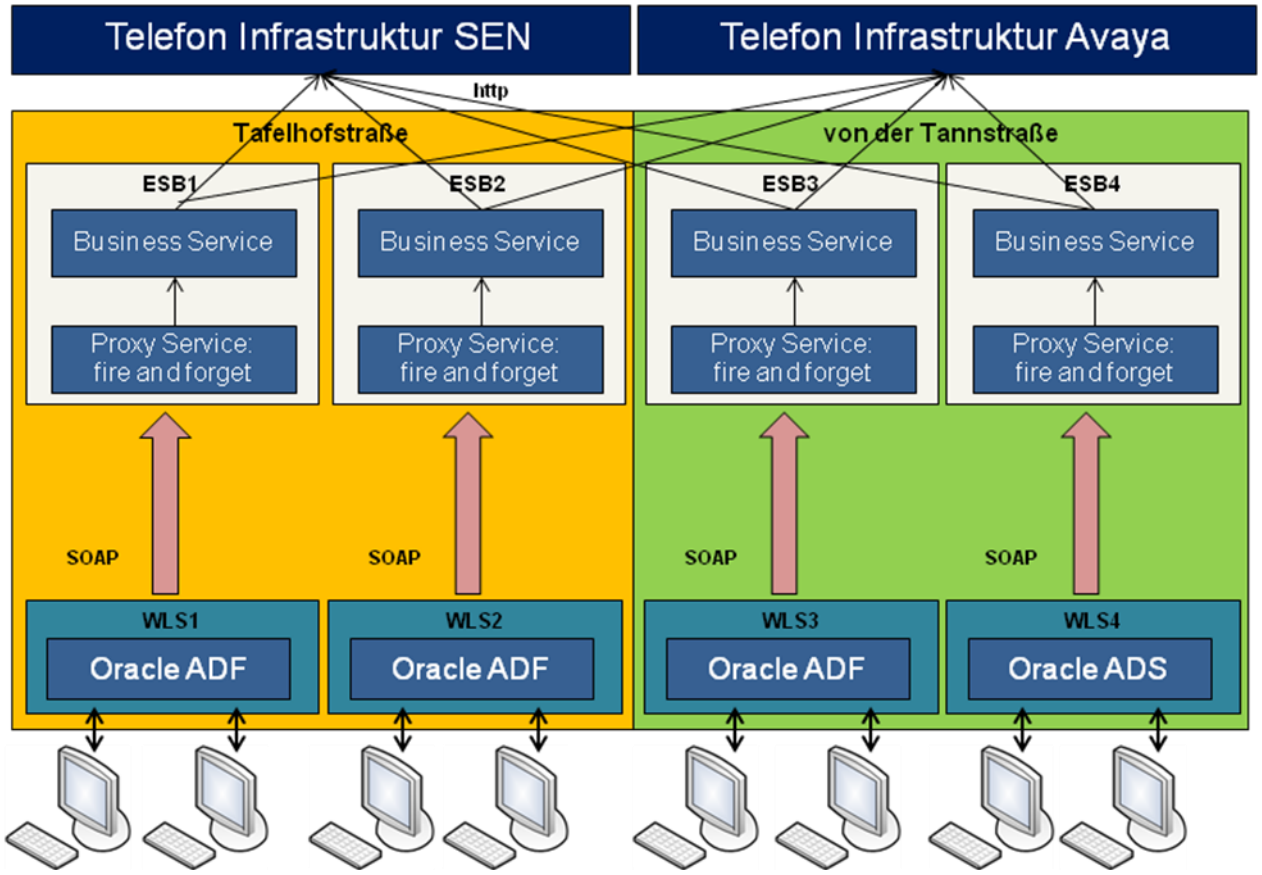


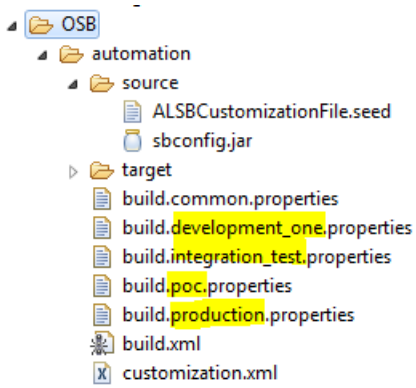
Abb. 6 Systemarchitektur ausgehender Anruf

Staging

Wie immer muss auch diese Lösung einen definierten Staging Prozess von der Entwicklung über Test und Integration bis hin zur Produktion durchlaufen:

Zu diesem Zweck muss eine bequeme und einfache Möglichkeit existieren notwendige Änderungen an Software Artefakte durchzuführen. Insbesondere sind hier die URLs und Anzahl der Endpunkte für die private Proxy Services und Adressen der JMS Queues, sowie die „OSB Service Accounts“ für die Basic Authentication zu beachten.
Für die ersten (Endpunkte) bietet der OSB die s.g. **customization.xml**, die er für jedes seiner internen „Projekte“ generieren kann.
Im Falle der Service Accounts für die Basic Authentification ist das leider nicht möglich. Es ist notwendig eine Deployment JAR eines OSB Projekts zu entpacken und dann die entsprechenden Benutzerdaten manuell einzutragen. Eine sehr gute Alternative dazu, die auch im Betrieb automatisiert verwendet werden kann, bietet die Nutzung von Ant an. Alle für die Customization notwendigen Properties und deren Werte werden in umgebungsabhängige Property Files ausgelagert und beim

jeweiligen Deployment von Ant verarbeitet:



Ausblick

Die hier implementierte Lösung ist grundsätzlich für jede Art von Benutzerbenachrichtigung nutzbar. So kann über die hier vorgestellte Lösung auch ein Benutzer der Applikation X einem Benutzer eine Nachricht senden, bei der dieser je nach Topic eine bestimmte Aktion in der empfangenden Applikation auslösen kann.

Die Lösung zeigt zum einen die Mächtigkeit des Oracle Service Busses wenn es darum geht, über verschiedene Message-Pattern heterogene Welten entkoppelt voneinander zu integrieren. Zur Realisierung der Lösung war dank des OSBs nur ein Minimum an Java-Code zu implementieren, die eigentliche Implementierung der Gesamtlösung war mit wenig Aufwand verbunden. Die weitgehende Entkopplung der Systeme voneinander vereinfachte auch unabhängige Tests der verschiedenen Beteiligten.

Darüber hinaus demonstriert die vorgestellte Lösung aber auch die Vorteile einer asynchronen Kommunikation zum Benutzer einer Web-Anwendung mittels Oracle ADS.

Wir sind überzeugt, dass gerade in einem Prozesskontext in Verbindung mit z.B. der Oracle BPMN Suite noch weitere Anwendungsmöglichkeiten für asynchrone GUI-Services schlummern.

Kontaktadressen:

Lyubomir Yordanov
Yordanov Consulting
Bergstr. 11
D-90403 Nürnberg
Telefon: +49 (0) 0911-179 2379
E-Mail lyubomir@yordanov.eu

Ralf Ernst
IT-Systemhaus der BA
Regensburgerstr. 104
D-90478 Nürnberg

Telefon: +49 (0) 0911-179 2379
Fax: +49 (0) 0911-179 904765
E-Mail ralf.ernst@arbeitsagentur.de
Internet: www.arbeitsagentur.de