eIntegrationsportal

Konzeptdokument

KO - Konzeption / Spezifikation
Konzeptdokument

Architekturkonzept

Architekturkonzept eIP 1.5.0

DOKUMENTINFORMATION

Projektbezeichnung	Pflege und Weiterentwicklung eJustice-Basisdienste (eKP und eIP)
Projektleiter	Jörg Brendler
Verantwortlich	Alexander Schulze
Stand	1.0 vom 24.06.2019
Status	Fertig gestellt
Dokumentablage	EIP.KO.Konzept_Architekturkonzept_eIP_1.5.0_v1.0.docx
V-Modell XT Version	1.4

ÄNDERUNGSVERZEICHNIS

Änderung			Geänderte Kapitel	Beschreibung der Änderung	Autor	Zustand
Nr.	Datum	Version				
1	06.10.2015	0.1	Alle	Initiale Erstellung	Alexander Schulze	In Bearbeitung
2	02.12.2015	1.0	Alle	Finalisierung nach Review	Alexander Schulze	Fertig gestellt
3	15.12.2015	1.0.1	4, 6.2, 6.3, 7.3, 7.4, 10	Anpassung Formulierungen nach Review durch Auftraggeber	Alexander Schulze	Fertig gestellt
4	28.11.2016	1.1	Alle	Anpassung und Erweiterung der Architekturentscheidungen, Ergänzung der Architektur um Sicherheitsarchitektur und Offline-Fähigkeit Aktualisierung auf eIP 1.2	Alexander Schulze	Fertig gestellt
5	23.01.2017	1.1.1	6.4.7, 6.4.11	Formulierung aufgrund Kommentare von Herrn Wolff angepasst	Alexander Schulze	Fertig gestellt
6	01.08.2018	1.3.0	Alle	Aktualisierung auf eIP 1.3 Neues Cachingkonzept Mandantenfähigkeit	Alexander Schulze	Fertig gestellt
7	17.01.2019	1.4.1	Alle	Formale Überarbeitung	Lars de Vries	In Bearbeitung
8	26.02.2019	1.4.1	1, 5, 10	Ergänzung Architekturentscheidungen und Aktualisierung Datenmodell für 1.4.1	Alexander Schulze	Fertig gestellt
9	24.06.2019	1.5.0	7, 8, 10	Ergänzungen zum Signaturprüfdienst und Aktualisierung Datenmodell für 1.5.0	Alexander Schulze	Fertig gestellt

PRÜFVERZEICHNIS

Die folgende Tabelle zeigt einen Überblick über alle Prüfungen – sowohl Prüfungen durch die eigenständige Qualitätssicherung als auch formale Prüfungen – des vorliegenden Dokumentes.

Datum	Geprüfte Version	Anmerkungen	Prüfer	Neuer Produktzustand
21.05.2014		Review des eIP Sicherheitskonzepts	Herr Mahr (It-Stelle)	In Bearbeitung
24.11.2015	0,1	Kommentare und Ergänzungen in den Kapiteln	Thomas Hofmann	In Bearbeitung
01.12.2015	0,2	Kommentare und Ergänzungen in den Kapiteln	Ronny Kröhne	In Bearbeitung
08.12.2015	1.0	Einige Kommentare zu Formulierungen und Ergänzungen	Alexander Wolff (IT-Stelle)	Fertig gestellt
16.03.2017	1.1.1	QS; Version 2.0 erstellt	S. Voth	Fertig gestellt

INHALTSVERZEICHNIS

1 EINLEITUNG

Das eIntegrationsportal (eIP) ist eine desktopseitige Integrationslösung, welche verschiedene Anwendungen in einer integrierten Oberfläche darstellen kann und diese mit einer elektronischen Akte zusammenbringt.

Dieses Architekturkonzept beschreibt die Gesamtarchitektur von eIP, welche als Grundlage für die Umsetzung herangezogen wird. Dabei werden die grundlegenden Anforderungen aus technischer Sicht, die Prämissen und Begrifflichkeiten erklärt. Weiterhin werden die Architekturentscheidungen dokumentiert und die resultierende Gesamtarchitektur, Kommunikationsbeziehungen, Datenmodell und Architektursichten dargestellt. Die Darstellung der Auswirkungen auf den Betrieb, einige Programmiervorgaben sowie ein Ausblick auf weitere Themen runden das Gesamtbild ab.

Das Architekturkonzept bezieht sich auf die eIP Version 1.4.1 auf Basis von Eclipse RCP 4.3.1 mit dem Stand vom Februar 2019.

Für eIP existiert ein Integrationsleitfaden, der eine Einführung in eIP beinhaltet, die fachlichen Anforderungen an eIP beschreibt und die fachliche und technische Integration eines Fachverfahrens grob darstellt. In diesem Integrationsleitfaden sind auch Beispielabläufe mit einem integrierten Fachverfahren (forumSTAR) und einer angebundenen Kommunikationsplattform (eKP) beschrieben. Nachdem diese Abläufe nicht die grundlegende Architektur von eIP beeinflussen, sind diese nicht in diesem Dokument dargestellt.

2 BEGRIFFLICHKEITEN

2.1 Vorgangsbearbeitung

Unter einer Vorgangsbearbeitung wird ein System verstanden, welches Aufgaben und Vorgänge verwalten kann, welche persönlich oder Gruppen zur Bearbeitung zugeordnet werden können. Dabei müssen Vertreterregelungen ebenso berücksichtigt werden wie unterschiedliche Prioritäten von Aufgaben. In der Vorgangsbearbeitung können auch Termine, Wiedervorlagen, elektronische Nachrichteneingänge reflektiert werden.

2.2 Aktenumgang

Unter Aktenumgang werden sämtliche fachlichen Funktionen verstanden, die im Umfeld einer elektronischen Akte benötigt werden. Dies sind u.a. das Anlegen einer eAkte, Hinzufügen/Verschieben/Löschen von Dokumenten und Ordnern, Verwalten von Querbezügen und Anmerkungen zu Dokumenten. Außerdem sind fachliche Grundfunktionen wie z.B. Paginierung, Akten verbinden und trennen von essentieller Bedeutung für den Aktenumgang.

2.3 Document Management System

Ein Document Management System (DMS) ist für die Speicherung und Verwaltung von Dokumenten sowie die Volltextsuche verantwortlich. Dabei sind wichtige Themen wie Versionierung, revisionssichere Speicherung entweder im Produkt oder in nachgelagerten Komponenten zu beachten.

Ein DMS ist nicht zwingend ein eAkte-System oder eine Vorgangsbearbeitung oder ein DOMEA[1]-Produkt.

2.4 Elektronische Kommunikationsplattform

Die eKP stellt eine Kommunikationsplattform dar, die auf einer JEE Middleware aufsetzt. Die eKP besteht dabei aus wiederverwendbaren Services und Prozessen.

[1] Dokumentenmanagement und elektronische Archivierung im IT-gestützten Geschäftsgang, siehe auch https://de.wikipedia.org/wiki/DOMEA-Konzept

Der primäre Zweck der eKP ist, die Kommunikation zwischen unterschiedlichen Parteien (z.B. Fachverfahren) einheitlich zu regeln. Dazu bietet die eKP Mehrwertdienste an, so dass diese nicht von den jeweiligen Kommunikationspartnern realisiert werden müssen. Derzeit ist die zentrale Kommunikationsart dabei die Kommunikation über die Virtuelle Poststelle (VPS).

Auf Basis der eKP können spezifische Services für einzelne Anwendungsfälle angeboten werden sowie neue Prozesse und Kommunikationsszenarien umgesetzt und betrieben werden.

Die Anbindung an die eKP ist für eIP nicht zwingend erforderlich.

2.5 Begriffe aus dem Bereich Sicherheit

2.5.1 Authentisierung

Unter Authentisierung wird verstanden, wer ein Benutzer ist – d.h. es wird seine Identität durch ein Anmeldeverfahren (z.B. Benutzerkennung – Passwort) festgestellt. Es wird somit die Frage nach dem „Wer" beantwortet.

„Authentifizierung (griechisch αυθεντικός authentikós ‚echt', ‚Anführer'; Stammform verbunden mit lateinisch facere ‚machen') ist der Nachweis (Verifizierung) einer behaupteten Eigenschaft einer Entität, die beispielsweise ein Mensch, ein Gerät, ein Dokument oder eine Information sein kann, und die dabei durch ihren Beitrag ihre Authentisierung durchführt."[2]

2.5.2 Autorisierung

Unter Autorisierung wird verstanden, was ein Benutzer in einem System tun darf – d.h. es wird ausgesagt, welche Aktionen der Benutzer ausführen darf und welche Daten er sehen darf. Es wird somit die Frage nach dem „Was" beantwortet.

„Autorisierung ist im weitesten Sinne eine Zustimmung, spezieller die Einräumung von Rechten gegenüber Interessenten, ggf. zur Nutzung gegenüber Dritten. Die Autorisierung überwindet Mechanismen von Sicherungen gegen Unbefugte. Eine Autorisierung hebt keinen Schutz auf. Eine Autorisierung gilt gegebenenfalls eingeschränkt nur in einem Kontext und/oder Modus. Die Autorisierung erfolgt sinnvollerweise nicht ohne eine vorherige erfolgreiche Authentifizierung."[3]

[2] http://de.wikipedia.org/wiki/Authentisierung

[3] http://de.wikipedia.org/wiki/Autorisierung

2.6 Darstellung der grundlegenden Komponenten eines eAkte Systems

Im Jahr 2013 wurden die grundsätzlichen Komponenten, deren fachliche Zuordnungen sowie die notwendigen Schnittstellen im Rahmen von verschiedenen Workshops mit dem Auftraggeber, HP und dataport erarbeitet. Das folgende grundlegende Komponentendiagramm ist dabei das zusammengefasste Ergebnis dieser Workshopreihe, welches als Grundlage für das eIP-Architekturkonzept weiterhin Bestand hat.

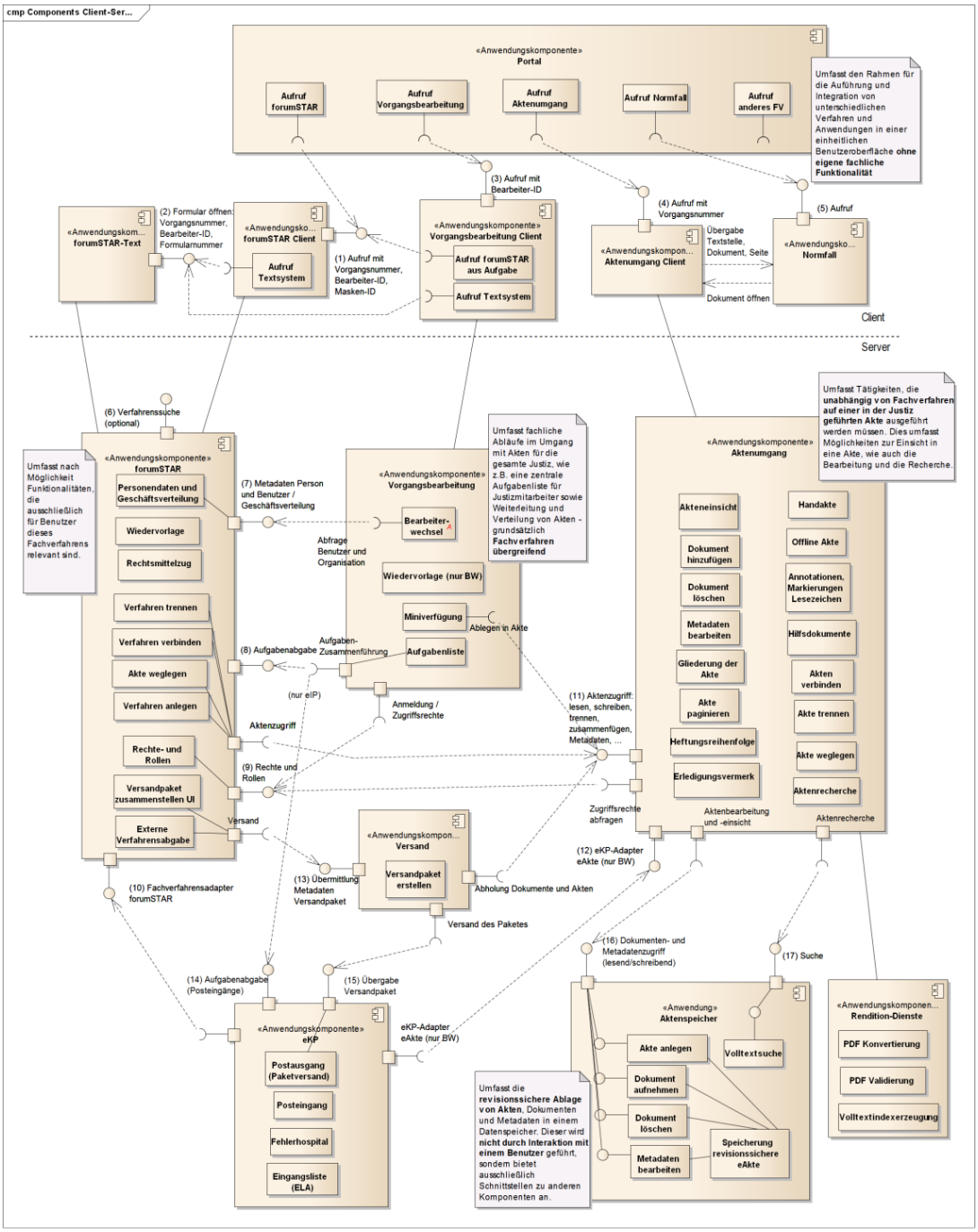

Abbildung 2-1: Darstellung der grundlegenden Komponenten eines eAkte Systems im Zusammenspiel

3 PRÄMISSEN FÜR EIP

In diesem Kapitel werden die Prämissen für eIP festgeschrieben, die vom Auftraggeber für die Konzeption und Umsetzung festgelegt und beachtet wurden.

3.1 Fachverfahrensunabhängige Implementierung

eIP wird unabhängig von Fachverfahren umgesetzt. Dies bedeutet, dass eIP kein Wissen über die integrierten Fachverfahren hat und somit auch keine spezifischen Umsetzungen für dedizierte Fachverfahren in Basiskomponenten beinhalten kann.

Eine Ausnahme stellen Konfigurationswerte dar, die zentral gespeichert werden und auch z.B. den Zugriff auf das Fachverfahren regeln können.

3.2 Wiederverwendbarkeit und Unabhängigkeit von Operationen

Eine Serviceoperation muss wiederverwendbar sein. Dies bedeutet, sie hat klar definierte Ein- und Ausgabeparameter und darf nicht auf Informationen aus anderen Services oder Quellen vertrauen bzw. zwingend darauf Bezug nehmen. Die Eingabeparameter beschränken sich auf die Parameter, die diese Serviceoperation zur Verarbeitung benötigt.

3.3 Produktunabhängige Konzeption und Umsetzung

Das Architekturkonzept wird produktunabhängig entworfen. Ziel ist es hierbei, geltende Standards auszunutzen und basierend auf diesen, das Gesamtkonzept produktunabhängig umzusetzen.

Die folgenden anerkannten Spezifikationen und Software sind dabei relevant:

- JEE ab Version 5 inkl. der abhängigen Spezifikationen (z.B. JAX-WS)
- Eclipse Rich Client Platform (ab Version 4.3)

Proprietäre Erweiterungen eines Middleware Herstellers werden nicht bevorzugt behandelt, außer es gibt diese Erweiterungen bei unterschiedlichen Herstellern in ggf. verschiedenen Ausprägungen (Beispiel: Maximale Transaktionszeit eines EJB ist laut JEE Spezifikation nicht pro EJB spezifizierbar, aber alle großen JEE Middleware Hersteller bieten dieses Feature an).

3.4 Investitionsschutz

Bereits angeschaffte Infrastruktur, Middleware-Komponenten (Lizenzen) sowie umgesetzte Funktionalitäten in Fachverfahren sollen auch mit eIP weiter genutzt und wiederverwendet werden.

Dies bedeutet, dass nicht alle Funktionen in eIP vollständig neu umgesetzt werden sollen.

Im Rahmen des Auftraggebers sind dies konkret die folgenden Anforderungen:

- Oracle FusionMiddleware 11 bzw. 12
- Oracle Datenbank 11 bzw. 12
- forumSTAR elektronischer Versand von Dokumenten (eVvD)

  Hinweis: eIP ist fachverfahrensunabhängig umgesetzt. Dennoch wurde die Konzeption und Umsetzung mit dem Fokus der Integration von forumSTAR begonnen. Demnach sollten nicht alle Services, die in forumSTAR bereits vorliegen, erneut in eIP umgesetzt werden, auch wenn diese einen querschnittlichen Charakter für mehrere Fachverfahren darstellen würden. Dies bedeutet nicht, dass diese Services nicht irgendwann Bestandteil von eIP werden können. Dies ist gut am Beispiel der Anbindung der Signaturanwendungskomponente zu erkennen. Zu Beginn nutzte eIP diese aus forumSTAR, seit Version 1.0.8 bietet eIP dies als Querschnittsfunktion an.

Die Anbindung an den Dokumentenspeicher kann nicht von einem einheitlichen DMS in allen Verbundländern ausgehen, da ggf. bereits heute Länderentscheidungen für ein DMS existieren. Demzufolge muss eIP in der Lage sein, unterschiedliche DMS Produkte anzubinden.

3.5 Länderunabhängige Konzeption und Umsetzung

Die Konzeption und Umsetzung ist unabhängig von den Bundesländern des Entwicklungsverbundes. Es kann zwar sein, dass bestimmte Basis-Services und Komponenten nur in einigen Verbundländern im Einsatz sind. Innerhalb der Umsetzung von eIP sind keine länderspezifischen Umsetzungen erlaubt. Es ist allerdings durchaus möglich, beim Deployment die Verknüpfung der Komponenten bei identischen Schnittstellen zu verändern oder unterschiedliche Plugins im Client zu aktivieren.

Es existieren keine Länderschalter (vergleichbar zu den forumSTAR Länderschaltern), die zur Laufzeit unterschiedliche Reaktionen innerhalb der Anwendung auslösen.

Eine Lieferung von eIP beinhaltet immer alle Komponenten aller Länder. Es gibt keine länderspezifische Lieferung von eIP. Dies betrifft auch die Integrationskomponenten für unterschiedliche Fachverfahren. Es kann spezifische Clientausprägungen in Bezug auf die paketierten Plugins (z.B. eIP ohne eAkte) geben, die nach Abstimmung mit IBM zusätzlich geliefert werden.

3.6 Prämissen der Sicherheitsarchitektur

Mit der Justiz wurden folgende Prämissen für die Sicherheitsarchitektur abgestimmt:

- Prämisse 1: Umsetzung des serviceorientierten Gedankens (lose Kopplung, Wiederverwendung von bestehenden Komponenten, modularer Aufbau) auch für die Sicherheitsarchitektur
- Prämisse 2: Keine doppelte Datenhaltung von Benutzern, Rollen oder Rechten und somit auch keine Synchronisations-/Abgleichmechanismen
- Prämisse 3: Keine Abbildung von erweiterten Rechten für die elektronische Akte innerhalb von eIP

  Dies bedeutet konkret, dass derzeit keine erweiterten Rechte (z.B. Akte nur lesen) angeboten werden, die über die Rechte eines Benutzers an einem Verfahren hinausgehen.
- Prämisse 4: Vergabe von Rechten in der Hoheit der Fachverfahren
- Prämisse 5: Grundsätzliche Anbindung an eine SAFE[4]-Domäne ermöglichen

[4] Secure Access to Federated e-Justice/e-Government,
siehe auch http://www.justiz.de/elektronischer_rechtsverkehr/grob-und-feinkonzept

4 BASISANFORDERUNGEN AN EIP

Im Rahmen des Architekturkonzepts werden die Basisanforderungen nur rudimentär beschrieben, da diese in der fachlichen Dokumentation detailliert erfasst sind. Prämissen aus dem vorherigen Kapitel werden in den Basisanforderungen nicht erfasst.

- Philosophie des eJustice-Arbeitsplatzes der Zukunft mit Start über eIP und integrierten Anwendungsfenstern

  Es soll ein Integrationsportal für den Desktop geschaffen werden, welches in der Lage ist, verschiedene Anwendungen mit unterschiedlichen Technologien in einer integrierten Oberfläche darzustellen.

- Ergonomische, intuitive Bedienbarkeit des Integrationsportals und der elektronischen Akte

  Eine ergonomische, intuitive Bedienbarkeit des Gesamtsystems sowie eine ansprechende Oberfläche mit einfachen Abläufen sind für die Akzeptanz von enormer Bedeutung. Vor allem im Umfeld der Entscheider wird ein neues System nur angenommen, wenn es modernsten Ansprüchen genügt.

- Fensterverwaltung mit Zuordnung der Fenster zu einem Arbeitskontext

  Das Integrationsportal soll die Fenster verwalten und die Fensterpositionierung vornehmen. Darüber hinaus müssen die Fenster minimiert, maximiert sowie in einen Vollbildmodus geschaltet werden können. Jedes Fenster soll dabei einem Arbeitskontext zugeordnet werden können, wenn es im Kontext eines (Gerichts-)Verfahrens geöffnet wird. Dadurch übernimmt das Portal den Kontextbezug der Fenster, welche in der Oberfläche dargestellt werden soll und der zu Komfortfunktionen (z.B. Schließen aller Fenster eines Kontextes) führen soll. Auf einem Monitor sollen maximal drei Fensterbereiche sichtbar sein.

- Mehrmonitorbetrieb

  Es soll möglich sein, mehrere (max. 3) Monitore anzuschließen und die Fenster über das Integrationsportal einfach zwischen den Monitoren zu verschieben. Auf jedem Monitor soll eine Fenstersteuerungs-/verwaltungskomponente genutzt werden.

- Umsetzung einer Aufgabenverwaltung zur aggregierten Anzeige von persönlichen und einheitenbezogenen Aufgaben

Im Rahmen des Integrationsportals soll eine Komponente zur Aufgabenverwaltung umgesetzt werden, welche bei Start des Integrationsportals die aktuelle Aufgabenliste für einen Benutzer sowie für Gruppen anzeigt. Diese Aufgaben können in Bearbeitung genommen, mit einer integrierten Anwendung (z.B. Fachverfahren, eAkte) geöffnet werden und nach der Bearbeitung erledigt oder weitergeleitet (abhängig vom Aufgabentyp). Zusätzlich sind Zugriffe auf die Aufgaben einer anderen Person im Vertretungsfall zu realisieren.

- Umsetzung einer elektronischen Akte

  Im Rahmen des Integrationsportals soll eine Komponente realisiert werden, welche eine elektronische Akte abbildet. Diese Komponente muss eine Aktenstruktur verwalten, Dokumente anzeigen und Dokumente bearbeiten können (Anmerkungen o.ä.). Die Speicherung der elektronischen Akte findet in einem Dokumentenspeicher statt. Neben der Oberfläche sind auch serverseitige Schnittstellen vorzusehen, da Veränderungen an der Akte auch durch serverseitige Aktionen im Fachverfahren bzw. einer integrierten Anwendung initiiert werden können. Es soll die Möglichkeit bestehen, die gleiche sowie unterschiedliche elektronische Akten parallel öffnen zu können, um z.B. das vergleichende Lesen zu ermöglichen. Dafür ist die Fensterverwaltung in eIP mit Arbeitskontext Grundvoraussetzung.

- Offline-Modus für das Lesen der elektronischen Akte auch ohne Serververbindung (Ausfall o.ä.)

  eIP soll einen Offline-Modus bieten, um bei Ausfall einer serverseitigen Systemkomponente weiterhin den Zugriff auf lokal zwischengespeicherte Akten zu erlauben. Im Offline-Modus können keine Veränderungen an der Akte vorgenommen werden, allerdings kann die Akte unter Einsatz des eIP-Aktenviewers vollständig gelesen werden. Um dies zu ermöglichen, müssen verschiedene Sicherheitsinformationen lokal zwischengespeichert werden. Dadurch ist der Zugriff auf berechtigte Akten sichergestellt, auch wenn bspw. das Fachverfahren nicht erreichbar ist. Weiterhin werden Aktenstrukturen und Dokumente in einem Cache gehalten, der auch beim Start von eIP aktualisiert werden kann.

- Unterstützung unterschiedlicher Datensenken (Produktunabhängigkeit) für die Speicherung der elektronischen Akte mit einer beispielhaften Umsetzung auf Basis von CMIS und Anbindung von Alfresco

  In den Prämissen wurde verdeutlicht, dass eine einheitliche Datensenke in einem Entwicklungsverbund für Dokumente nicht anzutreffen sein wird. Deshalb muss die Datensenke für die elektronische Akte flexibel austauschbar sein und unterschiedliche Produkte unterstützen. Im Rahmen der Prototypen- und Pilotierungsumsetzung soll der Zugriff per CMIS (Content Management Interoperability Standard) auf Alfresco als Open Source DMS umgesetzt werden.

- Transaktionsfähige Schnittstellen für eine Vollständigkeitsgarantie

  Bei einer elektronischen Akte müssen Mechanismen vorgesehen werden, um fachliche Transaktionen auch technisch abzubilden und zu unterstützen, um eine Vollständigkeitsgarantie für Aktenveränderungen anbieten zu können.

- Anbindung eines Strukturierungswerkzeugs (Normfall Manager) für die Durchdringung

  Vor allem große Akten können mit Hilfe von Strukturierungswerkzeugen durchdrungen werden. In das Integrationsportal soll der Normfall Manager als eine konkrete Produktausprägung eines Strukturierungswerkzeugs integriert werden und mit der elektronischen Akte interagieren können.

- Integration des Fachverfahrens forumSTAR und forumSTAR-Text

  Als wichtige Fachverfahren mit mehreren tausend Benutzern sollen forumSTAR und forumSTAR-Text vollumfänglich integriert werden. Alle neu geöffneten Fenster der Anwendungen sollen als Fenster innerhalb des Integrationsportals erscheinen. Für jedes Verfahren ist ein eigenes Fenster innerhalb des Portals vorzusehen.

- Integration der Microsoft Office Produkte und direkte Speicherung in der elektronischen Akte

  Die Microsoft Office Produkte (Word, Excel, PowerPoint) sollen in das Integrationsportal integriert werden und es soll eine Speicherung der Dokumente im nativen Format des jeweiligen Produkts in der Handakte möglich sein.

- Anbindung an die eKP zur performanten, frühzeitigen Übernahme von elektronischen Nachrichten

  Die eKP informiert Fachverfahren im Rahmen des Empfangsprozesses über den Nachrichteneingang. Das Fachverfahren entscheidet dann, entweder vollautomatisch oder durch manuelle Aktivitäten eines Sachbearbeiters, über die Zuordnung einer Nachricht zu einem konkreten Verfahren. Um die Speicherung der u.U. großen Anhänge in der elektronischen Akte performant zu ermöglichen, ist eine frühzeitige Übernahme der Inhalte noch während des Empfangsprozesses der eKP sinnvoll.

- Keine Datenreplikation jeglicher Art zur Unterbindung von notwendigen aufwändigen Synchronisationsmechanismen

  eIP soll als Integrationsportal für verschiedene Anwendungen genutzt werden und dabei auf die bereits bestehenden Fachverfahren und Infrastrukturkomponenten aufbauen. Eine vollständige Unabhängigkeit von allen anderen Anwendungen ist somit nicht das Ziel. Dadurch sollen auch die verschiedenen Daten an den Stellen verbleiben, an denen sie originär gehalten, verwaltet und verändert werden. Jeder Replikationsprozess

um eine Konsistenz über die verschiedenen Systeme zu gewährleisten ist kosten- und ressourcenintensiv. Dies soll im Rahmen des Integrationsportals auf alle Fälle vermieden werden. Es wird in diesen Fällen von einer Online-Verfügbarkeit ausgegangen.

- Dokumentenkonvertierung von nicht-PDF-Dokumenten

  In der elektronischen Gerichtsakte, d.h. in der Verfahrensakte mit paginierten und unveränderlichen Dokumenten, können nur PDF-Dokumente existieren, damit eine Unveränderlichkeit und eine seitenstabile Anzeige umgesetzt werden kann. Somit muss eine Dokumentenkonvertierung von Fremdformaten angeboten werden, die sowohl Eigenproduktionen in Fremdformaten (z.B. MS Word) als auch elektronische Eingänge bearbeiten kann. Dabei soll der Benutzer von der Konvertierung möglichst wenig beeinflusst werden.

- Unterstützung des gesamten Arbeitsablaufs für elektronische Nachrichteneingänge und Posteingangsbearbeitung

  Es muss im Integrationsportal möglich sein, den gesamten Arbeitsablauf für elektronische Nachrichteneingänge und Posteingangsbearbeitung medienbruchfrei zu unterstützen, so dass Servicekräfte und Entscheider mit einem Werkzeug den gesamten Ablauf bestreiten können. Dabei müssen die bereits umgesetzten Mechanismen der Fachverfahren weiterhin unterstützt und genutzt werden können.

- Modulare Grundarchitektur am Client und am Server

  Ein Integrationsportal soll unterschiedliche Anwendungen integrieren und bedienen können. Damit ist der Einsatzzweck nicht auf eine Konstellation von Anwendungen beschränkt, sondern kann sich je nach Bundesland und Bereich vielfältig unterscheiden. Um diesem Umstand gerecht zu werden, ist eine modulare Grundarchitektur sowohl für den Client als auch für den Server unerlässlich. Ohne eine entsprechende Architektur und klaren Verantwortlichkeiten ist der flexible Einsatz nicht möglich.

- Clusterfähige Lösung für Ausfallsicherheit und Performance

  Die gesamte Lösung muss moderne Konzepte für Ausfallsicherheit und Performance auf Basis von geclustertern Infrastruktur und Middleware-Komponenten bestmöglich unterstützen.

5 ARCHITEKTURENTSCHEIDUNGEN

Die Architekturentscheidungen in diesem Kapitel stellen die klassischen Weichenstellungen im Projekt für die technische Architektur dar. Jede einzelne stellt eine klare Ausrichtung in einer der dargestellten Alternativen dar. Ein späterer Wechsel ist oftmals nicht mehr problemlos möglich. Entsprechende Möglichkeiten sind in den Architekturentscheidungen dokumentiert.

Architekturentscheidungen sind wichtige Rahmenbedingungen für ein Projekt. Nicht jede Entscheidung kann zu einer Einzelfallentscheidung deklariert werden, da damit für jeden Einzelfall spezifisch erneut diskutiert und entschieden werden muss. Es muss in einer großen Architektur somit ein vorgegebener Pfad existieren, der nur in Ausnahmefällen bzw. bei speziellen Bedingungen verlassen werden darf. Sonst sind die Einzelabstimmungen und zeitlichen Abläufe bei der Konzeption und Umsetzung viel zu aufwändig und führen mittelfristig zu auseinandergleitenden Lösungen.

5.1 Architekturentscheidung Clienttechnologie

Themenbereich	**Clienttechnologie**	
Architekturentscheidung	ID	1.1
Issue oder Problem	Im Rahmen des Integrationsportals muss die Frage nach der Client-Technologie gestellt und dokumentiert werden. Dabei müssen die Wartbarkeit, Betreibbarkeit, Zukunftssicherheit und viele weitere Aspekte betrachtet werden, um keine falsche Entscheidung zu treffen. Der wichtigste Aspekt stellt aus Sicht der Basisarchitektur für das Integrationsportal die Integrationsfähigkeit von unterschiedlichen Anwendungen mit unterschiedlichen Technologien und Interaktionsmöglichkeit mit der elektronischen Akte dar.	
Annahmen		
Motivation	Ergonomie, Zukunftssicherheit, Wartbarkeit muss sichergestellt sein	
Alternativen	1. Thin-Client 2. Rich-Client	
Entscheidung	Alternative 2 wurde gewählt	
Erläuterung der Entscheidung	Ein Thin-Client hat unbestreitbare Vorteile in einem heterogenen Umfeld, bei dem die Zielplattform (Betriebssystem, Laufzeitumgebung usw.) nicht bekannt oder bestimmbar ist. Auch die unmittelbare Möglichkeit der mobilen Nutzung sind klare Vorteile. Aus betrieblicher Sicht sind auch die Thin-Clients im Vorteil, da kein Rollout auf Endgeräte notwendig ist. Dies erzwingt allerdings, dass ein kompatibler Browser ausgerollt vorliegt. Neuere Versionen der Browser können oftmals auch zu fehlerhaften Darstellungen führen. Dem gegenüber stehen bei Rich-Clients zu erwartende geringere Entwicklungs- und Wartungskosten, um ein UI in der gleichen Güte und Funktionsumfang zu erreichen	

Themenbereich	Clienttechnologie
	sowie eine wesentlich höhere Benutzerakzeptanz durch erhöhte Performance und Interaktionsmöglichkeiten vor allem mit nativen Anwendungen. Die Anforderungen an die Integrationsfähigkeiten und clientseitigen Kommunikationsmöglichkeiten sind mit einem Thin Client nicht bzw. nur sehr schwer umsetzbar. Ein Offline-Modus bei Ausfall der Serverbestandteile sind mit einer Thin Client Architektur ebenfalls nur erschwert umsetzbar (Einsatz von AngularJS und Browser Cache als denkbare Alternative). Aus Sicht der IBM ist der Einsatz einer Rich-Client Technologie der einzig gangbare Weg für die Umsetzung eines Integrationsportals mit den definierten Prämissen und Basisanforderungen.
Auswirkungen	Nachteile des Rich-Clients lassen sich auch durch geänderte Betriebsmodelle ausgleichen/verringern (z.B. Rollout durch automatische Updatemechanismen im Sinne eines Self-Managed Clients).
Abgeleitete Anforderungen	Ein späterer Wechsel kommt einer Neuentwicklung der Benutzeroberfläche gleich.
Verbundene Entscheidungen	Architekturentscheidung 1.2 - Architekturentscheidung Rich Client Platform

5.2 Architekturentscheidung Rich Client Platform

Themenbereich	Clienttechnologie	
Architekturentscheidung	ID	1.2
Issue oder Problem	Im Rahmen des Integrationsportals muss die Frage nach der Rich-Client-Platform und des verwendeten Toolkits gestellt und dokumentiert werden. Dabei müssen die Wartbarkeit, Betreibbarkeit, Zukunftssicherheit und viele weitere Aspekte betrachtet werden, um keine falsche Entscheidung zu treffen. Letztendlich spielen auch die Integrationsfähigkeit und Interaktionsmöglichkeit mit weiteren Anwendungen eine entscheidende Rolle.	
Annahmen		
Motivation	Ergonomie, Zukunftssicherheit, Wartbarkeit muss sichergestellt sein	
Alternativen	1. .NET Framework unter Einsatz von nativen Windows APIs 2. Adobe Flash / Shockwave / Flex / Air 3. Java SE 4. Spring Framework 5. Eclipse RAP 6. Netbeans 7. Eclipse RCP a) Java Swing & AWT b) Java FX c) SWT	
Entscheidung	Alternative 7) c) wurde gewählt	

Themenbereich	Clienttechnologie
Erläuterung der Entscheidung	Die Varianten haben gezeigt, dass native Umsetzungen (1 und 2) aufgrund der Prämissen in Richtung Java nicht sinnvoll oder möglich sind. Java SE, Spring und RAP bieten verschiedene Nachteile und sind de facto keine Rich Client Platform. Netbeans und Eclipse RCP sind somit die einzig sinnvollen Varianten, wobei Netbeans auf Java Swing basiert und nicht mehr weiterentwickelt und mittelfristig durch Java FX abgelöst wird. Eine wesentliche Anforderung stellt die Integration von nativen Anwendungen dar (z.B. per OLE, Nutzung eingebetteter Browser wie Internet Explorer). Diese ist mit JavaFX als leichtgewichtiges Framework nicht umsetzbar. Mit SWT existieren fertige Komponenten für die OLE-Integration und die Einbettung von Webbrowsern. Es kommt somit die Eclipse Rich Client Platform (RCP) ab Version 4.3 zum Einsatz. Die Benutzeroberfläche wird mit dem Standard Widget Toolkit (SWT) erzeugt. Punktuell kann der Einsatz von Java FX erfolgen.
Auswirkungen	-
Abgeleitete Anforderungen	Ein späterer Wechsel kommt einer Neuentwicklung der Benutzeroberfläche gleich.
Verbundene Entscheidungen	Architekturentscheidung 1.1 - Architekturentscheidung Clienttechnologie

5.3 Architekturentscheidung synchrone und asynchrone Services

Themenbereich	Prozesssteuerung	
Architekturentscheidung	ID	2.1
Issue oder Problem	Viele Fachverfahren sind bisher an den meisten Stellen auf eine synchrone Verarbeitung ausgelegt. Es gibt einige wenige Ausnahmen (z.B. Statistikerstellung), aber grundlegend laufen alle Anfragen nach dem gleichen synchronen Schema ab. Das Integrationsportal mit einer integrierten Aufgabenverwaltung sowie elektronischer Akte stellt hier keine Ausnahme dar. Im Rahmen der eKP Konzeption und Umsetzung wurde aus verschiedenen Gründen auf asynchrone Aufrufe verzichtet (siehe eKP Architekturkonzept). Allerdings hat sich in den vergangenen Jahren gezeigt, dass diese Entscheidung nicht für alle Szenarien sinnvoll und durchhaltbar ist.	
Annahmen		
Motivation	Stabilität, Überwachbarkeit und Betreibbarkeit muss sichergestellt sein	
Alternativen	1. Synchrone Services 2. Asynchrone Services	
Entscheidung	Grundlegend wird Variante 1) umgesetzt und verwendet. Für Bereiche, die asynchrone Services erforderlich machen, sind diese zu verwenden, hierfür sind klare Richtlinien für Variante 2 auszuarbeiten.	

Themenbereich	Prozesssteuerung
Erläuterung der Entscheidung	Grundlegend wird das Integrationsportal auf synchrone Services ausgelegt sein, da der Endbenutzer am Client eine synchrone Antwort seiner Aktion erwartet. Sobald allerdings bestimmte Rahmenbedingungen (z.B.: erwartetes Antwortzeitverhalten) vorliegen bzw. geschätzt werden, werden diese Komponenten als asynchrone Services ausgeprägt. Dabei ist im Integrationsportal zwischen serverseitigen und clientseitigen asynchronen Aufrufen zu unterscheiden. Es kommt eine Rich Client Architektur für den Client auf Basis von Eclipse RCP zum Einsatz. Innerhalb des Clients sind auch mit synchronen Services asynchrone Hintergrundaktivitäten umsetzbar. Diese werden dann eingesetzt, wenn der Benutzer trotz Aktion weiterarbeiten soll.
Auswirkungen	Durch den Einsatz von asynchronen Abläufen verändern sich ggf. auch bestehende Anwendungsfälle, da die Antwort und meist auch die technischen Transaktionen unterschiedlich ausgeprägt werden müssen (z.B. Einführung von Sperrzuständen von Daten, Verwaltung der Sperrzustände bei Anwendungsfehlern usw.). Dies bedeutet, das asynchrone Abläufe oft auch komplexer zu verwalten und zu beherrschen sind, was auch alternative Testkonzepte notwendig macht. Diese Testkonzepte müssen vor allem auch die nicht-funktionalen Testbereiche (z.B.: Programmabstürze) hinreichend mit betrachten. An die Endanwender stellen asynchrone Abläufe oft auch höhere Anforderungen in der Bedienung, da sie nicht, wie gewohnt, sofort eine Rückmeldung erhalten. Aus diesen Gründen sollen auch in Zukunft asynchrone Services nur sehr begrenzt zum Einsatz kommen.
Abgeleitete Anforderungen	Ein späterer Wechsel auf vollständig asynchrone Services hätte weitreichende fachliche und technische Auswirkungen und müsste konkret erneut betrachtet werden.
Verbundene Entscheidungen	-

5.4 Architekturentscheidung Zugriff auf eIP Services durch Kommunikationspartner

Themenbereich	Prozesssteuerung	
Architekturentscheidung	ID	2.2
Issue oder Problem	Einige eIP Services stellen wiederverwendbare Basisservices dar, die von unterschiedlichen Fachverfahren genutzt werden können. Die Verteilung der Services auf die Infrastrukturkomponenten ist dabei nicht festgelegt. Die Stabilität der Schnittstelle ist für angebundene Nutzer essentiell wichtig.	
Annahmen		
Motivation	Anpassungsaufwand beim Betrieb und bei der Entwicklung reduzieren	
Alternativen	1. Direkter Aufruf der Services aus den Fachverfahren 2. Aufruf einer Steuerungsschicht, die den Serviceaufruf weiterleitet 3. Verwendung eines Integration Webservice für die jeweiligen eIP Services auf Basis von Nachrichten	

Themenbereich	Prozesssteuerung
Entscheidung	Alternative 3 wurde gewählt
Erläuterung der Entscheidung	Alternative 1 bietet die verschiedenen Nachteile, die oben bei „Issue oder Problem“ beschrieben wurden. D.h. bei Umzug eines Service auf eine andere Infrastruktur müssen alle Nutzer ihre Kommunikationsendpunkte anpassen. Weiterhin sind alle Schnittstellenänderungen direkt nach einem Deployment für alle Nutzer sichtbar, weshalb eine Anpassung der Nutzer ggf. notwendig wird. Alternative 2 kann mit Hilfe des Oracle Mediator Konzepts oder des Oracle Service Bus umgesetzt werden. Dabei ruft der Nutzer den Mediator bzw. den entsprechenden Kommunikationsendpunkt auf dem OSB mit seiner Webservice-Schnittstelle auf. Diese Komponente routet die Anfrage an den Service weiter und kann dabei eingehende als auch ausgehende Daten transformieren. Bei einer Änderung des Kommunikationsendpunktes ist nur diese Kommunikationskomponente anzupassen. Transformationsänderungen benötigen eine Umsetzung innerhalb der Kommunikationskomponente. Dies hat unbestrittene Vorteile, erfordert aber zusätzliche Infrastrukturkomponenten. Um die Komplexität zu reduzieren, wird diese Alternative nicht zwingend gefordert. Alternative 3 stellt einen Integration Webservice dar, der Nachrichten für die verschiedenen eIP Services verarbeiten kann. In dem Fall können die gleichen Nachrichten wie bei der Interprozesskommunikation genutzt werden. Dies bietet somit die Flexibilität, einzelne Nachrichten entweder über die clientseitige oder serverseitige Schnittstelle umzusetzen. Schnittstellenänderungen der fachlichen Nachrichten können über versionierte Nachrichten erreicht werden.
Auswirkungen	Die eIP Services stellen über den Integration Webservice eine identische, nachrichtenbasierte Schnittstelle für die Aufrufe der fachlichen Funktionen bereit. Die Verwendung eines Enterprise Service Bus ist mit dieser Technik weiterhin möglich. In dem Fall wird der ESB als Webservice-Proxy eingesetzt.
Abgeleitete Anforderungen	Die Webservice Schnittstellen wird vom JEE Container zur Verfügung gestellt. Die Nachrichten müssen ein Versionierungskonzept unterstützen, um auch eine Abwärtskompatibilität der Schnittstelle zu ermöglichen.
Verbundene Entscheidungen	5.9

5.5 Architekturentscheidung Transaktionssteuerung

Themenbereich	Transaktionssteuerung		
Architekturentscheidung		ID	3.1
Issue oder Problem	Die Veränderung an mehreren Datensenken innerhalb einer fachlichen Transaktion kann unterschiedlich abgebildet werden. Entweder kommen übergreifende Transaktionen (XA TX) oder alternativ dazu Compensation Tasks zum Einsatz. Es muss eine generelle Vorgabe existieren, mit deren Hilfe entschieden werden kann, welche Transaktionssteuerung umgesetzt werden muss.		
Annahmen			
Motivation	Stabilität, Konsistenz und Überwachbarkeit muss sichergestellt sein		

Themenbereich	Transaktionssteuerung
Alternativen	1. XA-Transaktionen 2. Compensation Tasks
Entscheidung	Grundlegend wird Variante 1) umgesetzt, es werden allerdings konkrete Vorgaben für die Nutzung und die Auswirkungen der Alternative ausgearbeitet
Erläuterung der Entscheidung	Die Vorteile für die Anwendungsentwicklung, Wartung und Pflege überwiegen an einigen Stellen die betrieblichen Aspekte. Aus diesem Grund werden technische Transaktionen eingesetzt, um die Transaktionssicherheit herzustellen. Dabei kommen auch XA-Transaktionen zum Einsatz. Compensation kommt nur bei dedizierten asynchronen Aufrufen und bei Services, die keine XA-Transaktionen unterstützen oder bei langlaufenden Aktionen zum Einsatz.
Auswirkungen	Der Einsatz von synchronen und asynchronen Services leitet sich z.T. aus dieser Entscheidung ab. Asynchrone Services können nie in eine XA Transaktion eingebunden werden, womit in dem Fall nur Kompensation zum Einsatz kommen kann.
Abgeleitete Anforderungen	Ein späterer Wechsel kann zu veränderten fachlichen Abläufen und Anpassungen an bestehenden Anwendungsfällen führen. Weiterhin sind die betrieblichen Auswirkungen detailliert darzulegen.
Verbundene Entscheidungen	2.3

5.6 Architekturentscheidung Stateful vs. Stateless Session Beans

Themenbereich	Komponentendesign	
Architekturentscheidung	ID	4.1
Issue oder Problem	Für verschiedene Anwendungsfälle sind Informationen über den aktuell angemeldeten Benutzer oder vorangegangene Schritte essentiell. Diese Statusinformationen müssen somit übertragen werden oder in einem Statusspeicher am Server gehalten werden. Dazu bieten sich grundsätzlich mehrere Varianten an. Einige etablierte Justiz-Anwendungen nutzen ein Stateful Session Bean, um Anmeldeinformationen serverseitig zu halten. Andere Justiz-Anwendungen verzichten bereits vollständig auf Stateful Session Bean, da keinerlei Statusinformationen gehalten werden müssen.	
Annahmen		
Motivation	Stabilität und Betreibbarkeit muss sichergestellt sein	
Alternativen	1. Stateful Session Beans 2. Stateless Session Beans	
Entscheidung	Es wird Variante 2 umgesetzt	
Erläuterung der Entscheidung	Die Vorteile von Stateless Session Beans und damit umgekehrt die Nachteile von Stateful Session Beans lassen sich folgendermaßen zusammenfassen:	

Themenbereich	Komponentendesign
	• Pooling von Bean-Instanzen problemlos möglich • Identische Bean-Instanzen für alle Benutzer • Wiederverwendung einer Bean-Instanz jederzeit möglich • Verbesserte Lastverteilung • Jeder Request eines Clients an jeden beliebigen Server möglich • Lastverteilung bei jedem Request neu möglich • Neue Server im Cluster werden bei der Lastverteilung sofort berücksichtigt • Stateful: Bean-Instanzen verbleiben auf initialem Server, neue Instanzen können auf neuen Servern erstellt werden • Geringerer Verwaltungsaufwand • Stateless Session Beans: keine Statusinformationen zu verwalten • Impliziter Cluster-Support • Kein Replikationsmechanismus für Statusinformationen o.ä. notwendig • Stateful Session Bean FailOver-Support teuer durch Replikationsmechanismus Ein gravierender Nachteil von Stateful Session Beans ist, dass pro Zeiteinheit nur eine Anfrage pro Benutzer bearbeitet werden kann. Dies ist in der EJB Spezifikation begründet, welche einen Single-Threaded-Zugriff auf eine Bean-Instanz zusichert. Nachdem es von einem Stateful Session Bean nur genau 1 Instanz pro Benutzer existiert, können auch nicht mehrere Anfragen pro Zeiteinheit verarbeitet werden. Dies ist aber bei Hintergrund-Jobs, die vom Client initiiert werden, zwingend notwendig, um den Online-Zugriff nicht zu behindern. Die Konsequenz ist, dass für das Integrationsportal auf Stateful Session Beans verzichtet wird und Stateless Session Beans durchgängig analog zur eKP Architektur zum Einsatz kommen.
Auswirkungen	-
Abgeleitete Anforderungen	Die Entwicklung muss sich an die JEE Programmiervorgaben für EJBs (z.B. keine Verwendung von Threads) halten.
Verbundene Entscheidungen	-

5.7 Architekturentscheidung Konvertierungsvorgehen

Themenbereich	Komponentendesign		
Architekturentscheidung		ID	4.2
Issue oder Problem	In der elektronischen Gerichtsakte können nur PDF-Dokumente, aus Sicht der Revisionssicherheit sogar nur PDF/A-Dokumente abgelegt werden. Dies ist für die Unveränderlichkeit als auch für die seitenstabile Darstellung zwingend notwendig. Es muss eine Entscheidung getroffen werden, zu welchen Zeitpunkten mit welchen Produkten und Komponenten die bestmöglichen Ergebnisse ohne Beeinflussung des Endbenutzers erzielt werden können. Aus Gründen der Barrierefreiheit muss ein durchsuchbarer Textlayer im PDF/A Dokument enthalten sein.		
Annahmen			

Themenbereich	Komponentendesign
Motivation	Hohe Qualität, hohe Performance, Stabilität und Betreibbarkeit muss sichergestellt sein
Alternativen	1. Nutzung des Scan Subsystems 2. Nutzung des RenditionService des DMS 3. Nutzung von clientseitigen Konvertierungsmethoden aus Kaufprodukten wie Nuance Power PDF Advanced 4. Nutzung von Microsoft Office 5. Nutzung von Open Office / Libre Office 6. Nutzung eines zentralen Rendition Service
Entscheidung	Es wird eine Kombination aus Variante 1, 3, 4 und 6 umgesetzt
Erläuterung der Entscheidung	Im Rahmen der Posteingangsbearbeitung und des Bestandsaktenscans erfolgt eine Konvertierung nach PDF/A mit OCR innerhalb des Scan Subsystems. Damit sind diese Eingänge mit einer gewissen Qualität verbunden. Der Einsatz eines alternativen Konvertierungsdienstes könnte potentiell schlechtere Ergebnisse liefern. Der Nachteil ist die notwendige Kommunikation mit einem Serversystem und damit auch die Übertragung der Dokumente. Der Rendition Service eines DMS ist produktabhängig und erhöht die Anforderung an die DMS Produktauswahl und limitiert somit die Prämisse der Produktunabhängigkeit. Die Nutzung einer clientseitigen Konvertierung ist ebenfalls ein Fremdprodukt, welches unterschiedliche Ergebnisse zum Scan Subsystem liefern könnte und außerdem gesondert ausgerollt und ggf. überwacht werden muss. Dafür können diese Produkte wie z.B. Nuance Power PDF Advanced verschiedene Formate konvertieren. Microsoft Office liefert sehr schnell sehr gute Ergebnisse bei den eigenen Formaten inkl. Textlayer. Allerdings können bei alternativen Formaten, die laut Rechtsverordnung (z.B. TIF) zugelassen sind, die MS Office Produkte nicht eingesetzt werden. Die MS Office Produkte sind beim Auftraggeber meist lizenziert und am Client installiert. Open Office / Libre Office können clientseitig als auch serverseitig eingesetzt werden, haben allerdings vor allem mit den aktuellsten MS Office Formaten (OpenXML) Darstellungsprobleme, was sich bei einer Konvertierung verbietet. Eine weitere Möglichkeit sind Zusatzprodukte wie z.B. der Luratech Rendition Server, die als weitere Infrastrukturkomponente betrieben und als Konvertierungsdienst angesprochen werden können. Aufgrund der Forderung nach möglichst geringen Kopplungen und Berücksichtigung von möglichen Serverausfällen wird eine Kombination der Möglichkeiten umgesetzt: a) primäre Konvertierung erfolgt über eine Webservice Schnittstelle, die das Scansubsystem und optional ein weiterer Rendition Server implementiert b) alternativ kann Microsoft Office für die MS Office Formate genutzt werden c) zusätzlich wird eine Konvertierung durch Ablage in Ordner und Auslesen aus einem Konvertierungsordner angeboten, welche auf Basis von Nuance Power PDF Advanced umgesetzt wird
Auswirkungen	-
Abgeleitete Anforderungen	Es muss eine Rückfallstrategie bei Ausfall einer Konvertierungskomponente umgesetzt werden, um den produktiven Betrieb nicht einzuschränken. Die Konfiguration muss so ausgestaltet sein, dass für jedes Dateiformat mehrere Konvertierungskomponenten konfiguriert und mit Priorität versehen werden können.

Themenbereich	Komponentendesign
	Neben den konvertierten Dokumenten müssen immer auch die Originaldokumente im DMS abgelegt werden, damit ein Vergleich mit dem Original möglich ist. Wenn ein Dokument bereits als PDF eingeliefert wird, kann über die Güte des Textlayers keine Aussage getroffen werden. Aus diesem Grund werden diese Dokumente erneut in ein PDF/A mit OCR Erkennung konvertiert mit entsprechendem Textlayer erzeugt. Elektronische Eingänge können von der eKP automatisch im Rahmen des Empfangsprozesses durch Aufruf des Scan Subsystems bzw. des zentralen Konvertierungsservices konvertiert und im Zwischenspeicher mit Relation zum Originaldokument abgelegt werden. Dadurch entfallen die Konvertierungen während einer Benutzerinteraktion und die entsprechende Wartezeit. Bereits konvertierte Dokumente müssen durch entsprechende Metadaten gekennzeichnet werden, um eine doppelte Konvertierung bei späteren Aktivitäten zu vermeiden. Das Scan Subsystem muss externe, durch Java aufrufbare Schnittstellen für die Konvertierung in Form eines Webservice bereitstellen. Diese müssen für die Konvertierung auf Basis von Dokumenten im eKP Zwischenspeicher und auch auf Basis von übergebenen Dokumenten ausgelegt sein. Weitere Konvertierungsdienste müssen die identische Webservice-Schnittstelle umsetzen, damit die verschiedenen Dienste problemlos ausgetauscht werden können.
Verbundene Entscheidungen	-

5.8 Architekturentscheidung Kommunikationstechnik für Interprozesskommunikation zwischen Anwendungen

Themenbereich	Kommunikationsarchitektur	
Architekturentscheidung	ID	5.1
Issue oder Problem	Innerhalb der Eclipse Rich Client Platform sind verschiedene Kommunikationsprotokolle zwischen den verschiedenen Plugins technologisch möglich. Es ist sinnvoll, die Möglichkeiten zu beleuchten und die für eIP nutzbaren Varianten festzulegen.	
Annahmen		
Motivation	Hohe Qualität, Stabilität und Einheitlichkeit für eine gute Wartbarkeit muss sichergestellt sein; möglichst geringer Integrationsaufwand und Infrastrukturaufwand	

Themenbereich	Kommunikationsarchitektur
Alternativen	1. Direkte Nutzung der Java Schnittstellen 2. Direkte Socket-basierte Kommunikation (lokaler Webservice, MBean, RESTful Service) 3. Kommunikation über Named Pipes 4. Nutzung von URL Handlern 5. COM[5] 6. MQTT[6]
Entscheidung	Es wird Variante 3 und optional 6 umgesetzt, in Einzelfällen kann Variante 1 oder 5 zum Einsatz kommen.
Erläuterung der Entscheidung	Die direkte Nutzung der Java-Schnittstellen ist nur für Java-basierte Rich Client Anwendung mit identischer Java Virtual Machine für eIP und Fachverfahren möglich. Die Vorteile sind sehr schnelle, performante, typisierte Schnittstellen mit klarer Fehlerbehandlungsstrategie und synchroner Request/Response Umsetzung. Diese verstärkte Kopplung der Anwendungen wird nicht von allen Fachverfahren gewünscht und wird für Fachverfahren anderer Entwicklungsverbünde nicht empfohlen und genutzt. Eine direkte Socket-basierte Kommunikation zwischen eIP und der integrierten Fachanwendung hat einen gravierenden Nachteil in einer Terminalserverumgebung aufgrund fehlender Portvirtualisierung und Sicherheitsaspekte. Es sind dabei zusätzliche Schnittstellen zur Identifikation des Kommunikationsports notwendig, wobei synchrone Request/Response-Kommunikation problemlos umsetzbar wäre. Aufgrund dieser Problematik ohne direkte Mehrwertdienste wird dieser Ansatz nicht genutzt. Die Kommunikation über Named Pipes (bzw. Mailslots, die sich ähnlich verhalten) stellt technisch eine stabile und bewährte Kommunikationsart dar, die vom Betriebssystem zur Verfügung gestellt wird und je nach Betriebssystem unterschiedlich gehandhabt wird. Request/Response ist entweder über bidirektionale Pipes oder mehrere Pipes zu lösen, wobei die Zuordnung der relevanten Nachrichten über Correlation-Ids zu lösen ist. Der Zugriff bzw. die Bereitstellung per Java sind über Zugriff auf die Windows API (native Bibliotheken) möglich. Der Vorteil von Named Pipes ist, dass keine zusätzliche Infrastrukturkomponenten benötigt werden und erweiterte Absicherungen über das Betriebssystem möglich sind. Die Anbindung von Webanwendungen ist über Named Pipes ohne Zusatzkomponente nicht möglich. Die Kommunikation über Pipes stellt vor allem für Rich Client Anwendungen auf Windows Basis die bevorzugte Option dar. URL-Handler sind für ein Protokoll im Windows hinterlegte Anwendungen, welche automatisch gerufen werden. Der entscheidende Nachteil ist dabei, dass jedes Mal ein neuer Prozess vom Betriebssystem gestartet wird, um den URL-Aufruf zu behandeln. Der jeweilige Prozess bzw. die Logik in der Fachanwendung muss dann den Einsprung in bestehende, laufende Instanzen eigenständig lösen. Dies ist z.B. für Java Anwendungen derzeit nur problematisch über Java Shared Memory o.ä. zu erreichen, womit keine konkrete Instanz einer Anwendung adressiert werden kann. Weiterhin sind die Fehlerbehandlung und der Transport von großen Datenmengen nicht spezifiziert und genau zu definieren und zu prüfen. Auf Grund dieser Nachteile werden URL Handler nicht genutzt, obwohl ein synchroner Request/Response Betrieb einfach möglich wäre.

[5] Component Object Model, siehe https://de.wikipedia.org/wiki/Component_Object_Model

[6] Message Queue Telemetry Transport, siehe http://mqtt.org/

Themenbereich	Kommunikationsarchitektur
	Eine COM-Schnittstelle ist nur für native Windows-Anwendungen unter Einsatz bestimmter Frameworks möglich. Dabei sind synchrone Methodenaufrufe mit Request/Response problemlos möglich und es können COM-Events für die aktive Benachrichtigung genutzt werden. Dies stellt somit für native Windows Anwendungen eine valide Option dar, welche zusätzlich angeboten werden kann. MQTT stellt einen leichtgewichtigen Nachrichtentransport für Publish-Subscribe Mechanismen dar, der vor allem in der Gerätesteuerung weit verbreitet ist. Dies stellt ein zusätzliches Broker-Produkt am Client dar (z.B. Mosquitto[7]), allerdings sind neben dem reinen Nachrichtentransport auch Mehrwertdienste wie garantierte Zustellung o.ä. vorhanden. Für die Kommunikation ist eine Socket-Kommunikation notwendig, die allerdings durch eine zentrale MQTT Instanz pro Client (auch bei Terminalserver / Citrix) gelöst werden kann. Mit MQTT sind weiterhin Push-Dienste möglich, so dass auch ein Server gezielt Nachrichten an eine oder viele Clients übermitteln kann. Diese Möglichkeiten sind derzeit in der eIP Architektur nicht berücksichtigt und könnten mit diesem Ansatz ebenfalls umgesetzt werden. Aus diesen Gründen wird MQTT als weiterer optionaler Transportkanal angesehen, da z.B. über WebSockets auch Webanwendungen angebunden werden können. Allerdings wird für den Betrieb eine zusätzliche Infrastrukturkomponente (MQTT Broker wie mosquitto) benötigt, der installiert und clientseitig betrieben werden muss.
Auswirkungen	mosquitto bietet bereits eine Unterstützung von WebSockets an, so dass die Einbindung von Webanwendungen ebenfalls vereinfacht wird. Ohne mosquitto ist dies nicht gegeben. Aus diesem Grund wird für die Kommunikation mit einer Webanwendung ein WebSocket-Server in eIP umgesetzt, um darüber die Kommunikation zu ermöglichen.
Abgeleitete Anforderungen	Es sind erweiterte Überwachungsmechanismen für das MQTT am Client notwendig. Named Pipes sollten über Betriebssystemmittel vor unberechtigtem Zugriff abgesichert werden. Der Transportkanal soll aus Gründen der Austauschbarkeit durch eine Integrationsbibliothek versteckt werden. Dies bedeutet, dass Anwendungen eine Bibliothek für die bidirektionale Kommunikation zur Verfügung gestellt wird, der eigentliche Transportkanal aber nicht unmittelbar sichtbar ist.
Verbundene Entscheidungen	5.2, 5.4

5.9 Architekturentscheidung Datenformat für Interprozesskommunikation zwischen Anwendungen

Themenbereich	Kommunikationsarchitektur		
Architekturentscheidung		ID	5.2
Issue oder Problem	Für die Kommunikation zwischen den Anwendungen müssen Daten ausgetauscht werden. Nachdem als Kommunikationstechnik ein beliebiger Kommunikationskanal genutzt werden soll, muss das Datenformat entsprechend darauf ausgelegt sein. Ein		

[7] http://mosquitto.org/

Themenbereich	Kommunikationsarchitektur
	Transport als reiner Text (String) bietet dabei die maximale Kompatibilität zwischen unterschiedlichen Technologien.
Annahmen	
Motivation	Hohe Qualität, Stabilität und Einheitlichkeit für eine gute Wartbarkeit muss sichergestellt sein
Alternativen	1. Nutzung von XML 2. Nutzung von Protocol Buffers 3. Nutzung von JSON
Entscheidung	Es wird Variante 1 umgesetzt
Erläuterung der Entscheidung	Protocol Buffers (Variante 2) stellt eine hochperformante Technologie dar, bei der die Daten kodiert transportieren werden können. Die Technologie ist wenig verbreitet und es sind derzeit zusätzliche Bibliotheken notwendig, um Protocol Buffers ansprechen zu können. Neben typisierten Daten ist eine Prüfung der Nachrichten auf Korrektheit sowohl für den Empfänger als auch Absender möglich. Aufgrund der geringen Verbreitung und der zusätzlichen Anforderungen wird diese jedoch nicht eingesetzt. JSON stellt eine sehr einfach zu implementierende Technologie auf String-Basis dar, welche einfach Schlüssel-Wert-Paare ohne Typisierung und semantischer Beziehung abbildet. Somit können keine Standardprüfungen für Nachrichten erfolgen und Formatumwandlungen (z.B. Text → Zahl) müssen manuell je nach Attribut durchgeführt werden. Aufgrund dieser Nachteile wird diese Variante nicht gewählt. XML stellt eine bekannte Technologie dar, die vor allem durch den Einsatz von Webservices eine große Verbreitung erlangt hat. Es sind typisierte Daten möglich, die per Schemadefinition festgelegt sind und die darüber für Empfänger und Absender eine Prüfung ermöglichen. Die Verarbeitung von XML ist allerdings langsamer als bei den anderen Varianten. Nachdem allerdings die Aufrufhäufigkeit und die übermittelte Datenmenge eher gering ist, wird dieser Nachteil bewusst in Kauf genommen und diese Variante für die Umsetzung gewählt.
Auswirkungen	Die externen Schnittstellen von eIP werden in Domänen unterteilt (z.B. Sicherheit, Aktenbockaufgaben, Suche usw.). Nachdem nicht alle Domänen von denselben Anwendungen implementiert werden (müssen), wird für jede Domäne ein eigenes Schema mit eigenem Namespace definiert. Dieses Schema beinhaltet alle Operationen mit Ein- und Ausgabeparametern. Wenn eine Anwendung mehrere Domänen bedient, muss diese entsprechend die jeweiligen Schemata nutzen.
Abgeleitete Anforderungen	Es müssen Schemata für die Schnittstellen entworfen werden, die für die Prüfung genutzt werden können. Weiterhin müssen die Schnittstellen versionierbar sein, damit mehrere Versionen der Schnittstellen parallel existieren und genutzt werden können.
Verbundene Entscheidungen	5.1

5.10 Architekturentscheidung Kommunikation innerhalb von Eclipse RCP

Themenbereich	**Kommunikationsarchitektur**		
Architekturentscheidung		ID	5.3
Issue oder Problem	Innerhalb der Eclipse Rich Client Platform sind verschiedene Protokolle für die Kommunikation zwischen verschiedenen Plugins technologisch möglich. Es ist sinnvoll, die Möglichkeiten zu beleuchten und die für eIP nutzbaren Varianten festzulegen.		
Annahmen			
Motivation	Hohe Qualität, Stabilität und Einheitlichkeit für eine gute Wartbarkeit muss sichergestellt sein		
Alternativen	1. Direkte Nutzung der Java Klassen der anderen Plugins durch exportierte Packages 2. Nutzung des OSGi Event Service 3. Nutzung von Extension Points 4. Nutzung der Kommunikationsprotokolle für Interprozesskommunikation (siehe 5.8)		
Entscheidung	Es wird eine Kombination aus Variante 1, 2 und 3 umgesetzt		
Erläuterung der Entscheidung	Jede Alternative erfüllt andere Anforderungen und benötigt andere Voraussetzungen. Es ergeben sich daraus unterschiedliche Kopplungsgrade – von loser bis starrer Kopplung, die je nach Einsatzzweck sinnvoll oder weniger sinnvoll sein können. Die direkte Nutzung der Java-Klassen anderer Plugins stellt die effizienteste und schnellste Kommunikationsart dar. Sie wird allerdings nur in Sonderfällen für stark gekoppelte Plugins (z.B. forumSTAR und forumSTAR-Text) genutzt, da damit eine enge Kopplung erfolgt. eIP stellt seine Komponenten und Services über diesen Weg zur Verfügung, da dieser von vielen Plugins typischerweise als Basisdienst verwendet wird. Für andere Plugins muss dies speziell abgestimmt und dokumentiert werden. Die Nutzung des OSGi Event Service stellt die loseste Kopplungsart dar, da diese rein auf Events basiert. Dabei stellt ein Plugin einen Event ein und 0..n andere Plugins können darauf reagieren. Eine synchrone Antwort ist mit diesem Mechanismus nicht umsetzbar, da ein Rückgabewert nicht vorgesehen ist und es 0..n Empfänger sein können, womit die Antwort wiederum einen asynchronen Event darstellen würde. Dieser Mechanismus stellt die bevorzugte Kommunikationsart innerhalb von eIP dar, wenn keine Basisdienste von eIP involviert sind. Die Nutzung von Extension Points ist ein Standardmechanismus von Eclipse RCP, um definierte Schnittstellen an einem Plugin zur Verfügung zu stellen. Dabei können andere Plugins nach Implementierern dieser Schnittstelle (d.h. nach Bereitstellern des Extension Points) suchen und diese gezielt ansprechen. Dieser Mechanismus ist bei klar definierten, mehrfach wiederverwendbaren Schnittstellen sinnvoll, die von mehreren Plugins umgesetzt werden sollen. Durch das notwendige Suchen und Auffinden der Extension Points ist dies nicht für alle Anwendungsfälle sinnvoll. eIP nutzt Extension Points für die definierten Schnittstellen, welche eIP von anderen Komponenten erwartet (z.B. Unterstützung der Suche durch SearchProvider, Sicherheitsdienst (SecurityProvider), Lieferant für Aktenbockaufgaben (WorklistProvider)). Die möglichen Kommunikationsprotokolle für die Interprozesskommunikation könnten auch innerhalb von Eclipse RCP eingesetzt werden. Allerdings verlassen diese alle den aktuellen Prozessraum und haben somit einen erhöhten Overhead für Protokollumset-		

Themenbereich	Kommunikationsarchitektur
	zung (z.B. XML Marshalling) und Transport. Aus diesem Grund kommen diese innerhalb einer Java VM nicht zum Einsatz.
Auswirkungen	-
Abgeleitete Anforderungen	-
Verbundene Entscheidungen	-

5.11 Architekturentscheidung Integrationsbibliothek

Themenbereich	Kommunikationsarchitektur	
Architekturentscheidung	ID	5.4
Issue oder Problem	Die Anbindung der Fachanwendungen an eIP soll so einfach wie möglich umgesetzt werden können, um die Integrationskosten, die Abstimmungsaufwände und die Abwehrhaltung niedrig zu halten.	
Annahmen		
Motivation	Stabilität und Einheitlichkeit für eine gute Wartbarkeit muss sichergestellt sein; möglichst geringer Integrationsaufwand	
Alternativen	1. Direkte Nutzung der Kommunikationstechnik 2. Bereitstellung einer Integrationsbibliothek	
Entscheidung	Es wird Variante 2 umgesetzt.	
Erläuterung der Entscheidung	Bei der ersten Variante muss jede Fachanwendung die jeweilige Kommunikationstechnik (z.B. MQTT oder Named Pipes-Anbindung) umsetzen und die Versand- und Empfangskanäle nutzen, überwachen, korrelieren und umsetzen. Dies erhöht die Anforderungen an die Fachanwendungen und das entsprechend notwendige Wissen. Bei Bereitstellung einer Integrationsbibliothek (Variante 2) bindet der Entwickler des jeweiligen Fachverfahrens die von eIP bereitgestellte Bibliothek ein und muss nur noch die Nachrichten für den Versand bereitstellen bzw. die Nachrichten für einen Empfang entgegennehmen. Der Transportkanal ist vollständig in der Bibliothek gekapselt, so dass ein späterer Wechsel des Kanals ohne Auswirkung auf die angebundenen Fachanwendungen wäre. Nachteilig an dieser Lösung ist, dass eIP für unterschiedliche Programmiersprachen (z.B. Java, C++, C#, C usw.) entsprechende Bibliotheken bereitstellen muss. Dies wird aufgrund der Kapselung des Transportkanals und der vereinfachten Nutzung für die zu integrierende Anwendung in Kauf genommen.	
Auswirkungen	Die Integrationsbibliotheken regeln die Kommunikation mit dem Integrationsportal und müssen thread-safe implementiert werden, so dass auch parallele Anfragen aus einer Anwendung korrekt umgesetzt werden. Die darunterliegende Kommunikationsarchitektur ist für die angebundene Anwendung nicht sichtbar (Ausnahme: lokale Konfigurationsparameter). In Zukunft sind typisierte bzw. teiltypisierte Schnittstellen als High-Level-APIs denkbar. Eine entsprechende Entscheidung wird erst nach den ersten Integrationsprojekten getroffen.	

Themenbereich	Kommunikationsarchitektur
	Mit dieser Vorgehensweise wird eine möglichst lose Kopplung zwischen verschiedenen Anwendungen erreicht, die durch dedizierte Aufrufmethode aber optimalen Komfort bieten. Es wird eine Interprozesskommunikation mit geringen Abhängigkeiten zwischen den verschiedenen Produkten angeboten.
Abgeleitete Anforderungen	-
Verbundene Entscheidungen	-

5.12 Architekturentscheidung visuelle Integration von Fremdanwendungen

Themenbereich	Kommunikationsarchitektur	
Architekturentscheidung	ID	5.6
Issue oder Problem	Die Anbindung der Fachanwendungen an eIP soll so einfach wie möglich umgesetzt werden können, um die Integrationskosten, die Abstimmungsaufwände und die Abwehrhaltung der Projekte niedrig zu halten. Dabei ist die visuelle Integration aus Sicht von eIP ein wichtiger Aspekt.	
Annahmen		
Motivation	Stabilität und Einheitlichkeit für eine gute Wartbarkeit muss sichergestellt sein	
Alternativen	1. Einfangen aller Fenster einer Fachanwendung 2. Einfangen der Fenster unter Mitteilung des Window Handle 3. Integrierte Darstellung von Java Anwendungen durch Nutzung der Java Möglichkeiten 4. Integrierte Darstellung unter Einsatz von OLE Controls (ActiveX Controls)	
Entscheidung	Es wird Variante 2 favorisiert umgesetzt, mit der Möglichkeit, Variante 3 und 4 ebenfalls zu verwenden.	
Erläuterung der Entscheidung	Die direkte Nutzung der Java Schnittstellen (Variante 3) ist nur für Java-basierte Rich Client Anwendung mit identischer Java Virtual Machine für eIP und Fachverfahren möglich. Bei Eclipse RCP Anwendungen muss weiterhin die identische Eclipse RCP Version vorliegen. Die Vorteile sind eine sehr gute Integrationsmöglichkeit durch z.B. Einhängen in die Event-Queues und automatischer Darstellung der korrekten Fenster im eIP Rahmen. Diese verstärkte Kopplung der Anwendungen wird nicht von allen Fachverfahren gewünscht und wird für Fachverfahren anderer Entwicklungsverbünde nicht empfohlen und genutzt. Die Nutzung von OLE Controls (bzw. ActiveX Controls) ist eine altbewährte Technik von Microsoft, um Anwendungen bzw. Anwendungsteile aus anderen Bereichen in eigenen Prozessen nutzen und verwenden zu können. Die Microsoft Standardprodukte wie Microsoft Office bieten bereits entsprechende Controls, weshalb eine Einbindung und Integration sowie visuelle Darstellung darüber möglich ist. Wenn eine Anwendung bereits entsprechende Schnittstellen und Controls zur Verfügung stellt, können diese auch sofort bei der visuellen Integration in eIP genutzt werden und damit einen ent-	

Themenbereich	Kommunikationsarchitektur
	sprechend tief integrierten Charakter erfüllen. Wenn entsprechende Controls nicht vorliegen, wird diese Variante 4 nicht favorisiert, da die Aufwände bei der Fachanwendungsseite sehr hoch sein können. Bei der visuellen Integration über Einfangen der Fenster wird ein Anwendungsfenster von eIP eingefangen und der Vater-Handle (ParentHandle) neu auf einen View (Part) innerhalb der eIP Fensterverwaltung gesetzt (Variante 1 und 2). Damit unterliegt das Fenster der Steuerung von eIP und kann somit ebenso minimiert, maximiert, auf andere Monitore verschoben oder in den Vollbildmodus versetzt werden. Grundsätzlich bestehen dabei zwei Möglichkeiten, die Fenster, die integriert dargestellt werden sollen, zu identifizieren. In Variante 1 werden alle Fenster einer Anwendung eingefangen, in dem sie z.B. durch Einhängen in die Windows Event Queue oder durch kontinuierliches Abfragen der einem Prozess zugeordneten Fenster identifiziert und integriert werden. In Variante 2 erfolgt eine Mitteilung des Fenster-Handles (WindowHandle) durch die Anwendung an eIP zum Einfangen des jeweiligen Fensters. In eIP wird aktuell nur Variante 2 umgesetzt. Damit liegt die Verantwortung der Identifikation der zu integrierenden Fenster bei der integrierten Anwendung. Nachteilig daran ist die minimale Anpassung in der jeweiligen Anwendung, die Vorteile der Identifikation der Fenster überwiegen diesen Anpassungsaufwand jedoch deutlich.
Auswirkungen	-
Abgeleitete Anforderungen	Details zu den abgeleiteten Anforderungen an zu integrierende Anwendung sind im Integrationsleitfaden erfasst.
Verbundene Entscheidungen	-

5.13 Architekturentscheidung Integration von Webanwendungen

Themenbereich	Kommunikationsarchitektur		
Architekturentscheidung		ID	5.7
Issue oder Problem	Die Anbindung der Fachanwendungen an eIP soll so einfach wie möglich umgesetzt werden können, um die Integrationskosten, die Abstimmungsaufwände und die Abwehrhaltung der Projekte niedrig zu halten. Fachverfahren auf Basis von Webtechnologien sind ebenfalls verfügbar und müssen ebenfalls Kommunikationsmöglichkeiten am Client erhalten sowie eine visuelle Integration ermöglichen.		
Annahmen			
Motivation	Stabilität, Sicherheitsanforderungen und Einheitlichkeit für eine gute Wartbarkeit muss sichergestellt sein		
Alternativen	1. Umsetzung einer JavaScript-Integrationsbibliothek für die Kommunikation zwischen Webanwendung und eIP Rahmen 2. Umsetzung einer rein serverseitigen Kommunikationslösung 3. Nutzung des Eclipse RCP Webbrowser-Plugins und der Java-JavaScript-Bridge		
Entscheidung	Es wird Variante 1 umgesetzt.		

Themenbereich	Kommunikationsarchitektur
Erläuterung der Entscheidung	Alle drei Varianten haben Vor- und Nachteile. Bei Variante 1 ist eine Kommunikationslösung aus dem Browser mit lokalen Ressourcen notwendig. Dabei bestehen Bedenken bzgl. der grundsätzlichen Sicherheit und der Umgehung der Sandbox des Webbrowsers. Weiterhin sind für eine sichere Kommunikation operative Probleme bei der Zertifikatsverwaltung möglich. Ob und in wie weit Absicherungsmaßnahmen bei einer lokalen Verbindung notwendig sind, steht demgegenüber allerdings noch in Diskussion. Eine lokale Kommunikation via WebSockets ist bidirektional möglich. Diese Lösung ist unabhängig vom Browserhersteller nutzbar. Bei Variante 2 ist der Nachteil, dass Push-Mechanismen für die Verteilung der Nachrichten an den Client notwendig sind sowie ein erheblicher Kommunikationsoverhead besteht. Weiterhin ist eine Nutzung dieser Technik bei Ausfall der Serverinfrastruktur nicht möglich, was den Vorgaben zur Offline-Fähigkeit widerspricht. Variante 3 hat den Nachteil, dass diese nur mit Internet Explorer verwendbar ist und die Weiterentwicklung des Internet Explorers aktuell ungewiss ist. Weiterhin ist mit dem Internet Explorer Control nur eine Websession für alle Browser-Fenster möglich, was z.B. bei web.sta als Hinderungsgrund dieser Lösung angesehen wurde. Dem gegenüber steht eine sehr stabile und schnelle Kommunikationslösung, die lokal am Client ohne Umgehung von Sicherheitsrichtlinien möglich ist. Aufgrund der aufgeführten Nachteile der alternativen Lösungen und der Notwendigkeit mehrere Websessions sowie unterschiedliche Browserhersteller zu unterstützen wird Variante 1 umgesetzt.
Auswirkungen	Für die Integration von Webanwendungen ist ein WebSocket-Server im eIP-Client notwendig. Die visuelle Integration ist durch Einfangen des Browser-Fensters wie bei Fachverfahrensintegrationen umzusetzen. Aufgrund der Sicherheitsbedenken für diese Variante dürfen über diesen Kommunikationsweg nur die absolut zwingend notwendigen Nachrichten übermittelt werden. Alle anderen Nachrichten müssen über die serverseitigen Schnittstellen realisiert werden.
Abgeleitete Anforderungen	Um eine einheitliche Lösung für das Datenformat für alle Integrationsarten aufrecht zu erhalten, werden auch bei der Anbindung von Webanwendungen XML-basierte Nachrichten für den Austausch verwendet. Dies bedeutet, dass im JavaScript Code der Webanwendung XML verarbeitet werden muss. Die Verwendung des Arbeitskontextes und der Umgang mit mehreren Browserfenstern ist mit jeder Webanwendung separat zu klären.
Verbundene Entscheidungen	5.2, 5.4, 5.6

5.14 Architekturentscheidung Datenhaltung elektronische Akte

Themenbereich	Komponentendesign		
Architekturentscheidung		ID	4.3
Issue oder Problem	Die elektronische Akte besteht aus Metadaten, Strukturen, Dokumenten sowie Anmer-		

Themenbereich	Komponentendesign
	kungen (Annotationen) zu Dokumenten. Diese unterschiedlichen Inhalte können technisch in unterschiedlichen Speicherorten abgelegt werden. Eine begründete Ablagestruktur für die Datenhaltung ist anzustreben.
Annahmen	
Motivation	Stabilität und Einheitlichkeit für eine gute Betreibbarkeit (z.B. Backup & Recovery) muss sichergestellt sein
Alternativen	1. Datenhaltung an einer Stelle 2. Aufteilung der Datenhaltung in Metadaten und Dokumente
Entscheidung	Es wird Variante 1 umgesetzt.
Erläuterung der Entscheidung	Bei einer Aufteilung der Datenhaltung ist eine Synchronisation der verschiedenen Datensenken erforderlich. Dies ist vor allem bei Backup & Recovery problematisch, da beide Datensenken zum gleichen Zeitpunkt gesichert und wieder hergestellt werden müssen und sich Veränderungen während der Sicherungsläufe nicht auf die Konsistenz der Daten auswirken dürfen. Bei einer Datenhaltung in einer Stelle ist diese Problematik in der jeweiligen Datensenke (bzw. im jeweiligen Produkt) zu lösen. Als Nachteil kann angesehen werden, dass für die unterschiedlichen Datenarten nicht die optimalste Speichertechnologie zum Einsatz kommen kann. Bei einer Aufteilung ist der Austausch der Datensenke problematischer, da an verschiedenen Stellen der Austausch vorgenommen werden muss. Nachdem aber genau die Datenhaltungskomponente (DMS) im Sinne des Investitionsschutzes (siehe Kapitel 3.4) oftmals festgelegt ist, ist ein einfacher Austausch an einer Stelle von großem Vorteil. Somit wird die Datenhaltung an einer Stelle aufgrund der verschiedenen Vorteile umgesetzt.
Auswirkungen	-
Abgeleitete Anforderungen	Der eingesetzte Datenspeicher muss in der Lage sein, die Anforderungen zu erfüllen.
Verbundene Entscheidungen	-

5.15 Architekturentscheidung Offline-Fähigkeit

Themenbereich	Komponentendesign	
Architekturentscheidung	ID	4.4
Issue oder Problem	Die elektronische Akte soll die Papierakte ablösen und als zentrales Arbeitsmittel für Richter, Rechtspfleger und Servicekräfte werden. Aus diesem Grund ist eine hohe Verfügbarkeit der Lösung zwingend notwendig, um das Arbeiten zu ermöglichen. Dabei sind auch systemseitige Ausfälle sowie Infrastrukturausfälle mit zu betrachten. Dabei sind verschiedene Stufen der Verfügbarkeit zu betrachten. Letztlich soll der Richter auch ohne Zugriff auf die zentralen Systeme in der Lage sein,	

Themenbereich	Komponentendesign
	die elektronische Akte zu lesen (z.B. bei einem Vor-Ort-Besuch).
Annahmen	-
Motivation	Die Entwicklungs- und Betriebskosten dürfen durch eine Hochverfügbarkeit der Lösung nicht über Gebühr steigen.
Alternativen	1. Vollständige redundante Infrastruktur mit dezentraler Verwaltung, ausfallsicheren Leitungen und Fallback-Szenarien 2. Herstellen einer Offline-Fähigkeit im eIP Client mit Caching der relevanten Informationen für einen rein lesenden Zugriff auf die elektronische Akte und den Aktenbock 3. Herstellen einer Offline-Fähigkeit im eIP Client mit Caching der relevanten Informationen für einen lesenden und schreibenden Zugriff auf die elektronische Akte und den Aktenbock 4. Herstellen einer Offline-Fähigkeit im eIP Client mit Caching der relevanten Informationen für einen lesenden und schreibenden Zugriff für alle integrierten und angebundenen Systeme
Entscheidung	Es wird Variante 2 umgesetzt.
Erläuterung der Entscheidung	Die erste Variante ist aus organisatorischen und kostentechnischen Gründen für eine „100% Verfügbarkeit“ nicht umsetzbar. Weiterhin sind die erforderlichen Leitungsmöglichkeiten (z.B. redundante 150Mbit Standleitungen unterschiedlicher Provider) nicht in allen Lokationen / Bundesländern verfügbar. Schreibende Zugriffe werden schnell extrem komplex und aufwändig, da Rücksynchronisationsmechanismen notwendig werden. Diese müssen mit Konfliktsituationen umgehen können oder Behebungen der Konflikte anbieten. Vor allem bei der Veränderung der Gerichtsakte können hier durch stark zeitlich verzögerte konkurrierende Zugriffe automatische Lösungsmöglichkeiten nicht mehr angeboten werden. Eine manuelle Behebung muss durch ein sinnvolles Interaktionskonzept und Führung des Benutzers umgesetzt werden. Die Kosten werden hier als sehr hoch eingestuft, weshalb dieser Ansatz nicht weiter verfolgt wird. Variante 4 stellt dabei nochmals eine Erweiterung dar, die ohne Anpassungen an den jeweiligen Anwendungen nicht realisierbar ist. Aus diesem Grund wird Variante 2 umgesetzt.
Auswirkungen	Schreibende Zugriffe auf die Systeme sind in einem Offline-Fall nicht möglich.
Abgeleitete Anforderungen	Es ist ein clientseitiges Caching für Sicherheitsinformationen, Aktenstrukturen, Aufgaben, Konfiguration sowie Dokumente umzusetzen. Der Offline-Modus soll nicht global für alle Komponenten gelten, sondern soll immer nur die ausgefallene Komponente betreffen.
Verbundene Entscheidungen	5.16 Architekturentscheidung Sicherheitsdelegation

5.16 Architekturentscheidung Sicherheitsdelegation

Themenbereich	Sicherheitsarchitektur		
Architekturentscheidung		ID	6.1
Issue oder Problem	Es muss entschieden werden, wer für Sicherheitsfragen bei Zugriffen auf die elektronische Akte und das eIntegrationsportal zuständig ist. Dabei sollen die bestehenden Infrastrukturkomponenten weiterhin genutzt werden.		
Annahmen	Jedes Fachverfahren bietet eine Authentisierung oder nutzt einen zentralen Dienst dafür. Jedes Fachverfahren bietet eine Autorisierung für seine Benutzer.		
Motivation	Die Administration darf durch die Einführung von eIP nicht maßgeblich erschwert und komplexer werden.		
Alternativen	1. Delegation der Sicherheitsfragen an eine weitere Instanz 2. Replikation der Sicherheitsinformationen nach eIP		
Entscheidung	Es wird Variante 1 umgesetzt.		
Erläuterung der Entscheidung	Durch eine Replikation der Sicherheitsinformationen nach eIP ist eine unabhängige Nutzung des gesamten eIP mit Aufgabenbock, eAkte usw. möglich, da keine Abhängigkeit zu einem externen System für die Übernahme der Sicherheitsfragen besteht. Der entscheidende Nachteil dieser Lösung ist, dass die Sicherheitsinformationen mit dem führenden System abgeglichen werden müssen. Das führende System ist in den meisten Fällen das Fachverfahren. Nur das Fachverfahren kann die Entscheidung treffen, wer welche Aktionen ausführen darf bzw. welche eAkten sehen darf. Eine Verlagerung der Autorisierung aus dem Fachverfahren heraus wird vom Auftraggeber ausgeschlossen. Jedes Fachverfahren hat unterschiedliche Mechanismen für die Umsetzung der Autorisierung. Einige (u.a. forumSTAR) basieren dabei auf Gerichts- und Organisationsstruktur und dem Geschäftsverteilungsplan. Diese Daten müssen dann in die relevanten Sicherheitsinformationen transformiert und in einer eigenen Datenhaltung gespeichert werden. Fachlich ist dabei zu definieren, welcher Zeitversatz bei Autorisierungsänderungen überhaupt akzeptabel ist. Davon hängen maßgeblich die Synchronisationsaufwände und –lösung ab. Bei einer Delegation der Sicherheitsfragen an eine weitere Instanz können diese Nachteile vermieden werden. Die Daten sind immer aktuell, ein Abgleich / eine Synchronisation ist nicht notwendig. Das Fachverfahren muss die entsprechenden Schnittstellen der Sicherheitsschnittstelle (genannt: SecurityProvider) anbieten. Der Aufwand für die Bereitstellung der Schnittstelle ist dabei wesentlich geringer, als die Umsetzung einer Synchronisationslösung. Des weiteren wird ein Grundsatz des Auftraggebers, einen einheitlichen Identitätsspeicher anzustreben maßgeblich unterstützt. Aus diesen Gründen wird Variante 1 gewählt.		
Auswirkungen	Die Details zur Ausgestaltung dieser Sicherheitslösung sind in Kapitel 6.4 zusammengefasst. Autorisierung auf Dokumentenebene ist durch diese Art der Umsetzung nicht einfach umsetzbar, da das Fachverfahren die Dokumente der eAkte typischerweise nicht kennt bzw. nicht kennen muss. In Zukunft soll SAFE die Speicherung der Identitäten und ggf. Rollen übernehmen sowie die Authentisierung eines Benutzers durchführen. Diese Authentisierung kann auch durch den SecurityProvider entsprechend in eIP umgesetzt und genutzt werden. Wenn ein Fachverfahren keine eigene Sicherheitslösung anbietet, kann ein entspre-		

Themenbereich	Sicherheitsarchitektur
	chender SecurityProvider mit eigener Datenhaltung umgesetzt werden, um eIP nutzen zu können. Die Aufwände sind in diesem Fall gering zu sehen, da keine umfassende Geschäftsverteilung o.ä. zu erwarten ist, da diese sonst bereits im Fachverfahren abgebildet worden wäre.
Abgeleitete Anforderungen	Im Offline-Fall, wenn der SecurityProvider (meist das Fachverfahren) nicht zur Verfügung steht, muss eine alternative Lösung umgesetzt werden. Siehe dazu Kapitel 6.5. Dies müsste aber auch bei Variante 2 bei Ausfall der Serversysteme ebenfalls gelöst werden.
Verbundene Entscheidungen	-

5.17 Architekturentscheidung Kommunikationstechnik für serverseitige Kommunikation zwischen Anwendungen

Themenbereich	Kommunikationsarchitektur		
Architekturentscheidung		ID	5.8
Issue oder Problem	Die clientseitige Interprozesskommunikation ist nicht die bevorzugte Kommunikationstechnik und für Webanwendungen auch nur schwer zu lösen. Somit wird eine bidirektionale serverseitige Kommunikation benötigt, um Daten zwischen eIP und Fachverfahren auszutauschen. Dabei ist eine bidirektionale Kommunikation notwendig, da entweder eIP oder das Fachverfahren der Initiator der Kommunikation sein kann.		
Annahmen			
Motivation	Hohe Qualität, Stabilität und Einheitlichkeit für eine gute Wartbarkeit muss sichergestellt sein; möglichst geringer Integrationsaufwand und Infrastrukturaufwand		
Alternativen	1. Direkte Nutzung spezifischer Schnittstellen 2. Direkte Nutzung von spezifischen Webservices 3. Nutzung von Integration Services unter Nutzung der XML-Nachrichten für die Interprozesskommunikation		
Entscheidung	Es wird Variante 3 umgesetzt.		
Erläuterung der Entscheidung	Die direkte Nutzung spezifischer Schnittstellen für jedes Fachverfahren ist aus Sicht der Wartbarkeit aufgrund der hohen Anzahl von Integrationsprojekten abzulehnen. Bei unterschiedlichen Technologien (Webservices, REST, Socket-Kommunikation uvm.) steigt die Komplexität damit auch noch extrem an und erschwert die Wartung weiterhin. Bei Nutzung einheitlicher aber für das Fachverfahren spezifischer Webservices ist die Technologie festgelegt, aber nicht die konkrete fachliche Schnittstelle bzw. müsste zusätzlich in jedem Fachverfahren umgesetzt werden. Mit der Architekturentscheidung 5.2 wurde für die clientseitige Kommunikation bereits fachliche Schnittstellen auf Basis von XML beschrieben. Die identischen Schnittstellen können auch auf dem Server realisiert werden. Somit ist als technische Kommunikationsschnittstelle nur ein einfacher Service mit einer Methode zur Übergabe der XML-Nachricht und Rückgabe der XML-Antwort notwendig. Aufgrund der grundsätzlichen Bestrebungen, Webservices in der Justiz zu verwenden, wird dies als Webservice umge-		

Themenbereich	Kommunikationsarchitektur
	setzt. Für jede fachliche Domäne wird dabei auf eIP-Seite ein eigener Service bereitgestellt (z.B. DigitalFileIntegrationService). Für den Aufruf der fachverfahrensseitig umgesetzten Services wird ein Proxy auf eIP-Serverseite umgesetzt, damit nicht der eIP-Client mit den Fachverfahrensservices kommunizieren muss.
Auswirkungen	Jedes Fachverfahren, welches serverseitige Kommunikation nutzen möchte, muss seinerseits einen IntegrationService nach der Schnittstellenvorgabe von eIP umsetzen.
Abgeleitete Anforderungen	-
Verbundene Entscheidungen	5.2

5.18 Architekturentscheidung Datenformat für Cachespeicher

Themenbereich	Komponentendesign		
Architekturentscheidung		ID	4.5
Issue oder Problem	Für die Ablage der lokalen Cachedateien ist in Datenformat notwendig, welches verschiedene Versionen des eIP-Clients unterstützen kann und für unerlaubten Zugriffen geschützt ist.		
Annahmen			
Motivation	Versionsstabilität und Sicherheit		
Alternativen	1. Nutzung von XML (ggf. mit Verschlüsselung) 2. Nutzung von Protocol Buffers 3. Nutzung von JSON (ggf. mit Verschlüsselung) 4. Nutzung von serialisierten Java Objekten 5. Properties Dateien		
Entscheidung	Es wird Variante 1 umgesetzt		
Erläuterung der Entscheidung	Protocol Buffers (Variante 2) stellt eine hochperformante Technologie dar, bei der die Daten kodiert transportieren werden können. Die Technologie ist wenig verbreitet und es sind derzeit zusätzliche Bibliotheken notwendig, um Protocol Buffers ansprechen zu können. Neben typisierten Daten ist eine Prüfung der Inhalte auf Korrektheit sowohl für den Empfänger als auch Absender möglich. Aufgrund der geringen Verbreitung und der zusätzlichen Anforderungen wird diese jedoch nicht eingesetzt. JSON stellt eine sehr einfach zu implementierende Technologie auf String-Basis dar, welche einfach Schlüssel-Wert-Paare ohne Typisierung und semantischer Beziehung abbildet. Somit können keine Standardprüfungen für Nachrichten erfolgen und Formatumwandlungen (z.B. Text → Zahl) müssen manuell je nach Attribut durchgeführt werden. Aufgrund dieser Nachteile wird diese Variante nicht gewählt. Serialisierte Java Objekte sind das Standardvorgehen für den Datenaustausch innerhalb		

Themenbereich	Komponentendesign
	von Java Anwendungen. Allerdings sind dabei die Klassen, die für die Serialisierung genutzt wurden, zwingend im Klassenpfad notwendig, was eine Versionierung und damit Versionskompatibilität erschwert. Aus diesen Gründen wird dieser Ansatz nicht mehr weiter verfolgt. Properties-Dateien unterstützen für einen Schlüssel nur einen Wert. Somit müssen Listen o.ä. umständlich in einem eigenen Format ergänzt werden, weshalb dieser Ansatz ebenfalls nicht weiter verfolgt wird. XML stellt eine bekannte Technologie dar, die vor allem durch den Einsatz von Webservices eine große Verbreitung erlangt hat. Es sind typisierte Daten möglich, die per Schemadefinition festgelegt sind und die darüber für Empfänger und Absender eine Prüfung ermöglichen. Die Verarbeitung von XML ist allerdings langsamer als bei den anderen Varianten. Nachdem allerdings die Lese- und Schreibhäufigkeit und die übermittelte Datenmenge eher gering ist, wird dieser Nachteil bewusst in Kauf genommen und diese Variante für die Umsetzung gewählt.
Auswirkungen	Für sicherheitsrelevante Inhalte wird eine zusätzliche Verschlüsselung umgesetzt, um einen unerlaubten Zugriff zu erschweren.
Abgeleitete Anforderungen	Es müssen Infrastrukturmaßnahmen für die gesicherte Ablage der Cachedateien nur für die jeweiligen Benutzer vorgesehen werden.
Verbundene Entscheidungen	-

6 GRUNDARCHITEKTUR

eIP stellt eine 3-Schichten-Architektur dar und ist modular — basierend auf einer service-orientierten Architektur — aufgebaut.

Die Präsentationsschicht wird als Fat-Client auf Basis der Eclipse Rich Client Plattform Version 4 (Eclipse RCP 4.3.1) realisiert. Dabei werden die verschiedenen Anwendungskomponenten als Eclipse RCP Plugins in die Oberfläche integriert. Jede Anwendungskomponente wird als eigenes Plugin repräsentiert, um den modularen Gedanken zu unterstützen.

Die eIP-eigene Logikschicht ist als Java Enterprise Edition (JEE) Module in der Ausprägung als Stateless EJB 3.0 umgesetzt. Die öffentlichen Logikmodule können auch eine Webservice-Schnittstelle anbieten, um fachliche Funktionen für serverseitige Zugriffe zugänglich zu machen. Die Logikschicht der jeweiligen integrierten Anwendungen ist von der jeweiligen Technik abhängig (z.B. Normfall Manager als .NET/COM-Server).

Die eIP-eigene Datenschicht ist in zwei unterschiedlichen Datensenken realisiert. Eine Datenbank hält eIP-eigene Daten (z.B. Aufgaben des Aktenbocks, Konfigurationsparameter). Die elektronischen Akten bestehend aus Ordnern, Dokumenten, Annotationen und Metadaten werden über einen eIP-DigitalFilesService abstrahiert in einer beliebigen Datensenke abgelegt. Die Datensenke wird über einen Adapter angesprochen. Aktuell existiert ein Adapter für den Zugriff auf ein Document Management System (DMS) per CMIS-Schnittstelle sowie ein datenbankbasierter Adapter für Testzwecke.

In eIP integrierte Anwendungen können weitere Datensenken nutzen – dies hängt von der jeweiligen Anwendung ab (z.B. Normfall Manager mit MS SQL Server).

Das folgende Schaubild zeigt die schematische Darstellung:

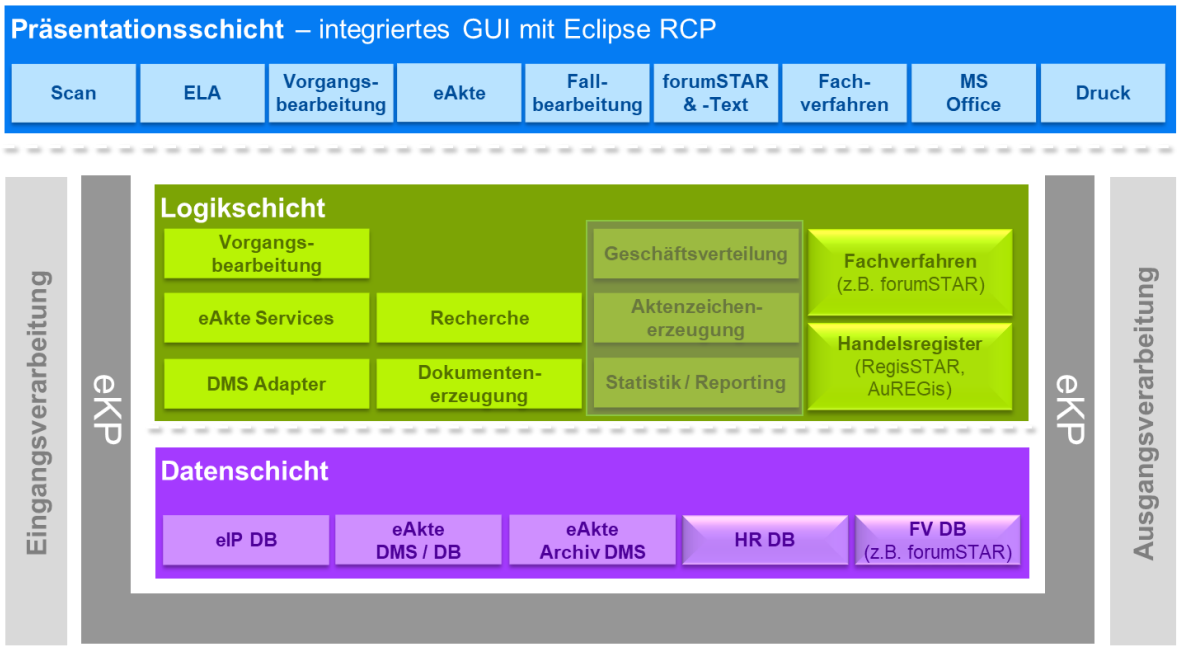

Abbildung 6-2: Schichtenarchitektur eIP

6.1 Architectural Overview Diagram

Das Architectural Overview Diagram zeigt die einzelnen Komponenten und deren Abhängigkeiten von eIP im Detail. Die <use>-Beziehungen sind dabei in fast allen Fällen lose Kopplungen, die per Nachrichten über eine Nachrichtenwarteschlange hergestellt wird. Dieses Komponentendiagramm zeigt vor allem im Bereich des Clients die logische Sicht auf die Komponenten und keine technische Sicht. Die Plugins haben in der Implementierung andere, technische Namen (z.B. Aktenbaum = de.justiz.eip.digitalfile.viewer.ui). Weiterhin existieren verschiedene, weitere technische Plugins, die z.B. für die Paketierung von wiederverwendbaren Bibliotheken genutzt werden. Diese sind in dem Diagramm aus Gründen der Übersichtlichkeit nicht dargestellt.

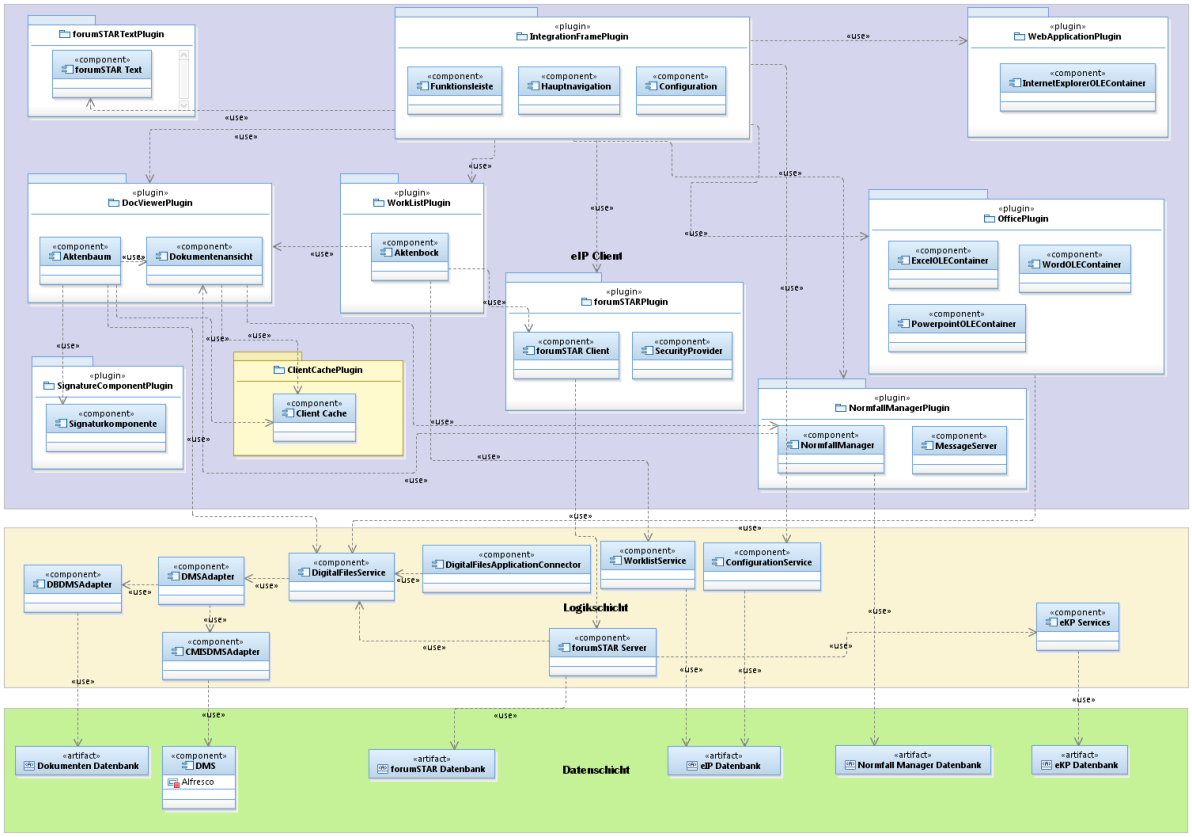

Abbildung 6-3: Architectural Overview Diagram eIP

6.2 Clientarchitektur

Die Clientarchitektur von eIP basiert vollständig auf der Eclipse RCP Grundarchitektur und nutzt die Möglichkeiten sowie das Programmiermodell von Eclipse RCP ab Version 4 aus. Zwischen Eclipse RCP 3 und Eclipse RCP 4 (e4) hat sich das Programmiermodell grundlegend geändert und eine wesentlich offenere Architektur mit größerer Flexibilität etabliert. Erst durch diese Architektur sind die fachlich gewünschten Anforderungen wie eine dynamische Fenstersteuerung umsetzbar geworden.

Jede Anwendungskomponente im eIP Client ist mindestens in einem eigenen Plugin gekapselt. Durch diese Kapselung ist ein Austausch jedes Plugins durch eine alternative Implementierung denkbar und möglich. Deshalb gilt die Vorgabe, dass keine direkten Verbindungen (starre Kopplung) zwischen fachlichen Plugins bestehen dürfen, da damit die Austauschbarkeit nicht gegeben wäre. Direkte Beziehungen bestehen typischerweise nur zu Plugins, die Framework-Klassen o.ä. zur Verfügung stellen, damit diese direkt genutzt werden können. Dies ist allerdings den Basiskomponenten im eIP-Client vorenthalten. Details dazu finden sich

auch in der Architekturentscheidung in Kapitel 5.10, welche die genutzten Kommunikationstechnologien im eIP-Client diskutiert und beschreibt.

Der eIP-Client unterteilt sich in sichtbare (d.h. mit Präsentationsschicht) und nicht-sichtbare (d.h. ohne Präsentationsschicht) Plugins.

Sichtbare Plugins stellen Komponenten dar, die eine visuelle Darstellung im Client beinhalten. Diese Plugins müssen neben einem Application Model zur Definition des Modells vor allem ein oder mehrere Parts (=technischer Begriff für einen Fensterbereich im Eclipse) beinhalten. Diese Parts stellen die visuelle Repräsentation (z.B. Aktenbock, eAkte) dar. Ob diese Parts die Bedienoberfläche direkt mit Hilfe von SWT darstellen oder diese als externe Komponenten einbinden (z.B. über OLE Controls) hängt von der jeweiligen fachlichen Anforderung ab.

Freifliegende, eigenständige Fenster sind in eIP nicht gewünscht, da diese nicht der Integrationsphilosophie von eIP entsprechen. Somit wird für alle eigenständigen Fensterbereiche ein eigener Part im Plugin definiert. Dialoge sind von dieser Vorgabe ausgenommen.

Nicht-sichtbare Plugins kapseln entweder wiederverwendbare Basisbibliotheken (z.B. Frameworks), bieten Mehrwertdienste (z.B. Caching) oder Kommunikationsdienste (z.B. EJB Clientkommunikationsbibliotheken) für andere Komponenten oder stellen eine weitere Strukturierungsmöglichkeit für fachliche Plugins dar (z.B. Unterteilung einer Fachverfahrensintegration in eine visuelle Sicht als Plugin sowie ein oder mehrere Plugins mit Fachlogik ohne visuelle Repräsentation im Client).

Die verschiedenen Plugins des eIP-Clients folgen einer strikten Hierarchie. Dabei können Beziehungen nur zu einer niedrigeren Hierarchieebene bestehen. Eine Beziehung zu einem Plugin einer höheren Ebene ist verboten und würde dem modularen Gedanken von eIP widersprechen. Somit können aber Plugins in unteren Ebenen verschiedene Basisdienste von eIP generell nutzen. Die folgende Darstellung zeigt die Hierarchieebenen und Basiskomponenten von eIP auf:

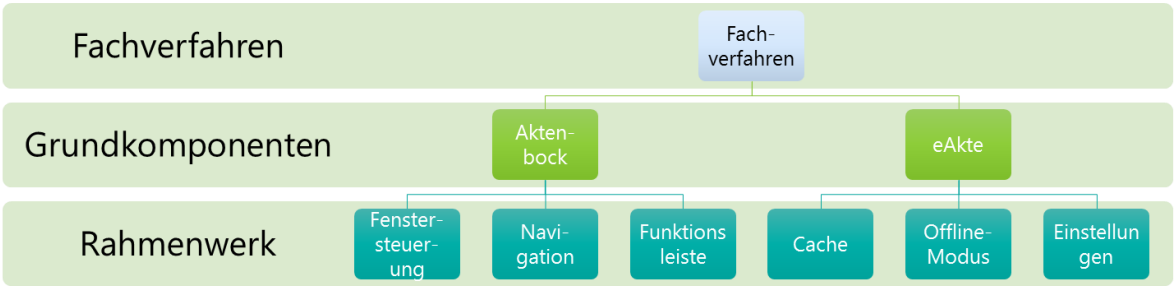

Abbildung 6-4: Darstellung der Plugin-Hierarchie im eIP Client

Hintergrundaktivitäten sind entsprechend der Möglichkeiten von Eclipse RCP über Jobs realisiert.

Für langlaufende Aktionen bietet eIP eine Kennzeichnung für sichtbare Fensterbereiche an (Progress Indicator). Diese kann bei bestehenden Fenstern (Parts) durch eine API aktiviert / deaktiviert werden, so dass dem Benutzer eine laufende Aktion einheitlich im Rahmen angezeigt wird. Bei neuen Fensterbereichen wird dieser sofort und ggf. leer dargestellt, damit die Anzeige der laufenden Aktion erfolgen kann. Die inhaltliche Darstellung erfolgt, sobald die Aktion beendet ist.

Die verschiedenen Komponenten und Plugins können Einstellungen benötigen. Dafür kommt der Eclipse RCP Preference Mechanismus zum Einsatz, welcher innerhalb von eIP für eine Speicherung in einer Datenbank (über den ConfigurationService, siehe 7.1) erweitert wurde. Wenn die Einstellungen für den Endbenutzer veränderbar sein sollen, kann jedes Plugin ein oder mehrere Einstellungsseiten für den Einstellungsdialog (zugänglich über die eIP Funktionsleiste) zur Verfügung stellen. Die Speicherung in der Datenbank erfolgt dann für den Entwickler (bei entsprechend hinterlegten Metadaten) transparent.

Erweiterbare, generelle Schnittstellen werden über Eclipse Extension Points angeboten. Dadurch kann jedes Plugin für bestimmte Bereiche Erweiterungen im eIP-Client anbieten. Dabei existieren u.a. Erweiterungspunkte für Sicherheitsdienste (SecurityProvider), Verfahrensdaten (LawsuitProvider), Suchen (SearchProvider) und Aufgaben (WorklistEntryProvider). eIP identifiziert beim Start bzw. bei der ersten Verwendung mögliche Umsetzungen von Extension Points in den Plugins, ruft die durch eIP definierte Methoden auf und verarbeitet die Ergebnisse. Wenn beispielsweise ein Plugin weitere Aufgaben für den Aktenbock liefern möchte, kann es einen oder mehrere WorklistEntryProvider definieren, welche dann beim Aktualisieren des Aktenbocks gerufen werden und die zusätzlichen Aufgaben im Aktenbock visualisieren. Bestimmte Erweiterungspunkte müssen zwingend vorhanden sein, damit eIP funktionsfähig ist (z.B. SecurityProvider), andere sind optional zu sehen. Details dazu sind im Integrationsleitfaden von eIP zu finden.

Die Anbindung der Fachverfahren erfolgt dabei grundsätzlich über die Integrationsbibliothek am Client für die Interprozesskommunikation zwischen dem Fachverfahrensprozess und dem eIP Prozess. Am Server werden Integration Webservices genutzt. In beiden Fällen werden Nachrichten im XML-Format für die fachlichen Schnittstellen transportiert.

6.3 Serverarchitektur

In Bezug auf die Serverarchitektur sind in der generellen Einleitung und in den Architekturentscheidungen die wesentlichen Aspekte bereits beschrieben worden.

Die verschiedenen Services (z.B. ConfigurationService, WorklistService, DigitalFilesService, LawsuitContextService) sind als Stateless Enterprise Java Beans ausgeprägt, die nach der EJB

3.0 Spezifikation umgesetzt sind. Es kommen keine produktabhängigen Features zum Einsatz. Die Zugriffe auf die Datenbank erfolgen mittels Java Persistence API (JPA). Die Aufrufe zwischen den Services werden durch Injection (@EJB Annotation) folgend der EJB-Spezifikation umgesetzt. Aus diesen Gründen ist es sinnvoll, dass alle Services auf dem gleichen Application Server deployed werden.

6.3.1 Dokumentenspeicheradapter und Anbindung unterschiedlicher Datensenken

In den Prämissen und Basisanforderungen wurde bereits dokumentiert, dass die Produktunabhängigkeit vor allem im Umfeld des Document Management Systems entscheidend für die Verbreitung von eIP sein wird, da ein einheitliches DMS in einem Entwicklungsverbund nicht vorzufinden sein wird.

Realisiert wird dies durch einen DMS Adapter, der innerhalb des eAkte-Service (DigitalFiles-Service) genutzt wird. Der DigitalFilesService stellt dabei die fachlichen Methoden nach außen zur Verfügung. Die fachlichen Methoden führen zu einem oder mehreren technische Methodenaufrufen gegen den Datenspeicher. Diese Aufrufe werden über den DMS Adapter in die jeweilige technische Schnittstelle des Zielsystems übersetzte und ausgeführt.

Alfresco ist derzeit über CMIS angebunden, weshalb ein entsprechender CMIS-basierter Adapter existiert.

DMS Hersteller, welche die CMIS Spezifikation inkl. der optionalen Erweiterungen unterstützen, können aufbauend auf dieser Umsetzung angebunden werden.

Proprietäre Schnittstellen können in eigenen Adaptern ebenfalls genutzt werden, um eine Anbindung über Produktschnittstellen zu ermöglichen.

Nachdem die unmittelbaren Anforderungen von eIP an den Datenspeicher sehr gering sind, ist auch eine datenbankbasierte Schnittstelle möglich. Eine Umsetzung für Testsysteme existiert. Dabei wurden verschiedene Funktionen wie Versionierung, Locking, Volltextsuche usw. ebenfalls betrachtet und größtenteils gelöst.

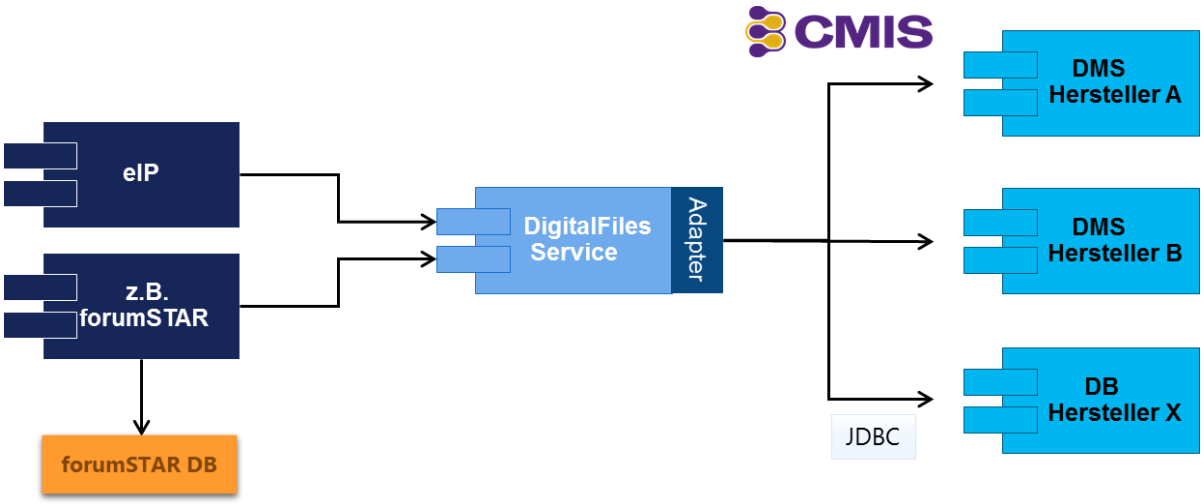

Abbildung 6-5: Anbindung unterschiedlicher DMS Systeme

Innerhalb des DigitalFilesService wird aufgrund verschiedenen Konfigurationsparametern der passende Adapter für das angebundene DMS System über einen Factory-Ansatz ermittelt, instanziiert und genutzt.

6.3.2 Bereitstellung von serverseitigen Schnittstellen für Fachverfahren

eIP bietet, wie im vorigen Kapitel ausgeführt, einen eAkte-Service (DigitalFilesService) als Abstraktionsschicht für den Zugriff auf die elektronische Akte an. Darüber können

- fachliche Manipulationen der Akte (z.B. Akte verbinden, trennen, Paginierung usw.) durchgeführt,
- Dokumente und Ordner mit entsprechenden Eigenschaften angelegt,
- Dokumente abgelegt,
- Dokumente abgefragt und
- eine Suche angestoßen werden.

Diese Schnittstellen sind vor allem auch für Fachverfahren von Interesse, um serverseitige Zugriffe auf die elektronische Akte zu erlangen. Sie können somit serverseitig den entsprechenden Service für Abfragen gegen das eAkte-System oder für Veränderungen der eAkte direkt nutzen.

Die Services werden folgend der Architekturentscheidung 5.8 (siehe Abschnitt 5.17) über einen Integration Webservice angesprochen, der XML-Nachrichten als fachliche Schnittstellen verarbeiten kann. Das Fachverfahren übergibt entsprechend die XML-Nachricht an den Digital File Integration Webservice und erhält die Antwort als XML-Antwort zurück.

Dieser Integration Webservice wird per Annotationen durch den JEE Container als Webservice Schnittstellen bereitgestellt. Diese werden nicht gesondert abgesichert. Die Absicherung muss durch die Infrastruktur sichergestellt werden.

In der aktuellen Umsetzung nutzt forumSTAR direkt die EJB 3.0 Schnittstellen, um eine transaktionssichere Einbindung anzubieten. Diese wird durch den Einsatz des eIP DigitalFile Integration Service abgelöst werden.

Die Details für die serverseitige Ausprägung sind dazu sind in Kapitel 7.6 ausgeführt.

Generelle Details zur Integration sind im Integrationsleitfaden von eIP zu finden.

6.3.3 Protokoll für den EJB Zugriff

Der eIP Client als auch z.B. der forumSTAR Server kommunizieren mit den eIP Services über das t3-Protokoll (RMI-over-t3). Die identische Kommunikationsart wird vom forumSTAR Client zum forumSTAR Server als auch im Rahmen der Eingangslistenapplikation und Fehlerhospital genutzt.

Bei RMI-over-t3 (oder auch RMI-over-IIOP) wird eine performante aber proprietäre Java Kommunikation eingesetzt, welche serialisierte Java Objekte transportiert. Ein Angriff (z.B. Ausspähung der Inhalte) auf eine EJB Kommunikation ist nur mit erheblichem Aufwand möglich, da die serialisierten Java Objekte in eine lesbare Form gebracht werden müssen. Die Bereitstellung eines alternativen Clients (im Sinne eines Austauschs der anfragenden Anwendung) ist nur mit umfangreichem Entwicklerkenntnissen und mit internen Kenntnissen der EJB-Schnittstellen möglich.

Die Kommunikation erfolgt in allen oben genannten Szenarien und Anwendungen mit dem unverschlüsselten t3-Protokoll. Oracle WebLogic bietet auch die Möglichkeit an, eine verschlüsselte Variante des t3-Protokolls namens t3s zu nutzen. Dies ist vergleichbar mit HTTPS und geht möglicherweise entsprechend auf die Gesamtperformance der Anwendung. Konkrete Performanceeinbußen müssten mit Oracle diskutiert werden, wenn dies näher beleuchtet werden soll. Unterschiedliche Verschlüsselungsprotokolle für verschiedene Serviceaufrufe (z.B. Aufgaben unverschlüsselt, Aktenmetadaten verschlüsselt) wären möglicherweise umsetzbar, müssten aber konkret erprobt und geklärt werden.

6.4 Sicherheitsarchitektur

Grundsätzlich wird das Fachverfahren als führend bei der Rechtevergabe für den Zugriff auf ein Verfahren gesehen. Dies bedeutet, dass das Fachverfahren definiert und bestimmt, welche Personen ein Verfahren öffnen bzw. ändern dürfen. Die entsprechende Administration obliegt dem Fachverfahren.

Es wurden verschiedene offene Punkte in dem Bereich der Sicherheitsarchitektur identifiziert sowie weiterführende Gedanken zur Sicherheitsarchitektur festgehalten. Diese sind in Kapitel 14.1 zusammengefasst und sind derzeit noch nicht in die Gesamtarchitektur eingeflossen. Implementierungstechnische Details sind nicht im Architekturkonzept enthalten.

6.4.1 Delegation der sicherheitsbezogene Abfragen (SecurityProvider)

eIP fordert eine Anmeldung sowie weitere Schnittstellen für sicherheitsbezogene Fragestellungen. Die verschiedenen notwendigen Funktionen sind in einer Schnittstelle als SecurityProvider zusammengefasst. Jedes Fachverfahren muss einen SecurityProvider bereitstellen.

Sämtliche sicherheitsbezogene Abfragen erfolgen gegen einen SecurityProvider. eIP kennt dabei den Bezug zwischen Verfahren und SecurityProvider, um bei Verfahren aus unterschiedlichen Fachverfahren an den korrekten SecurityProvider zu delegieren.

Für die Implementierung werden ein einheitliches Interface sowie die Struktur der Benutzer- und Organisationsmetadaten zur Verfügung gestellt.

Die Bereitstellung des SecurityProviders für ein Plugin erfolgt über Eclipse RCP Extension Points. Die reale Umsetzung des SecurityProviders erfolgt bei lose gekoppelten Fachverfahren in der Logik hinter der Integrationsbibliothek (siehe 5.11). Der SecurityProvider ist aus Sicht von eIP clientseitig definiert und wird vom eIP clientseitig genutzt und gerufen. Ob der SecurityProvider die Umsetzung wiederum client- oder serverseitig durchführt, hängt von der jeweiligen Implementierung ab.

6.4.2 Anmeldung an eIP (Authentisierung)

eIP kennt keinen Anmeldemechanismus sondern delegiert diese Funktionalität an das bzw. die integrierte(n) Fachverfahren – genauer an den/die SecurityProvider. Beim Start von eIP wird automatisch die Anmeldefunktion des SecurityProviders gerufen.

Nach einer erfolgreichen Anmeldung muss im eIP Kontext der aktuelle Benutzer mit seinen Metadaten hinterlegt werden. Diese Daten muss der SecurityProvider nach der Anmeldung zur Verfügung stellen.

6.4.3 Unterstützte Anmeldetechniken (Authentisierungsverfahren)

Nachdem eIP keinen eigenen Anmeldemechanismus anbietet, ist dieses von den jeweils integrierten Fachverfahren / SecurityProvider abhängig.

6.4.4 Unterstützung von unterschiedlichen Datensichtbarkeiten (Autorisierung)

eIP bietet grundsätzlich die Unterstützung von drei Datensichtbarkeiten in der elektronischen Akte. Dies stellt somit eine Art Autorisierungsstufe innerhalb von eIP dar. eIP unterscheidet derzeit nicht zwischen schreibenden und lesenden Rechten. Somit kann jeder Benutzer sämtliche verändernden Aktionen in der jeweiligen Stufe durchführen, wenn er berechtigt ist, die Stufe zu sehen.

Die folgenden Stufen werden dabei unterstützt, die sich auf die verschiedenen Komponenten wie z.B. Akte, Volltextsuche, Normfall Manager auswirken.

Stufe	Beschreibung	Inhalt	Abfrage / Regel

Freigegeben	Freigegebene Inhalte sind von allen Benutzern, welche ein Verfahren öffnen können, zu sehen und zu verändern.	• Dokumente in
		  ○ Akte
		  ○ Eingänge
		  ○ Entwürfe
		• freigegebene Annotationen
			Boolean
			SecurityProvider.
			isCurrentUserAllowed
			ToOpenLawsuit(String)

OE-weit	OE-weite Inhalte können von allen Benutzern gesehen und verändert werden, denen der SecurityProvider das Recht zuspricht.	• Dokumente in
		  ○ OE-weite Handakte
		  ○ OE-weite Annotationen
		• freigegebene Normfall Manager Projekte
			Boolean
			SecurityProvider.
			isCurrentUserAllowed
			ToViewOrganizational
			EntityWideContent (String)

Persönlich	Persönliche Inhalte werden nur von einem Benutzer gesehen und können nur von diesem verändert werden.	• Dokumente in
		  ○ Handakte
		  ○ persönliche Annotationen
		• persönliche Normfall Manager Projekte
			Benutzerkennung
			SecurityProvider.
			getCurrentUser().getLoginId()

Tabelle 6-1: Stufen von Datensichtbarkeiten

6.4.5 Änderung der Rechte durch Rollenwechsel

Einige Fachverfahren (z.B. forumSTAR) unterstützen für einen Benutzer unterschiedliche Rollen. Abhängig von der gewählten Rolle hat der Benutzer unterschiedliche Rechte in der Anwendung und kann ggf. unterschiedliche Inhalte sehen. Die Rolle definiert somit die Autorisierung in der Anwendung. Ein Rollenwechsel kommt somit quasi einer Neuanmeldung gleich.

Der Rollenwechsel wird in eIP explizit ermöglicht und durch den SecurityProvider implementiert.

Nachdem ein Rollenwechsel zu anderen Berechtigungen führen kann, erlaubt eIP einen Rollenwechsel nur, wenn alle Fensterbereiche zuvor geschlossen werden.

Nach Rücksprache mit verschiedenen Fachverfahren (forumSTAR, web.sta, MESTA, EUREKA-Fach, DaBaG, SolumSTAR) sind unterschiedliche Rechte je nach Rolle sowie ein expliziter Rollenwechsel derzeit nur in forumSTAR umgesetzt. Alle anderen Fachverfahren kennen auch Rollenkonzepte, allerdings aggregieren diese die Rechte der verschiedenen Rollen, so dass ein expliziter Rollenwechsel nicht notwendig ist.

6.4.6 Abbildung der Rechte in der elektronischen Akte (Autorisierung)

eIP unterstützt das Öffnen einer elektronischen Akte aufgrund von Aufgaben im Aktenbock oder im Kontext eines geöffneten Verfahrens im Fachverfahren. In beiden Fällen wird ein Arbeitskontext in eIP mit den Stammdaten des Verfahrens etabliert. Beim Etablieren des Arbeitskontextes wird implizit durch eIP über den SecurityProvider geprüft, ob das entsprechende Verfahren (basierend auf dem Aktenzeichen, genauer: dem eineindeutigen Aktenkennzeichen LawsuitId) von dem angemeldeten Benutzer geöffnet werden darf. Nur bei positiver Rückmeldung wird der Arbeitskontext etabliert und die Akte geöffnet. Eine serverseitige Überprüfung der Rechte erfolgt derzeit nicht.

Jeder Benutzer hat alle Rechte in der elektronischen Akte, d.h. jede in der eAkte Komponente angebotene Funktion (Ordner anlegen, Dokumente umbenennen, Dokumente zur Akte hinzufügen inkl. Paginierung, Dokumente entfernen inkl. Fehlblatt erzeugen etc.) ist für jeden Benutzer, der die Akten sehen kann, derzeit nutzbar.

Die Manipulation der Akteninhalte in Bezug auf Verbinden/Trennen von Verfahren, Entfernen/Verschieben von Dokumenten der Akte in andere Verfahren inkl. des Einfügens von Fehlseiten ist mit Aktionen im Fachverfahren verbunden. Die entsprechenden Rechte werden somit vom Fachverfahren geprüft.

Dieser Mechanismus wurde aufgrund der hohen Dynamik und impliziten Rechtevergabe in den Fachverfahren umgesetzt. In forumSTAR wird beispielsweise das Recht, ein Verfahren zu sehen, anhand der Rolle, dem zugeordneten Recht sowie der Zugehörigkeit zu einer Organisationseinheit und dem Geltungsbereich der Rolle bestimmt (siehe 14.3.3). Die Abbildung der Rechte in der Akte hätte einen hohen Synchronisationsaufwand zur Folge, die bei jeder möglichen Änderung im Rechtesystem (z.B. Anpassung der Rolle, Veränderung in der Organisationsstruktur, Zuordnung des Benutzers zu anderen Bereichen, Sperren eines Verfahrens) sofort erfolgen muss, da das Fachverfahren führend bei der Rechtevergabe ist.

6.4.7 Volltextsuche in der elektronischen Akte

Grundsätzlich müssen bei der Volltextsuche alle Bereiche und Inhalte der elektronischen Akte betrachtet werden. Somit sind hier auch die unterschiedlichen Sichtbarkeiten von Handakte, OE-weite Handakte usw. zu betrachten. Dabei wird die Suche grundsätzlich auf das aktuelle Gericht eingeschränkt, da gerichtsübergreifende Suche in Akten fachlich ausgeschlossen wurde.

eIP unterstützt die Suche in der aktuell geöffneten elektronischen Akte sowie in allen elektronischen Akten.

Für die Volltextsuche kann der Benutzer in der Funktionsleiste einen Suchtext eingeben. In der aktuellen Umsetzung wird dieser Suchtext (über die serverseitige Schnittstelle und Adapter von eIP) direkt dem DMS übergeben. Es erfolgt keine Anpassung / Abstraktion des Suchbegriffs durch eIP bzw. den entsprechenden eIP Services. Somit kann der Benutzer die Möglichkeiten des DMS innerhalb des Suchbegriffs zur Optimierung der Suche verwenden (z.B. Wildcards, Ersetzungszeichen, Verknüpfungen UND/ODER). Für Alfresco existiert dazu eine entsprechende Anleitung auf der Alfresco-Webseite.

Die Suche wird durch eIP aufgrund weiterer Parameter, die der Suche übergeben werden, automatisch in Bezug auf die Sichtbarkeit eingeschränkt. Dabei müssen immer vollständige Parameter übergeben werden. Wildcards werden für die Sichtbarkeitsparameter nicht unterstützt. Dadurch kann eine Suche z.B. innerhalb von allen Verfahren ohne Berücksichtigung der Sichtbarkeit nicht erreicht werden.

Es wird für jede zu durchsuchende Akte ein Tupel aus Verfahrenskennzeichen und OE-weite Sichtbarkeit (als Boolean) übergeben. Zusätzlich wird immer die Benutzerkennung des Benutzers übergeben.

Das Verfahrenskennzeichen ist ein eindeutiges Kennzeichen eines Verfahrens, welches vom Fachverfahren vergeben / geliefert werden muss. Dieses Kennzeichen wird in der Akte zur Identifikation des Verfahrens verwendet. Dies ist aus Sicht von forumSTAR beispielsweise nicht das Aktenzeichen, da dies nicht eindeutig ist. Es wird in forumSTAR eine Kombination aus Gericht und Verfahrensschlüssel genutzt.

Im Fall der Volltextsuche in der aktuellen Akte wird entsprechend nur das Verfahrenskennzeichen und OE-weite Sichtbarkeit (als Boolean) sowie die Benutzerkennung an den eIP Service und auch an das DMS übergeben.

Im Fall der Volltextsuche in allen Akten wird lediglich eine Liste von Verfahrenskennzeichen und OE-weite Sichtbarkeit je Verfahrenskennzeichen (List<String,String>) sowie die Benutzerkennung an den eIP Service übergeben. An das DMS wird allerdings die Liste der Verfahrenskennzeichen nicht übergeben, da sich im Zusammenspiel mit Alfresco gezeigt hat, dass eine Filterung auf DMS Seite zu zeitintensiv ist (z.B. bei Übertragung von 90000 Verfahrens-

kennzeichen). Aus diesem Grund wird die Filterung der Ergebnismenge im eIP Service durchgeführt (genauer: im DMS Adapter). Aufgrund der Beschränkung auf das aktuelle Gericht / Behörde des Benutzers wird dies als beste Lösung angesehen, auch wenn potentiell mehr Daten vom DMS an den eIP Server übertragen werden. Bei der Volltextsuche werden allerdings nur wenige Attribute (Aktenzeichen) übermittelt. Die Einschränkung der Sichtbarkeit wird somit durch die Kombination aus eIP Servermodul und DMS gewährleistet.

Der entsprechend konfigurierte DMS Adapter, der den Zugriff auf das DMS durchführt, überführt diese Anfrage in eine für das entsprechende DMS verständliche Abfrage. Im Fall von CMIS wird eine CMIS Query mit allen Parametern erzeugt. Wenn dabei z.B. das Recht für OE-weite Sichtbarkeit vorliegt, wird die OE-weite Handakte in die Volltextsuche mit einbezogen, ansonsten nicht.

6.4.8 Zugriff auf die Dokumente der elektronischen Akte

Der Zugriff auf die Dokumente in der elektronischen Akte erfolgt per HTTP/HTTPS. Die Verwendung des Protokolls hängt von der Konfiguration von eIP und dem verwendeten DMS ab. Die URL für den Zugriff auf das Dokument ist identisch für jeden Benutzer.

Das folgende Beispiel bei Verwendung von Alfresco als DMS verdeutlicht dies:

- http://9.134.62.150:8080/alfresco/service/cmis/s/workspace:SpacesStore/i/4653edca-cb15-4aca-a70f-3ea1e24f78f2/content.pdf
    - 4653edca-cb15-4aca-a70f-3ea1e24f78f2 - ist die letzte Version des Dokuments
    - content.pdf - Dateiname

Eine Änderung der URL erfolgt bei der Erzeugung einer neuen Version des Dokuments im DMS. Bei eIP wird eine neue Version des Dokuments nur erzeugt, wenn eine Inhaltsänderung durchgeführt wird. Änderungen der Metadaten führen nicht zu einer Änderung der Version. Annotationen stellen eigene Dokumente dar mit eigenen Metadaten.

Es erfolgt keine erweiterte Zugriffsprüfung im DMS. Es erfolgt lediglich eine Absicherung des Aufrufs per Basic Authentication (siehe 6.4.9).

Aus Performancegründen kann der Abruf der Dokumente auch über einen dezentralen Dokumentencache erfolgen (DMS Cache – Eigenentwicklung der Fima it-novum). In dem Fall erfolgt der Zugriff auf das DMS durch den DMS Cache. Der DMS Cache verschlüsselt die Dokumente zusätzlich mit einer symmetrischen Verschlüsselung (AES256), die vom eIP Client entschlüsselt werden.

6.4.9 Absicherung des Zugriffes auf den Webserver von Alfresco

Der Zugriff auf den Webserver von Alfresco ist per Basic Authentication gesichert. Dabei können alternative Anmeldeinformationen wie für den Zugriff auf das CMIS-Interface zum Einsatz kommen. Der Zugriff auf den Webserver ist notwendig, um bei Ausfall des Caching Servers dennoch Dokumente abrufen zu können.

6.4.10 Sicherheit beim Zugriff auf einen zentralen Projektspeicher des Normfall Managers

Der Normfall Manager hat kein eigenes Rollen/Rechtekonzept, sondern regelt dies normalerweise über die Zugriffsrechte auf die zentrale Microsoft SQL Server Datenbank. Damit kann jeder Benutzer mit den entsprechenden DB Rechten alle Projekte im Projektspeicher öffnen.

Der Normfall Manager wird innerhalb von eIP in einem speziellen eIP-Modus gestartet. Dieser Modus reduziert die Oberfläche des Normfall Managers auf die in eIP notwendigen Funktionen und erlaubt kein Öffnen eines Projektes über die Oberfläche mehr. Dieser Modus wird durch einen Eintrag in der Windows Registry gesteuert, welcher im Justiz-Netz nur von einem Administrator geändert werden kann.

eIP übergibt nach dem Start des Normfall Managers das Aktenzeichen des aktuellen Verfahrens. Der Normfall Manager lädt das für dieses Aktenzeichen angelegte Projekt (bzw. legt ein neues Projekt an) aus dem zentralen Projektspeicher (MS SQL Server, Unterstützung für Oracle Datenbank möglich).

Jedes Normfall Manager Projekt wird persönlich für den aktuellen Benutzer angelegt. Somit hat nur der in eIP angemeldete Benutzer Zugriff auf dieses Projekt. Allerdings kann der Benutzer sein Projekt auch freigeben. In diesem Fall können alle Benutzer, welche das Recht für OE-weite Inhalte haben (siehe 6.4.4), dieses Projekt ebenfalls öffnen. Dazu wird dieses Recht als Boolean-Attribut an die OLE-Funktion createOpenProject des Normfall Managers übergeben. Ist sowohl ein persönliches als auch ein freigegebenes Normfall Manager Projekt verfügbar, bekommt der Benutzer eine Auswahlmöglichkeit angeboten. Alle Personen, welche Zugriff auf das Normfall Manager Projekt bekommen, haben einen Vollzugriff (lesend und schreibend).

Der Normfall Manager kann nur aus der Hauptnavigation von eIP gestartet werden. Für den Aufruf eines Projektes muss ein Arbeitskontext in eIP vorhanden sein. Dieser Arbeitskontext wird nur angelegt, wenn der Benutzer die Berechtigung hat, das Verfahren zu öffnen. Damit wird beim Zugriff auf den Normfall Manager implizit die gleiche Berechtigungslogik wie beim Zugriff auf die Akte umgesetzt.

Ein externer Start des Normfall Managers außerhalb von eIP mit Zugriff auf den identischen Projektspeicher sollte systemseitig vom Betrieb unterbunden werden. Durch die lokale Steuerung des eIP-Modus des Normfall Managers ist eine Umgehung im Justiz-Netz sehr unwahrscheinlich.

6.4.11 Zugriff auf die eIP Datenbank

Der Zugriff auf die eIP Datenbank für den Zugriff auf Konfigurationsdaten und eIP Aufgaben für den Aktenbock erfolgt über eine JEE DataSource. Diese ist identisch zu weiteren JEE-basierten Fachverfahren (forumSTAR, eKP) so konfiguriert, dass der Zugriff mit einem technischen Benutzer erfolgt (Oracle User, Eigentümer des Schemas). Diese Benutzer hat alle Rechte (schreibend, löschend und lesend), die für den Umgang mit den Daten und Tabellen notwendig sind.

Dieses Vorgehen ist gängige Praxis und wird auch in eIP genutzt.

Nachdem in der eIP Datenbank nur Konfigurationsdaten und Aufgaben der Benutzer gespeichert werden, ist eine zusätzliche Absicherung der Datenbankzugriffe aus Sicht von IBM nicht notwendig. Es liegen in der Konfigurationsdatenbank auch die technischen Benutzerdaten für die Zugriffe auf Fremdsysteme (z.B. DMS) sowie die Schlüssel zum Entschlüsseln der Dokumente, die je nach Einschätzung des Auftraggebers zusätzlich schützenswert sind und möglicherweise über weitere Schutzmaßnahmen (z.B. Views) gefiltert werden können.

6.4.12 Zugriff auf das Document Management System (DMS)

Der Zugriff auf das DMS wird über den DigitalFilesService und den DMSService mit dem entsprechenden DMS Adapter umgesetzt. Die aktuelle Umsetzung für den Zugriff auf Alfresco implementiert den DMS Adapter auf Basis von CMIS. Zusätzlich erfolgt der Zugriff auf das DMS System vom Client über HTTP/HTTPS, um Dokumente direkt abzurufen, wenn der Dokumentencache nicht erreichbar ist bzw. nicht im Einsatz ist.

Der Zugriff auf die CMIS-Schnittstelle (bzw. die HTTP-Schnittstelle) ist im Alfresco abgesichert, so dass eine Anmeldung mit Benutzerkennung und Passwort notwendig ist – analog zur Anbindung / Anmeldung an eine Datenbank. Die Anmeldedaten liegen in der eIP Datenbank im Konfigurationsbereich.

Folgend dem generellen Zugriffsmuster für Datenbanken (siehe Kapitel 6.4.11) erfolgt der Zugriff auf das DMS mit Hilfe eines technischen Benutzers mit der identischen Begründung wie für den Zugriff auf die Datenbank (Vermeidung der Synchronisation von Benutzerdaten und Pflege/Synchronisation des Rechtesystems).

Die Einbindung eines DMS Produktes wie z.B. Alfresco in eine SAFE Domäne wird als schwierig erachtet, da entweder das Produkt vom Hersteller erweitert werden muss oder entsprechende Erweiterungspunkte im Produkt für alternative Anmeldetechniken notwendig sind.

Es wäre möglich, die eIP Services zusätzlich abzusichern, um einen unerlaubten Zugriff von nicht vertrauenswürdigen Anwendungen zu unterbinden. Dies ist in Kapitel 14.2 ausgeführt.

6.4.13 Funktionen des Aktenbocks mit Sicherheitsbezug

Verschiedene Funktionen im Aktenbock haben einen klaren Bezug zur Sicherheitsarchitektur und gehen über die reine Anzeige von Aufgaben eines Benutzers hinaus. Die grundlegende Entscheidung der Delegation der Sicherheitsfragen an einen SecurityProvider wird damit jedoch nicht unterwandert, sondern weiter gestärkt, da alle sicherheitsrelevanten Entscheidungen an einer Stelle konzentriert werden..

6.4.13.1 Funktionspostfächer

In eIP sind Funktionspostfächer definiert, welche mit Postfächern für Gruppen bzw. mehrere Benutzer vergleichbar sind. Diese sind für eine Zustellung von Aufgaben an eine Gruppe von möglichen Bearbeitern gedacht ohne einen Bearbeiter konkret zu wählen.

In der ordentlichen Gerichtsbarkeit bedeutet dies beispielsweise die Weiterleitung einer Aufgabe von einem Richter an seine Servicestelle ohne direkt einen Bearbeiter zu adressieren. Die Mitarbeiter der Servicestelle können dann die Aufgaben entsprechend abarbeiten.

Sobald eine Aufgabe aus einem Funktionspostfach geöffnet wird, wird diese in den persönlichen Bereich verschoben. Dadurch entfällt die Notwendigkeit der Reservierung von Aufgaben.

Die möglichen Funktionspostfächer für die Auswahl als Adressat einer Aufgabe werden vom SecurityProvider für den aktuell angemeldeten Benutzer geliefert. Dabei werden ein Anzeigename sowie ein technischer Schlüssel geliefert. Der technische Schlüssel wird dabei als Eigentümer der Aufgabe in der Datenbank deklariert.

Jeder Benutzer von eIP hat die Möglichkeit, auf Funktionspostfächer zuzugreifen. Dazu ruft der Aktenbock beim SecurityProvider indirekt die für den aktuellen Benutzer hinterlegten Funktionspostfächer ab und zeigt diese als Unterordner im Aktenbock Eingangsbereich an. Die Funktionspostfächer werden dabei aus den Organisationseinheiten, denen der Benutzer zugeordnet ist, gebildet. Der SecurityProvider liefert die Organisationseinheiten, welche dann vom Aktenbock für die Funktionspostfächer genutzt wird.

6.4.13.2 Adressat einer Aufgabe

Für das Weiterleiten von Aufgaben bzw. die Neuerstellung einer Aufgabe muss der Adressat ausgewählt werden. Diese werden anhand des Aktenzeichens (bzw. genauer: Verfahrenskennzeichen) vom SecurityProvider angefordert. Der SecurityProvider ist somit für die Lieferung der möglichen Weiterleitungsadressen (inkl. Funktionspostfächer) verantwortlich. Dabei sollte der SecurityProvider darauf achten, dass nur die Adressaten geliefert werden, die diese Aufgabe auch potentiell bearbeiten können – also nur Adressaten, die Zugriff auf das Verfahren haben. Beim Öffnen der Aufgabe (genauer beim Etablieren des Arbeitskontextes) wird allerdings immer geprüft, ob der aktuelle Benutzer das Recht dazu besitzt.

Zusätzlich besteht in eIP die Möglichkeit, dass ein berechtigter Benutzer einem unberechtigten Benutzer ein zeitlich limitiertes Zugriffsrecht auf eine Akte erteilt. Dies wird fachlich u.a. für den dienstjüngsten Richter (z.B. am Wochenende) benötigt, der im Fachverfahren keine Berechtigung für ein bestimmtes Verfahren hat, aber dennoch Zugriff auf die Akte benötigt. Dabei muss das Fachverfahren (genauer: der SecurityProvider) eine alternative Schnittstelle für die Adressaten einer Aufgabe anbieten, um alle Nutzer des Gerichts als mögliche Adressaten adressieren zu können. Der berechtigte Benutzer wählt aus dieser Liste explizit einen unberechtigten Benutzer aus und in der Aufgabe wird damit das zeitlich limitierte Recht zum Öffnen hinterlegt.

6.4.14 Vertretungsregelung

eIP bietet die Möglichkeit an, den Aktenbock einer anderen Person im Falle eines Vertretungsfalls zu öffnen. Dafür wird der sogenannte Vertretungsaktenbock als weitere Karteikarte im Aktenbock angezeigt.

Die Auswahl der möglichen Personen zur Vertretung wird wiederum vom SecurityProvider geliefert.

Es erfolgt kein Identitätswechsel. Die Bearbeitung der Aufgaben aus dem Vertreteraktenbock erfolgt weiterhin in der Identität des Vertreters.

Sobald ein Vertretungsaktenbock geöffnet wird, wird dieser Umstand in einer speziellen Tabelle in der eIP Datenbank protokolliert. Damit hat der Vertretene die Möglichkeit, die Zugriffe auf seinen Aktenbock zu sehen und ggf. organisatorische Maßnahmen einzuleiten.

Durch dieses Konzept wird kein Sicherheitsrisiko eingegangen, da die Steuerung des Aktenzugriffs weiterhin von der Identität des angemeldeten Benutzers abhängt. Wenn der angemeldete Benutzer keinen Zugriff auf ein Verfahren aus dem Aktenbock des Vertretenen hat, kann er die Aufgabe, Akte und Verfahren nicht öffnen.

6.5 Ausfallsicherheitskonzept und Offline-Fähigkeit

Die grundsätzliche Notwendigkeit und die getroffene Architekturentscheidung zur Ausfallsicherheit sind in 5.15 dokumentiert.

Es ist somit derzeit ein rein lesender Offline-Modus in eIP umgesetzt.

Dabei ist ein dedizierter Offline-Modus von eIP konzipiert und umgesetzt. Dedizierter Offline-Modus bedeutet dabei, dass eine Verfügbarkeit nach den verschiedenen Systemen geprüft und beachtet wird. Wie aus dem operationalen Modell ersichtlich ist (siehe 8.2), erwartet eIP einen Zugriff auf die eIP Datenbank, das Fachverfahren (über den SecurityProvider), das DMS (über die eIP Services – DigitalFilesService) sowie den dezentralen DMS Cache. Der Offline-Modus bezieht sich in eIP auf einzelne Systeme bzw. Komponenten, ohne dass eine Beeinträchtigung der anderen Systeme gegeben ist. Dies bedeutet, dass beispielsweise ein schreibender Zugriff auf die eAkte auch dann möglich ist, wenn das Fachverfahren ausgefallen ist.

6.5.1 Aktenzugriff

Eine weitere Untergliederung in der Offline-Funktionalität stellt den Zugriff auf die notwendige Datenmenge dar – dies soll bedeuten, auf welche Akten hat der Benutzer in einem Offline-Betrieb Zugriff. Unter anderem existieren die folgenden Alternativen für diese Betrachtungen:

- Stufe 1: Explizite, vom Benutzer angeforderte, Offline-Akte

  Der Benutzer legt manuell fest, welche Akten er im Offline-Betrieb zur Verfügung haben möchte.

- Stufe 2: Implizite Offline Akte jemals bereits geöffneter Akten

  Der eIP Client speichert alle jemals geöffnete Akten in einem Cache (Aktenstruktur und Dokumente) und erlaubt im Offline-Zugriff implizit Zugriff genau auf diese Akten.

- Stufe 3: Sämtliche zugriffsberechtigte Akten als Offline-Akte

  Der eIP Client speichert alle zugriffsberechtigte Aktenstrukturen und aktualisiert diese permanent bzw. in festzulegenden Intervallen. Zusätzlich werden alle Dokumente zu jemals geöffneten Akten in einem Cache vorgehalten. Dokumente zu noch nicht geöffneten Akten sind vom DMS Cache abrufbar.

In verschiedenen Diskussionen hat sich herausgestellt, dass Stufe 3 zwingend benötigt wird, um ein sinnvolles Arbeiten zu ermöglichen, vor allem für Entscheider. Dies setzt allerdings voraus, dass Dokumente, welche nicht im Client Cache vorliegen, vom DMS (DMS Cache de-

zentral, zentral oder DMS) angefordert werden können. Somit muss die Netzwerkkommunikation im LAN für den Zugriff auf den gerichtsweiten dezentralen DMS Cache funktionsfähig sein. Wenn gar keine Netzwerkverbindung mehr vorhanden ist, fällt Stufe 3 auf das Niveau von Stufe 2 zurück.

6.5.2 Nutzungsszenarien für den Offline-Betrieb

Im Rahmen der Anforderungsdefinition haben sich für den Offline-Betrieb verschiedene Nutzungsszenarien abgezeichnet, die mit den Einstellungen abdeckbar sein sollten:

- Arbeitsplatzwechsel (vor allem des Entscheiders)

  Die Entscheider (z.B. Richter) nehmen ihre Computer meist nicht mit in den Sitzungssaal, sondern nutzen die im Sitzungssaal verfügbaren Computer. Dies bedeutet, dass sich der Entscheider mit seiner Kennung erneut im Sitzungssaal anmeldet. Bei einem Ausfall der serverseitigen zentralen Systeme soll aber dann auch im Sitzungssaal ein sinnvoller Offline-Betrieb möglich und die Akte im Zugriff sein. Aus diesem Grund kann der Client Objekt Cache im Benutzerprofil abgelegt werden. Bei einem Wechsel des Arbeitsplatzes ist dieser vollständig mit den Aktenstrukturen und Sicherheitsinformationen vorbelegt.

  Der Dokumenten Cache wird lokal im Dateisystem angelegt. Bei Zugriff auf die Akte werden die angefragten Dokumente vom dezentralen DMS Cache aus dem LAN des Gerichtes auf den Sitzungssaalrechner geladen.

- Lokaler Offline-Betrieb ohne Netzwerk

  Der lokale Offline-Betrieb ohne Netzwerk war der Ausgangspunkt der gesamten Diskussion zum Offline-Betrieb von eIP. In dem Fall müssen alle gecachten Informationen (Dokumenten Cache und Objekt Cache) auf lokalen Laufwerken verfügbar sein.

  Damit dieses Szenario weiterhin angeboten werden kann, wird der eIP Client Objekt Cache doppelt redundant in einen primären und sekundären Cache abgelegt. Beide Caches werden vom eIP Client bei Aktualisierungen konsistent gehalten. Der primäre Objekt Cache kann dann im Benutzerprofil abgelegt werden, der sekundäre Objekt Cache auf der lokalen Festplatte. Der Arbeitsplatzwechsel wird durch den primären Objekt Cache im Benutzerprofil unterstützt. Wenn überhaupt kein Netzwerk verfügbar ist (und damit auch kein Zugriff auf das Benutzerprofil), schaltet eIP automatisch auf den sekundären Objekt Cache für den Offline-Betrieb um.

Dieses Vorgehen ist in dem folgenden Schaubild dargestellt.

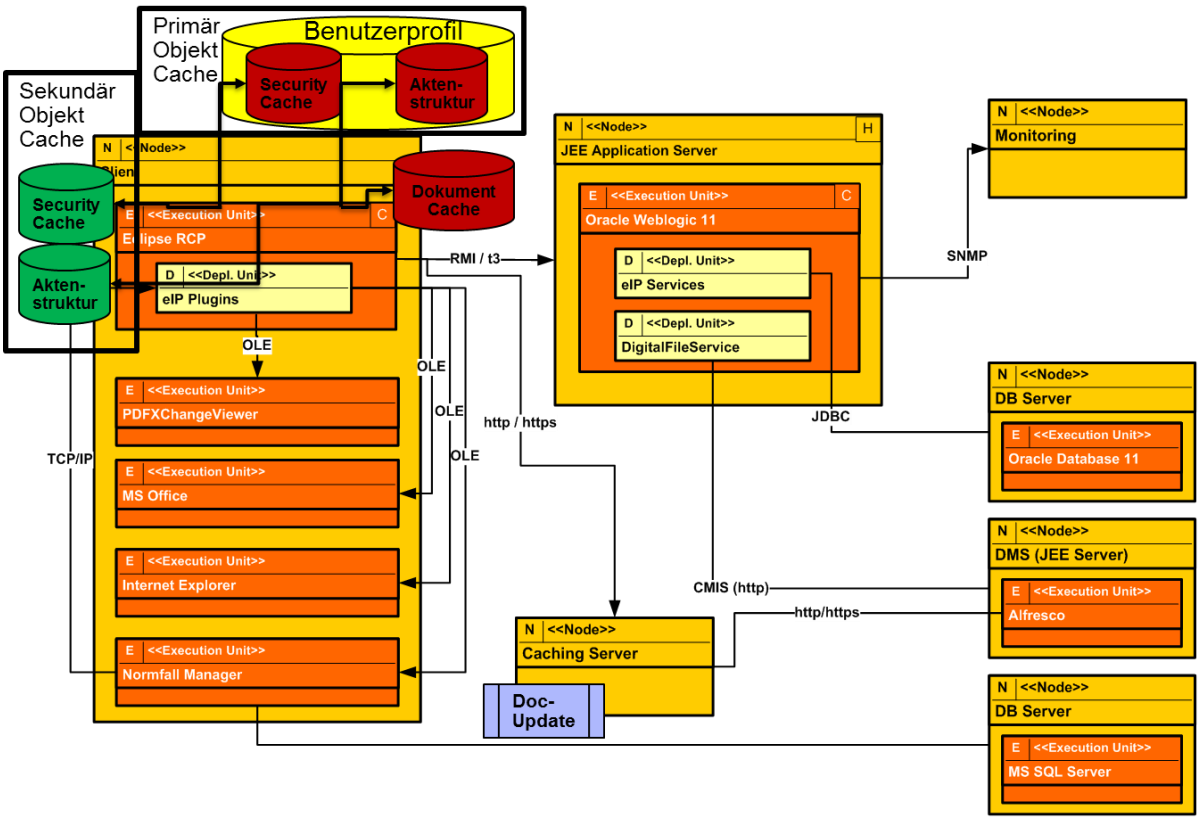

Abbildung 6-6: Cache-Ablageorte von eIP

Weiterhin kann es sein, dass der Client Dokumenten Cache nicht vollständig gefüllt ist, so dass bestimmte Dokumente der Akte nicht angezeigt werden können. In dem Fall wird versucht, die Dokumente aus dem dezentralen DMS Cache zu laden, wenn der Server dem eIP Client bekannt und erreichbar ist. Die Konfiguration für den dezentralen DMS Cache liegt in Systemumgebungsvariablen von eIP (Java Umgebungsvariablen). Wenn der dezentrale DMS Cache nicht erreicht ist, erfolgt ein direkter Zugriff auf das DMS zur Abfrage der Dokumente direkt vom DMS. Details sind in Kapitel 8.2 beschrieben.

6.5.3 Vorgelagerte Betrachtung über Ablagestrategien am Client

eIP delegiert sämtliche Sicherheitsfragen und Anmeldevorgänge an einen SecurityProvider (meist durch das Fachverfahren umgesetzt). Dabei ist derzeit die Anmeldestrategie (z. B. SSO) nicht festgelegt. Bei forumSTAR ist beispielsweise eine Anmeldung per Eingabe der Benutzerkennung und Passwort ebenfalls möglich. Dies bedeutet, dass erst nach der Anmeldung feststeht, als welcher Benutzer sich der Anwender angemeldet hat. Unterschiedliche Benutzer können aber auch unterschiedliche Rechte in der Anwendung haben.

Zusätzlich kann in forumSTAR ein Benutzer mehrere Rollen haben. Je nach Rolle kann er unterschiedliche Rechte in forumSTAR haben. Bei der Anmeldung ist eine Standardrolle aktiv.

Das Caching kann nur die Aktenstrukturen cachen, für die der Benutzer die Rechte hat. Somit bezieht sich das Caching der Sicherheitsinformationen, der Aktenstrukturen und Dokumente, welches beim Start von eIP als Hintergrundjob initiiert wird, immer auf die Rolle, welche als Standardrolle definiert ist. Die entsprechenden Objekte der anderen Rolle können dabei nicht geladen werden. Dies ist ein forumSTAR-spezifisches Verhalten, da kein bisher bekanntes Fachverfahren (SolumSTAR, MESTA, GO§A, EUREKA-Fach, web.sta) einen Rollenwechsel anbietet. In diesen Fachverfahren führen mehrere Rollen zu kumulierten Rechten.

Ein Wechsel des Benutzers oder der Rolle kann sich unmittelbar auf den Cache auswirken bzw. in diesem widerspiegeln. Wenn dies nicht der Fall ist, kann dies zu unerwünschten Nebeneffekten bei der Benutzung von eIP führen.

Aus diesem Grund wird für jeden Benutzer / Rolle ein eigener Cache aufgebaut. Es werden keine Cacheinformationen (weder Objekte, Aktenstrukturen noch Dokumente) zwischen verschiedenen Benutzern oder Rollen wiederverwendet. Damit ist dann auch ein Aufräumen des Caches (z. B. Löschen von Dokumenten) gefahrlos möglich.

Eclipse speichert bestimmte Artefakte auch im Workspace-Verzeichnis ab – dieses kann aber nicht dynamisch zur Anmeldeinformation erfolgen, da zu dem Zeitpunkt Eclipse RCP bereits läuft. Unterschiedliche Konfigurationseinstellungen der Benutzer bei gleichem Workspace sind somit nicht möglich.

Die Standardverzeichnisse für die unterschiedlichen Cache-Ablageorte werden eIP beim Start als Parameter übergeben. Diese Standardverzeichnisse werden von eIP um den Benutzer und die Rolle ergänzt, so dass erst darunter die zu cachenden Informationen zur Ablage kommen.

- z. B. `c:/eip/documentcache` → `c:/eip/documentcache/user1_rolle1/`

Es liegt in der Verantwortung des Auftraggebers bzw. der Infrastruktur (z. B. persönlich verschlüsselte Verzeichnisse o. ä.), dass auf das gewählte Standardverzeichnis nur der berechtigte Benutzer Zugriff hat.

Die Cachingverzeichnisse für Dokumentencache, Aktenstrukturcache und Object Cache können weiterhin unabhängig voneinander konfiguriert werden, damit diese auf unterschiedlichen Bereichen (z. B. Benutzerprofil und lokale Festplatte) abgelegt werden können.

Damit das Löschen von nicht mehr benötigten Inhalten umgesetzt werden kann, ist zwingend darauf zu achten, dass der Dokumentencache getrennt pro Windows-Benutzer abgelegt wird.

Das Laden des Caches kann erst nach dem Anmeldevorgang durchgeführt werden. Generische Konfigurationsparameter liegen nicht im benutzerabhängigen Verzeichnis liegen.

In forumSTAR (und auch in eIP) ist ein Rollenwechsel im laufenden Client möglich. In dem Fall werden alle Fenster geschlossen. Aufgrund der geänderten Ablagestruktur, die abhängig

von der Rolle ist, wird der in den Speicher geladene Cache entladen, die Ablageverzeichnisse im laufenden eIP-Client angepasst und der Cache wie bei einem Neustart geladen. Dies führt damit automatisch auch zur Ausführung des Hintergrundcachings für diese Benutzer/Rolle-Kombination.

Die Konsequenz ist hierbei, dass nur der Cache aktuell gehalten wird, bei dem der Benutzer angemeldet ist. Andere Caches können entsprechend veraltet sein. Dies lässt sich in der aktuellen Umsetzung aber nicht anders gestalten.

Für den Start im Offline-Fall ergibt sich daraus folgende Konsequenz:

Wenn der SecurityProvider beim Start von eIP nicht erreichbar ist, wird dem Benutzer eine neue Maske von eIP angezeigt, in der er den zu nutzenden Cache (auf Basis des persönlichen Standardverzeichnisses) wählen kann. Der Auswahldialog wird nur angezeigt, wenn mehrere Cachingverzeichnisse unterhalb des Standard-Cachingverzeichnisses existieren. Wenn nur genau ein Cachingverzeichnis existiert, startet eIP mit diesem Verzeichnis.

6.5.4 Lokaler Dokumentencache

Im lokalen Dokumentencache werden die Dokumente unverschlüsselt abgelegt, da sie sonst nicht von einem PDF-Viewer / -Editor zur Anzeige gebracht werden können. Neben den eigentlichen Dokumenten liegen weiterhin auch die Annotationsdateien im Dokumentencache.

Für die Umsetzung wird folgendes Speicherverhalten im DMS vorausgesetzt: Jede Speicherung eines Dokuments im DMS führt zur Ablage einer neuen Version. Bestehende Versionen können nie aktualisiert werden. Dadurch muss der Dokumentencache nicht auf geänderte Inhalte eines Dokuments reagieren, da Änderungen für den Dokumentencache neue Dokumente darstellen.

6.5.4.1 Ladestrategie

Ein Dokument wird nur in den Speicher geladen, wenn es angezeigt (durch den PDF-Viewer / -Editor) oder manipuliert (z. B. Trennen, Stempelfunktion, Signatur) werden soll. Es wird keine Vorladefunktion für diese Art von Aktionen umgesetzt, so dass kein Speicher frühzeitig beansprucht wird.

6.5.4.2 Aktualisierungsstrategie

Grundsätzlich wird ein noch nicht im Dokumentencache vorhandenes Dokument vom (dezentralen) DMS-Cache geladen und im Cache abgelegt. Wenn kein DMS-Cache vorhanden ist oder das Laden des Dokuments vom DMS-Cache zu einem Fehler führt (z. B. Zugriffsfeh-

ler, Entschlüsselungsfehler, Transportfehler o. ä.), wird das Dokument weiterhin direkt vom DMS geladen.

Es ist eine vierstufige Ladestrategie für Dokumente umgesetzt.

- Wenn ein Dokument angezeigt werden soll und noch nicht im Dokumentencache vorhanden ist, wird es entsprechend in den Cache geladen („lazy loading").

- Wenn eine Akte im eIP-Aktenviewer angezeigt wird, werden alle noch nicht im Cache vorhandene Dokumente in einem Hintergrundjob in den Dokumentencache geladen. Dieses Verhalten ist durch einen Konfigurationsschalter zu- / abschaltbar, da dieses nicht von allen Fachverfahren gleichermaßen gewünscht ist.

- Wenn die Akte aufgrund einer Verschiebe- / Kopieren-Aktion aktualisiert wird, werden auch alle neuen Dokumente in den Dokumentencache geladen und die nicht mehr in der Akte existierenden Dokumente (inkl. dazugehöriger Annotationen und Signaturen) gelöscht. Der Vorgang für die Aktualisierung des Aktenbaums kennt die alte und neue Struktur, weshalb dieser gelöschte Dokumente (nur aktuellste Version sowie dazugehörige Annotationen und Signaturen) sowie neue Dokumente identifizieren kann. Ein Dokument inkl. der dazugehörigen Annotationen und Signaturen wird erst dann physisch gelöscht, wenn es aus dem Papierkorb gelöscht wird oder es erst gar nicht im Papierkorb landet. Der Papierkorb gehört zur Funktionalität der eAkte und ist konkret einer Verfahrensakte zugeordnet, für die ein entsprechender Zugriffsschutz umgesetzt ist. Es ist nicht der Windows Papierkorb gemeint.

- In einem Hintergrundjob werden alle noch nicht im Cache vorhandenen Dokumente von allen gecachten Aktenstrukturen in den Dokumentencache geladen. Dies widerspricht allerdings dem „lazy loading"-Feature des Aktenstrukturcaches, da für das Laden der Dokumente die Aktenstruktur (mindestens einmalig) in den Speicher geladen (mit den Nachteilen der aktuellen Umsetzung), dort allerdings nicht langfristig gehalten werden muss. Dieses Verhalten ist durch einen Konfigurationsschalter zu- / abschaltbar, da dieses nicht von allen Fachverfahren und allen Benutzen gleichermaßen gewünscht bzw. benötigt wird.

Ein Löschen von alten, nicht mehr benötigten Dokumenten ist nicht einfach umzusetzen, wenn der eIP-Client von unterschiedlichen Benutzern genutzt wird und der Dokumentencache geteilt wird. Deshalb und wegen der möglichen unterschiedlichen Zugriffsrechte ist eine Ablage pro Benutzer auf dem eIP-Client zwingend notwendig.

Gelöschte Dokumente werden bei Öffnen einer Akte aus dem Dokumentencache gelöscht bzw. wenn diese durch eine Benutzeraktion an diesem Client gelöscht werden. Eine Hintergrundlöschung aller nicht mehr benötigter Dokumente ist abhängig von der Aktualisierungsstrategie des Aktenstrukturcaches, da nur im Zuge der Aktualisierung der Akte eine Löschung von Dokumenten erkannt werden kann.

Beim Start von eIP werden die aktuellen Aktenstrukturen, für die der aktuelle Benutzer laut SecurityProvider keine Rechte hat, aus dem lokalen Aktenstrukturcache gelöscht. Dies führt auch zu einer Bereinigung des Dokument Caches und damit zum Löschen der Dokumente dieser Akte(n).

Vorversionen von Dokumenten werden beim Öffnen der Akte gelöscht, wenn deren Ansichtsversion nicht mehr in der Akte vorhanden ist. Ein Verschieben von Dokumenten in einen anderen Bereich (z. B. von „Entwürfe“ nach „Hauptakte“) führt nicht zur Löschung der Vorversionen, da die Dokumenten-Id identisch bleibt.

Das Löschen der ältesten Dokumente im Cache aufgrund eines Schwellwerts wird umgesetzt. Für die Bestimmung des ältesten Dokuments ist keine eigene LRU[8]-Map o. ä. notwendig – es kann auf das Änderungsdatum des Dokuments auf der Festplatte dafür zurückgegriffen werden.

Es ist möglich, dass sich Dokumente im Dokumentencache befinden, die aufgrund eines zeitlich begrenzten Zugriffs auf eine Akte gespeichert wurden (EIP-29 – Dienstjüngster Richter). Nach Ablauf der Zugriffsberechtigung werden diese Dokumente aus dem Dokumentencache gelöscht, wenn die Aktenstruktur aus dem Aktenstrukturcache entfernt wird. Das Entfernen einer Aktenstruktur aus dem Aktenstrukturcache impliziert grundsätzlich die Löschung aller Dokumente dieser Aktenstruktur.

6.5.4.3 Ablagestrategie

Die Dokumente im Dokumentencache werden mit der eindeutigen, technischen ID aus dem DMS im Dokumentencache abgelegt. Diese technische ID beinhaltet (je nach DMS) auch die Version des Dokuments. Damit können mehrere Versionen eines Dokuments parallel im Dokumentencache existieren. Es müssen hier grundsätzlich eine ID und eine Versionsinformation kombiniert werden, damit dies bei allen DMS Systemen gleichermaßen funktioniert. Bei einem Wechsel des DMS-Produktes (z. B. von Alfresco zu OpenText) kann der Dokumentencache nicht beibehalten werden, da dies geänderte IDs zur Folge hätte.

Dadurch kann direkt aus den Metadaten des Dokuments auf den Dateinamen im Dateisystem geschlossen werden. Eine erweiterte Mappingdatei entfällt. Zusätzlich werden der in den Metadaten des Dokuments gespeicherte Dateiname mit Version ergänzt und als Speichername verwendet werden. Annotationsdateien stellen im DMS unabhängige Dokumente dar, weshalb diese eine eigene technische ID haben und damit ebenfalls mit diesem Namensschema abgebildet werden können.

[8] Least Recently Used

Im Dokumentencache wird pro Akte ein Unterverzeichnis angelegt. Das Unterverzeichnis erhält die technische Aktenkennung (DigitalFileId bzw. bisher LawsuitId) als Ordnername. Dadurch wird die Zuordnung der Dateien zu einer Akte bei einer manuellen Kontrolle erleichtert. Aus rein technischer Sicht ist diese Untergliederung nicht notwendig, erleichtert aber den Umgang mit möglichen Namenskonflikten.

Das Ablageverzeichnis des Dokumentencaches ist im eIP-Client konfigurierbar.

6.5.4.4 Integritätsprüfung für Dokumente

Beim Transfer der Dokumente aus dem DMS oder aus dem dezentralen DMS-Cache sind potentiell Transferfehler oder Angriffsszenarien möglich. Um defekte oder manipulierte Dokumente frühzeitig zu erkennen, wird nach der Speicherung des Dokuments im Dokumentencache eine Prüfsumme (SHA-256) berechnet und diese mit der Prüfsumme in den Metadaten des Dokuments verglichen. Wenn diese Prüfsumme nicht korrekt ist, wird das Dokument gelöscht und ein Transfer einmalig erneut versucht. Wurde das Dokument aus dem dezentralen DMS-Cache geladen, wird bei dem erneuten Versuch direkt auf das DMS zugegriffen, um ein defektes Dokument im DMS-Cache zu umgehen. Eine Anpassung / Mitteilung an den DMS-Cache erfolgt nicht. Damit lassen sich allerdings nicht alle Angriffsszenarien erkennen, da potentiell auch die Metadaten manipuliert werden könnten. Dies wird aber unter den Gesichtspunkten von Nutzen / Aufwand in Kauf genommen.

Beim Einlagern / Speichern eines neuen Dokuments im DMS wird die Prüfsumme automatisch vom *DigitalFilesService* serverseitig berechnet und in den Metadaten gespeichert. Um Übertragungsfehler vom Client oder vom Einlagerer erkennen zu können, muss dieser die Prüfsumme berechnen. Nachdem dies nicht von jedem Einlagerer zwingend gefordert werden kann, wird dies für den eIP-Client umgesetzt (z. B. beim Import von Dokumenten aus dem Dateisystem) und somit ein optionaler Methodenparameter definiert, welcher beim Einlagern eines Dokuments mit übermittelt werden kann. Wenn die Prüfsumme übermittelt wird, verifiziert der Server die entsprechende Prüfsumme vor Ablage des Dokuments im DMS.

6.5.4.5 Externes Löschen von Dokumenten

Der Dokumentencache liegt auf der lokalen Festplatte des Computers. Auf diesen kann der Benutzer auch manuell über den Windows Explorer zugreifen und beispielsweise Dokumente löschen. Das gleiche könnte auch durch z. B. den Virenscanner passieren.

Wenn der eIP-Client gestartet und eine eAkte geladen ist, wird das Vorhandensein bestimmter Dokumente im Dokumentencache im lokalen Speicher von eIP gehalten. Wenn zu diesem Zeitpunkt das Dokument gelöscht wird, würde dies bei einem Zugriff auf das Dokument

durch eine Fehlermeldung des PDF-Viewers auffallen, da eIP vor Übergabe des Dateinamens an den Viewer keine Existenzprüfung der Datei durchführt.

Die Existenzprüfung ist eine relativ teure Operation, die IBM aus diesen Gründen nicht vor den Aufruf des Viewers bei jedem Dokument umgesetzt hat.

Nachdem der Fall für die Produktion sehr unwahrscheinlich ist, wird keine Erweiterung der aktuellen Umsetzung in diesem Bereich vorgesehen. Das Problem lässt sich durch manuelles Löschen der lokalen Aktenstrukturcachedateien für diese Akte und einem Neustart des eIP-Clients lösen.

6.5.4.6 Darstellung des Gesamtablaufs

Das folgende Schaubild verdeutlicht die Ablagestrategie, die Zugriffswege sowie die verschiedenen involvierten Komponenten, die im Rahmen des Dokumentencaches genutzt werden. Das Zusammenspiel des dezentralen DMS-Cache mit dem zentralen DMS-Cache wird aus Übersichtlichkeitsgründen in dem Schaubild nicht dargestellt. Es wird auf die entsprechende Dokumentation von it-novum verwiesen.

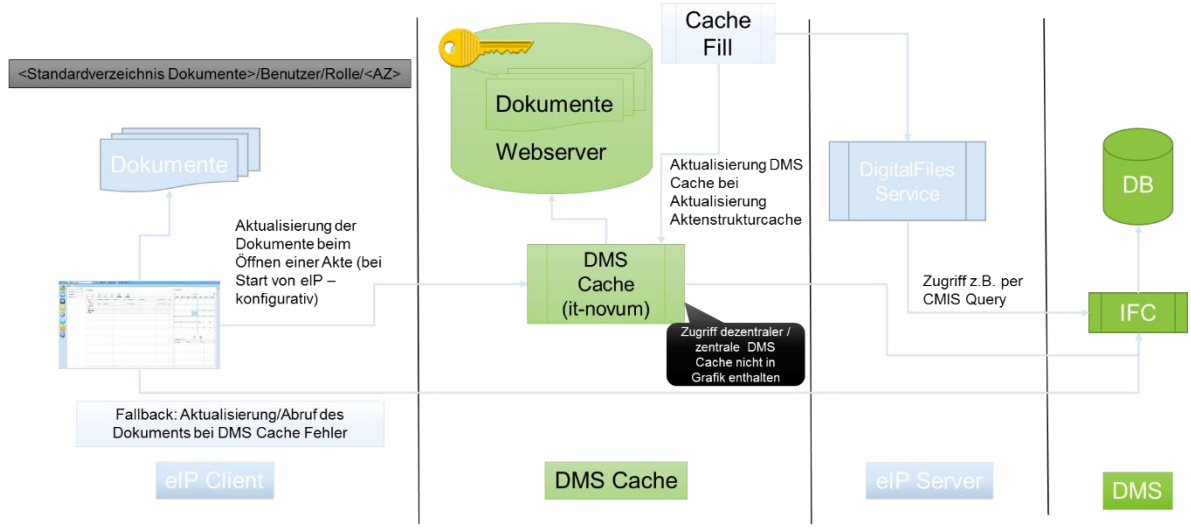

Abbildung 6-7: Gesamtablauf Dokumentencache

6.5.5 Object Cache

Der Object Cache nimmt den Sicherheits-, Aktenbock- und Konfigurationscache auf und wird vom Aktenstrukturcache getrennt in eigenen Verzeichnissen abgelegt.

Im Object Cache werden die folgenden Informationen am Client gespeichert:

- Konfigurationseinstellungen

- Sicherheitsdaten
  - Benutzerdaten wie Name, Rolle, Amtsbezeichnung / Titel, zugeordnete Organisationseinheiten, Gericht
  - Berechtigte Verfahren (Liste von Verfahrens-ID)
  - Zeitlich begrenzte Berechtigungen (dienstjüngster Richter)
- Aktenbockinformationen bestehend aus
  - Ordnern
  - Aktenbockaufgaben
  - Archivierte Aktenbockaufgaben
- Verfahrensinformationen je Verfahren mit
  - Verfahrens-ID
  - Anzeigename des Verfahrens
  - Kurzbeschreibung / Rubrum
  - Zugeordnete Organisationseinheit (z. B. 3. Zivilkammer)
  - Zugeordneter SecurityProvider
- Verfahrenshistorie (Aktenzeichen-ID, Anzeigename des Verfahrens, Kurzbeschreibung / Rubrum)

6.5.5.1 Ladestrategie

Der Object Cache wird beim Start von eIP vollständig in den Speicher geladen und sofort aktualisiert. Im Online-Fall sind die Daten damit vollständig aktuell. Wenn eine der Komponenten (z. B. *ConfigurationService*, SecurityProvider) nicht verfügbar ist und diese damit in den Offline-Modus geht, werden die Daten aus dem Cache verwendet.

Die Daten sind sofort beim Start von eIP notwendig. Weiterhin ist die Anzahl der Dateien und die Größe der einzelnen Cachedateien gering, weshalb das Laden beim Start von eIP keinen Nachteil mit sich bringt.

6.5.5.2 Aktualisierungsstrategie

Die Cacheinhalte werden entweder direkt beim Start (z. B. SecurityProvider), bei periodischen Abrufen (z. B. Aktenbock) oder konkreten Änderungen (z. B. Konfiguration, Neuanlage von Verfahren und damit Aktualisierung des Sicherheitscaches) aktualisiert. Die konkrete Strategie hängt vom jeweiligen Modul ab.

Es werden allerdings Synchronisationspunkte eingebaut, um das Speichern im Rahmen von Hintergrundjobs sicherer vor doppelter Manipulation der Daten zu machen. Außerdem wird beim Beenden von eIP darauf geachtet, dass alle Hintergrundjobs korrekt abgearbeitet wurden, bevor das System beendet wird. Dadurch werden mögliche Inkonsistenzen im Object Cache vermieden. Das Warten auf wichtige Jobs wird dabei im Rahmen eines Wartedialogs beim Schließen von eIP umgesetzt.

Für den Object Cache wird keine Löschung von Cachedateien aufgrund von Schwellwertgrößen umgesetzt, da der Object Cache grundsätzlich vollständig vorhanden sein muss.

6.5.5.3 Ablagestrategie

Der Object Cache ist essentiell und muss bereits beim Start verfügbar sein. Bei einem neuen Rollout muss der Object Cache abwärtskompatibel gelesen werden können, damit bei einem Fehler in der Infrastruktur trotzdem noch der Offline-Modus möglich ist.

Dies erzwingt, dass der Object Cache bei einem Update des eIP-Clients nicht gelöscht werden muss und damit eIP in der Lage sein muss, diesen Cache erneut zu laden.

Dafür bieten sich lesbare Formate an, die nicht z. B. auf serialisierte Java-Objekte angewiesen sind. Auch mit der Java-Objektserialisierung ist eine Abwärtskompatibilität umsetzbar – in dem Fall muss aber in den Serialisierungsmechanismus (z. B. durch Vergabe eigener SerialVersionUIDs) eingegriffen werden.

Für die Speicherung der Daten im Dateisystem sind somit drei Varianten favorisiert, die auch in der Architekturentscheidung 4.5 (siehe 5.18) dokumentiert ist:

- Java Properties Dateien (key=value Listen)
- XML
- JSON

Java-Properties-Dateien scheiden aufgrund der Begrenzung auf einen Wert aus (ohne dass dafür wiederum eine eigene Struktur angelegt werden muss).

JSON ist sehr schnell und einfach zu handhaben, XML entsprechend schwergewichtiger. XML bietet allerdings die beste Möglichkeit, um Strukturen sinnvoll abzubilden.

Aus diesem Grund wird XML als die Ablagestruktur für den Object Cache in Zukunft genutzt.

Für jeden Bereich (Sicherheitsdaten, Verfahrensdaten, Konfigurationsdaten, Aktenbockaufgaben) wird mindestens eine eigene XML-Datei geschrieben.

Die erlaubten Inhalte werden mit Hilfe von XSD für jeden Bereich definiert. Dafür wird ein Versionierungskonzept folgend der IntegrationLibrary umgesetzt, um unterschiedliche Versionen parallel abzubilden.

6.5.5.4 Integritätsprüfung für Object Cache

Im Object Cache liegen auch die Sicherheitsinformationen und damit die Berechtigung des Benutzers für bestimmte Akten. Änderungen am Object Cache können somit potentiell zu einem Zugriff auf eine Akte führen, für die der angemeldete Benutzer eigentlich keine Berechtigung hat. Dafür ist natürlich eine gewisse kriminelle Energie sowie die Kenntnis über den internen Schlüssel einer Akte (z. B. in forumSTAR: fstar_D2410_123456789 mit 123456789 als GPE_KEY_ID des forumSTAR-Verfahrens) notwendig.

Im Grunde muss eIP erkennen, ob eine Datei verändert wurde, nachdem diese vom eIP-Client geschrieben wurde. Die Erkennung einer Veränderung kann mit verschiedenen Mechanismen umgesetzt werden.

Der klassische Fall, welcher auch bei den Dokumenten zum Einsatz kommt, stellt die kryptographische Prüfsummenkontrolle dar. Dabei wird eine Prüfsumme bei Ablage der Datei berechnet und gespeichert und bei erneuter Verwendung der Datei diese erneut berechnet und kontrolliert. Wenn die Speicherung der Prüfsumme auch auf dem eIP-Client erfolgt sowie der Prüfsummenalgorithmus bekannt ist, ist dieses Vorgehen sofort mit entsprechender Kenntnis umgehbar und damit angreifbar.

Eine Verschlüsselung stellt eine erweiterte Hürde dar. In dem Fall werden z. B. die Hashdatei oder die Cachedatei verschlüsselt – nur wenn die Datei mit dem korrekten Schlüssel verschlüsselt wurde, kann diese auch wieder entschlüsselt werden. Nachdem bei der Justiz keine flächendeckende PKI-Infrastruktur (mit persönlichen Signaturkarten, welche auch für die Anmeldung am System genutzt werden) existiert, kann bei der Umsetzung nicht auf diese abgestellt werden. Als Alternative können die Schlüssel entweder per Konfiguration eingebracht werden (z. B. wie bei der Dokumentenverschlüsselung), im Code fest hinterlegt oder dynamisch im Code anhand von Systemparametern berechnet werden. In jedem dieser Fälle ist der Schlüssel im Speicher sichtbar und damit abgreifbar und angreifbar. Das Einbringen des Schlüssels per Konfiguration ist nicht sinnvoll, da dieser Einstiegspunkt für einen potentiellen Angreifer sehr einfach zu identifizieren ist. Sobald der Schlüssel im Code hinterlegt ist, ist

dieser bei Bekanntwerden des Schlüssels nur durch Patch des Clients anpassbar. Den Schlüssel aus Systemparametern zu bestimmen ist sehr elegant, allerdings ist damit eine Nutzung des Cache mit unterschiedlichen Systemen (z. B. Richterarbeitsplatz und Sitzungssaalarbeitsplatz) erschwert möglich, da dann unterschiedliche Systemparameter zum Tragen kommen könnten. Der große Vorteil ist, dass dieser Schlüssel je Benutzer unterschiedlich ist.

Ohne entsprechende Infrastrukturmaßnahmen ist eine vollständige Absicherung der schützenswerten Dateien im Object Cache nicht umsetzbar. Dies ist auch in der Theorie entsprechend belegt (z. B. https://de.wikipedia.org/wiki/Integrit%C3%A4t_(Informationssicherheit), https://en.wikipedia.org/wiki/File_verification).

Somit muss die Hürde entsprechend hochgelegt werden, um eine Manipulation höchst unwahrscheinlich zu machen.

Aus dieser Prämisse wird folgendes Vorgehen umgesetzt:

- Erzeugung eines Schlüssels für die Verschlüsselung der Cachedatei basierend auf Systemparametern im eIP-Client
- Verschlüsselung der Cachedatei mit dem errechneten Schlüssel
- Berechnung einer Prüfsumme für die verschlüsselte Cachedatei und Ablage in einer entsprechenden Zusatzdatei (`<filename>.sha256`)

Beim Laden der Cachedatei wird entsprechend die Prüfsumme kontrolliert und die Cachedatei entschlüsselt. Dieser Schritt könnte auch entfallen – wenn eine verschlüsselte Datei manipuliert wird, scheitert die Entschlüsselung – und ein potentieller Angreifer ist auch in der Lage eine korrekte Prüfsumme anzulegen.

Wenn die Entschlüsselung scheitert, wird die Cachedatei gelöscht und der Cache als ungültigt / ungefüllt betrachtet. Der Benutzer erhält darüber keine aktive Information. Dies wird lediglich im Log protokolliert. Dies führt zum Neuaufbau dieser Cachedatei.

Dieses Verhalten ist nicht für alle Dateien des Object Cache notwendig. Im Object Cache werden derzeit Verfahrensdaten, Sicherheitsdaten, Konfigurationsdaten und Aktenbockaufgaben abgelegt.

Schützenswert sind lediglich die Sicherheitsdaten, sämtliche anderen Cachedateien müssen nicht gesondert geschützt werden, weil dadurch keine erweiterten Rechte ermittelt werden können. Eine Verschlüsselung wird allerdings auch für die Konfigurationsdaten vorgesehen, damit auch Zugriffsdaten z. B. für das DMS darin abgelegt werden können.

6.5.5.5 Darstellung des Gesamtablaufs

Das folgende Schaubild verdeutlicht die Ablagestrategie, die Zugriffswege sowie die verschiedenen involvierten Komponenten, die im Rahmen des Object Cache genutzt werden. Nachdem die verschiedenen Provider von unterschiedlichen Komponenten umgesetzt werden können, ist eine integrationsübergreifende Darstellung mit Nennung aller Kommunikationswege nicht möglich. D. h. dieses Schaubild berücksichtigt nicht die Kommunikationswege innerhalb der Provider-Implementierungen (z. B. innerhalb des forumSTAR-SecurityProviders).

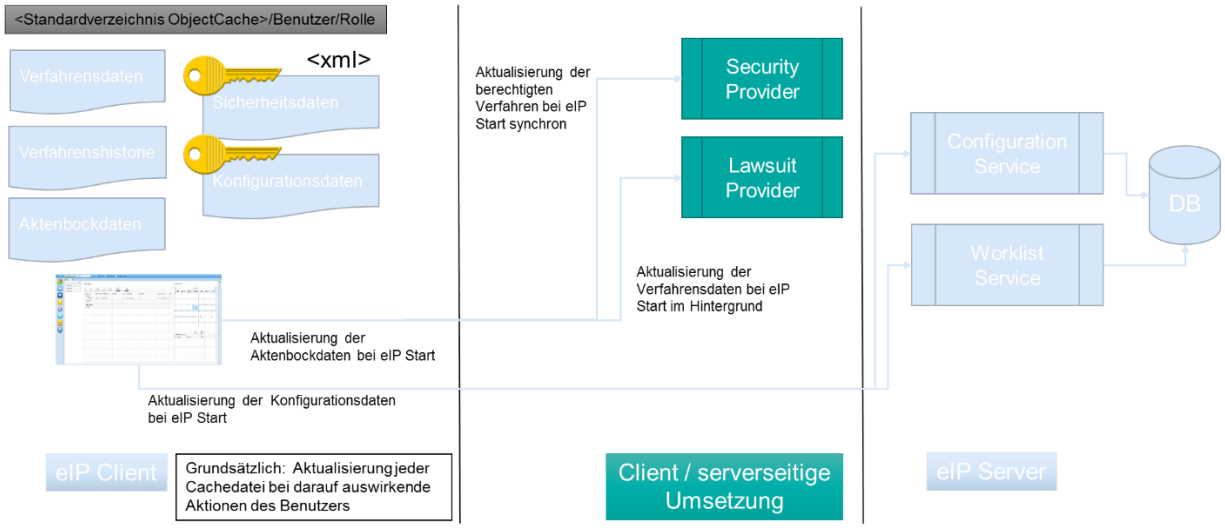

Abbildung 6-8: Gesamtablauf Object Cache

6.5.6 Aktenstrukturcache

Die Umsetzung des Aktenstrukturcaches stellt den größten Umfang im Rahmen des Caches dar. Wenn im weiteren Verlauf von einem Zugriff auf das DMS bzw. die entsprechenden Strukturen gesprochen wird, wird immer davon ausgegangen, dass diese Zugriffe über den eIP-*DigitalFilesService* erfolgen, da dieser die Abstraktionsschicht für den Zugriff auf die Datensenke implementiert. Es erfolgt nie ein direkter Zugriff auf das DMS-System (z. B. Alfresco) aus dem eIP-Client, um Strukturinformationen abzurufen. Dieser direkte Zugriff kommt nur beim Dokumentencache als Fallback zum Abruf eines Dokuments zum Einsatz.

Die Aktenstruktur ist grundsätzlich für alle Benutzer identisch, da auf Dokumentenebene keine gesonderte Rechtevergabe derzeit unterstützt wird. Unterschiede in der Aktenstruktur sind im Bereich der Handakten (OE-weite Handakte sowie persönliche Handakte) zu finden. Die OE-weite Handakte ist für alle Benutzer, die dieser Organisationseinheit angehören, identisch. Die persönliche Handakte ist nur für einen konkreten Benutzer.

Pro Dokument können eine oder mehrere Annotationsdateien existieren, die mit dem Dokument verlinkt sind. Diese liegen aus Sicht des DMS als eigenständige Dokumente im Format XFDF[9] im DMS. Es werden die folgenden Annotationsdateien unterstützt

- Freigegebene Annotationen für alle Benutzer (maximal 1 pro Dokument)
- OE-weite Annotationen für Benutzer der gleichen Organisationseinheit (0..n pro Dokument – pro Organisationseinheit)
- Persönliche Annotationen (0..n pro Dokument – pro Benutzer)
- Normfall Manager-Annotationen für ein freigegebenes Normfall Manager-Projekt – Freigabe erfolgt für gleiche Organisationeinheit (1..n pro Dokument)
- Normfall Manager-Annotationen für ein persönliches Normfall Manager-Projekt eines Benutzers (1..n pro Dokument)

Ein Pre-Caching der Aktenstruktur im Gericht für alle Akten des Gerichts wird empfohlen und ist umgesetzt. Hierfür sind verschiedene Alternativen (unabhängig vom Format) möglich.

Grundsätzlich wird angestrebt, pro Akte eine oder nur sehr wenige Cachedateien im lokalen und dezentralen Aktenstrukturcache zu halten, um die Anzahl der zu verwaltenden Dateien zu minimieren.

Es muss grundsätzlich entschieden werden, ob der dezentrale Aktenstrukturcache eine aktive oder eine passive Komponente darstellt – sprich ob dieser eine Abfrageschnittstelle mit z. B. Übergabeparameter unterstützt (aktiv) oder ob dieser lediglich eine Ablagestruktur im Dateisystem als Netzwerklaufwerk zur Verfügung stellt.

Bei Verwendung eines Netzlaufwerks sind keine dynamischen Anwendungskomponenten (z. B. Tomcat-Server) auf dem dezentralen Aktenstrukturcache notwendig, dies senkt die Infrastrukturanforderungen und Testanforderungen enorm – die Befüllung ist natürlich als eigenständige Anwendung immer notwendig.

Der Nachteil der passiven Lösung ist, dass die gesamte Selektionslogik beim Anforderer (eIP-Client) liegt. Dieser muss in die Lage versetzt werden, die relevanten Daten abzufragen, damit nicht der gesamte Aktenstrukturcache täglich vollständig aktualisiert wird. Eine Selektion neuerer Cachedateien könnte anhand des Änderungsdatums der Dateien erfolgen.

Ein weiterer Nachteil der passiven Lösung ist, dass parallel stattfindende Manipulationen an den Cachedateien durch z. B. die Befüllungsanwendung nicht einfach erkannt werden kön-

[9] XML Forms Data Format

nen. Es müssen hier Sperrmechanismen implementiert werden, damit keine unvollständigen Cacheinhalte aktualisiert werden. Dies lässt sich durch z. B. Signierung / Verschlüsselung der Daten umgehen, da inkonsistente Daten zu ungültigen Signaturen führen würden.

Wenn der dezentrale Aktenstrukturcache als passive Komponente in Form eines Netzwerklaufwerks abgebildet wird, kann jeder Benutzer die Inhalte des Caches in seinem Windows Explorer sehen. Eine potentielle Einsichtnahme in die Aktenstrukturen ist damit möglich. Dies kann durch eine Verschlüsselung der Cachedateien umgangen werden – allerdings mit der gleichen Problematik, wie diese in Kapitel 6.5.4.4 bei der Integritätsprüfung des Object Caches beschrieben wurde (Schlüsselablage – hier ist allerdings ein fester Schlüssel für alle eIP-Clients zwingend notwendig).

Bei Einsatz einer aktiven Anwendungskomponente für den dezentralen Aktenstrukturcache kehren sich Vor- und Nachteile de facto um.

Im weitesten Sinne hängt eine Entscheidung auch vom Datenformat ab und bis auf welcher Granularitätsebene eine Aktualisierung stattfinden kann und soll. Wenn eine Akte vollständig in einer Datei abgelegt vorliegt und z. B. keine dokumentenbasierten Einzelaktualisierungen unterstützt werden sollen, kann dies auch mit der passiven Lösung funktionieren.

Außerdem sind Überlegungen angestellt worden, ob eine direkte Verwendung des dezentralen Aktenstrukturcaches anstelle eines lokalen Caches möglich wäre. Dies ist nur mit einer passiven Lösung umsetzbar – stellt aber weitreichende Herausforderungen, da damit auch die Lösungen für OE-weite / persönliche Inhalte sicherheitstechnisch korrekt abgebildet werden müssen. Aus Sicht der Performance ist eine zusätzliche Entschlüsselung bei unmittelbarem Zugriff nicht wünschenswert. Aus diesen Gründen wird eine direkte Verwendung des dezentralen Aktenstrukturcaches durch den eIP-Client nicht weiter verfolgt.

Für Abfrage der OE-weiten und persönlichen Inhalte könnte auf das DMS zugegriffen werden, um diese dynamisch hinzuzufügen, wenn nur die allgemeinen Aktenstrukturinhalte im dezentralen Aktenstrukturcache liegen – dies würde aber wiederum zu einer großen Last auf der Infrastruktur führen (vergleichbar zu heute, mit Einschränkung auf einzelne Bereiche). Die Konsequenz ist, dass diese Inhalte bereits im dezentralen Aktenstrukturcache vorliegen müssen und ein Zugriff vom eIP-Client auf das DMS aus Gründen der Aktualisierung des Aktenstrukturcaches beim Start nicht umgesetzt wird.

Wenn die OE-weiten und persönlichen Inhalte im dezentralen Aktenstrukturcache vorliegen, sind wiederum drei Optionen denkbar:

- Auslieferung der Aktenstruktur mit allen OE-weiten und persönlichen Inhalten und nachgelagerte Filterung der relevanten Inhalte am eIP-Client
    - Diese Option ist auch mit einem passiven Cache umsetzbar, bedeutet aber, dass ggf. mehr Informationen als notwendig transportiert werden und dass am Client

potentiell Informationen (jedenfalls kurzfristig) vorliegen, die nicht für diesen bestimmt sind.

  - In Bezug auf die OE-weite und persönliche Handakte wird die Einschätzung getroffen, dass die Anzahl der zusätzlichen, unnötigen Informationen sich nicht groß auswirken wird. Persönliche Handakten haben typischerweise Richter und nicht die Servicekräfte. Mehrere OE-weite Handakten sind weiterhin untypisch, da fremde Organisationseinheiten selten zusätzliche Dokumente in eigenen OE-weiten Handakten zur Sichtung / Diskussion ablegen.

- Ablage der OE-weiten und persönlichen Aktenstrukturen getrennt von den restlichen Aktenstrukturinformationen und Zusammenbau der relevanten Inhalte am eIP-Client

  - Diese Option ist auch mit einem passiven Cache umsetzbar, bedeutet aber, dass pro Akte mehrere Dateien zu übermitteln sind (benutzerunabhängige Aktenbestandteile sowie OE-weite / persönliche Handakte).

  - Im eIP-Client müssen die verschiedenen Informationen zusammengefügt werden.

  - In Bezug auf die OE-weite und persönliche Handakte wird die Einschätzung getroffen, dass die Anzahl der zusätzlichen Dateien sich nicht groß auswirken wird. Persönliche Handakten haben typischerweise Richter und nicht die Servicekräfte. Mehrere OE-weite Handakten sind weiterhin untypisch, da fremde Organisationseinheiten selten zusätzliche Dokumente in eigenen OE-weiten Handakten zur Sichtung / Diskussion ablegen.

- Anforderung der Cacheinhalte mit Übergabe der Benutzerinformationen (Benutzerkennung und Organisationseinheit)

  - Für diese Option ist zwingend eine aktive Komponente notwendig, da diese aufgrund der entsprechenden Übergabeparameter die korrekten Strukturen (nur mit passender OE-weiten Handakte und persönlicher Handakte) zusammenstellen und zurückliefern muss. Dies stellt eine erhöhte Anforderung auch an die Verarbeitungsgeschwindigkeit des dezentralen Cachingsystems, da potentiell viele Anfragen aus einem Gericht gleichzeitig bearbeitet werden müssen. Abhängig von der Speicherstruktur und der Speichertechnik (z. B. Datenbanksystem) kann dies elegant oder aufwändig gelöst werden.

  - Durch Übergabe des letzten Aktualisierungszeitpunkts könnte aber mit einem Aufruf alle geänderten Aktenstrukturen auf einmal geliefert werden. Dies erhöht die Infrastrukturanforderungen an den dezentralen Cache, reduziert allerdings dafür die Anzahl an Anfragen. Hierbei kann allerdings keine Einschränkung nach der Berechtigung des Benutzers geprüft werden.

- Wenn zwingend nur die Aktenstrukturen an den eIP-Client übertragen werden dürfen, für die der Benutzer berechtigt ist, müssen entweder die berechtigten Akten-Ids übermittelt werden (unrealistisch für forumSTAR, web.sta aufgrund der hohen Anzahl) oder die Aktenstrukturen einzeln abgefragt werden. Die Einzelabfrage bedeutet in dem Fall, dass pro Akte eine Anfrage an die dezentrale Cachingkomponente gestellt wird und diese für jeden Benutzer diese Anfrage beantworten muss.

- Eine Berechtigungsprüfung, ob der Benutzer überhaupt berechtigt ist, diese Aktenstruktur abzufragen, ist auch mit einer aktiven Komponente nicht sinnvoll umsetzbar, da dazu z. B. das Fachverfahren (bzw. SecurityProvider) für eine Berechtigugsprüfung bei Abfrage der Struktur eingebunden sein müsste oder diese Informationen im dezentralen Cache repliziert vorliegen müssen. Eine Replikation der Sicherheitsinformationen wird aus Sicht der Architektur auf keinen Fall umgesetzt.

Bei diesen ganzen Überlegungen muss beachtet werden, dass das Gesamtsystem auch noch betreibbar sein muss. Wenn die Infrastruktur- und Betriebsanforderungen an das dezentrale Cachingsystem zu hoch werden, ist ein gesicherter Betrieb nur noch schwierig möglich (Überwachung, Fehlersuche usw.). Somit ist eine vergleichsweise einfache Infrastruktur im dezentralen Bereich für die Betriebsstabilität von Vorteil.

In Abstimmung mit dem Auftraggeber wurde unter der Annahme, dass die Sichtbarkeit der verfügbaren Aktenstrukturen im dezentralen Cache sowie eine Filterung auf bzw. Nachladen von OE-weiten / persönlichen Handakten am eIP-Client kein grundsätzliches Sicherheitsrisiko darstellen, der Einsatz einer passiven Lösung entschieden. Damit wird keine aktive Komponente für den dezentralen Aktenstrukturcache benötigt. Dafür muss der dezentrale Aktenstrukturcache verschlüsselt vorliegen sowie per Netzwerklaufwerk zugänglich sein.

Dabei wird die zweite Option umgesetzt, in der die Ablage der OE-weiten und persönlichen Aktenstrukturen getrennt von den restlichen Aktenstrukturinformationen im dezentralen Aktenstrukturcache erfolgt und der eIP-Client den Zusammenbau der relevanten Inhalte durchführt.

Der Aktenstrukturcache wird am eIP-Client in einem weiteren, neuen Cacheverzeichnis (DigitalFileCache) abgelegt – sowohl als Primary als auch Secondary Cache.

6.5.6.1 Ladestrategie

Der lokale Aktenstrukturcache wird nicht vollständig in den Speicher geladen – weder beim Start noch während der normalen Laufzeit.

Die gecachten Informationen einer Akte werden beim Öffnen der Akte vom Dateisystem geladen und im Speicher bereitgestellt (lazy loading). Dies bedeutet, dass das erste Öffnen einer Akte minimal verzögert stattfinden kann.

Ein Laden der gesamten Aktenstrukturen in den Speicher wird bei der Menge an möglichen Akten nicht als sinnvoll erachtet. Das synchrone Laden beim Start der Anwendung entfällt somit und verkürzt die Startzeit des eIP-Clients.

Diese Ladestrategie kann sich ändern, wenn z. B. Akteninformationen bereits im Aktenbock (z. B. Stammblattanzeige) notwendig werden. In dem Fall müssen ggf. die relevanten Akteninformationen im Speicher vorliegen – abhängig von der Umsetzung dieser Funktionen.

6.5.6.2 Aktualisierungsstrategie des dezentralen Aktenstrukturcaches

Der dezentrale Aktenstrukturcache wird durch eine separate Anwendung aktualisiert. Diese wird als eigenständige Java Anwendung umgesetzt und aktualisiert alle Akten der zu konfigurierenden Gerichte. In der Anwendung können dabei ein oder mehrere Gerichte konfiguriert werden, deren Aktenstrukturen geladen werden sollen. Dabei wird keine Berechtigung der Benutzer berücksichtigt – es werden alle Aktenstrukturen des jeweiligen Gerichts geladen / aktualisiert.

Die Abfrage der Aktenstrukturen erfolgt in mehreren parallelen Threads, Die Anzahl der maximalen parallelen Threads ist dabei konfigurierbar.

Für die Abfrage der aktualisierten Aktenstrukturen erfolgt ein Zugriff auf den eIP-*DigitalFileService*. Der letzte Aktualisierungszeitpunkt ist der Aktualisierungsanwendung bekannt. Der eIP-*DigitalFilesService* ermittelt die zu aktualisierenden Aktenstrukturen für ein Gericht und liefert die entsprechenden Akten-IDs zurück. Diese Liste an Akten-IDs wird anschließend verwendet, um die relevanten Akten zu aktualisieren. Dabei wird grundsätzlich die vollständige Aktenstruktur vom DMS über den eIP-*DigitalFilesService* angefordert und keine Teilaktualisierung durchgeführt. Die zu aktualisierenden Aktenstrukturen werden per Query vom DMS angefordert. Diese Query ist bekannt für eine längere Laufzeit und ggf. Belastung des DMS-Systems. Nachdem diese Query aber nur im Rahmen der Hintergrundanwendung zum Aktualisieren des dezentralen Aktenstrukturcaches genutzt wird, ist ein Einfluss auf das Gesamtsystem nicht zu erwarten. Eine alternative Umsetzung unter Einsatz eines Änderungsdatums auf Aktenebene hätte einen viel größeren Einfluss auf die Gesamtperformance des Systems, da dieses bei jeder Änderung zusätzlich gesetzt werden müsste.

Wenn die Anwendung eine Aktenstruktur aktualisiert hat, initiiert diese auch eine Aktualisierung des dezentralen DMS-Caches (Dokumentencache). Dies bedeutet, dass der entsprechende HTTP-Request an den DMS-Cache für die Dokumente der Akte abgesetzt wird und damit ein Nachladen der korrekten Dokumente nach sich zieht.

Es werden keine Metadaten von Vorversionen von Dokumenten geladen. Damit werden auch keine älteren Dokumentenversionen in den DMS-Cache geladen.

Die Aktualisierungsanwendung kann nicht erkennen, ob eine Akte im DMS gelöscht wurde. Somit ist ein Löschen der Aktenstruktur aus dem dezentralen Aktenstrukturcache nicht ohne weiteres möglich. Hier wären folgende Alternativen denkbar:

- Nachfrage beim DMS, ob eine bestimmte Akte noch existiert, die im lokalen Cache vorhanden ist → sehr teure Operation für jede Akte, nicht sinnvoll umsetzbar
- Übermittlung einer Liste von gecachten Akten an das DMS zur Prüfung → technisch machbar, Query-Abhängigkeiten beim DMS denkbar
- Löschen der Aktenstrukturen, die seit längerem nicht aktualisiert wurden → einfache Umsetzung möglich

Es wurde die dritte Option umgesetzt, da der Fall, dass eine Akte gelöscht wird, erst bei der Aussonderung / Archivierung zum Tragen kommt und somit eher in seltenen Fällen stattfindet. Eine umfassende Synchronisation ist nicht notwendig. Die Zeitabstände / Intervalle sowie die Vorhaltefristen können konfiguriert werden.

Die Aktualisierungsanwendung lädt sämtliche Metadaten der OE-weiten und persönlichen Handakten aller Akten in den dezentralen Aktenstrukturcache. Somit liegen für alle Benutzer und OEs die entsprechenden Aktenstrukturbestandteile im Cache.

Eine dynamische Aktualisierung des dezentralen Aktenstrukturcache durch Benutzerzugriffe (vergleichbar mit der dynamischen Aktualisierung des DMS-Caches für Dokumente) wird für den Aktenstrukturcache nicht umgesetzt. Dies bedeutet, dass eine Aktualisierung des dezentralen Aktenstrukturcaches nur durch die entsprechende Aktualisierungsanwendung erfolgt und vom Aufrufintervall abhängt. Ob diese Aktualisierungsanwendung mehrmals täglich ausgeführt werden kann, muss im Detail diskutiert werden (Stichwort: Belastung der DMS-Infrastruktur für die Abfrage der geänderten Strukturen).

Der letzte Aktualisierungszeitpunkt des dezentralen Aktenstrukturcaches wird in einer Datei des dezentralen Aktenstrukturcaches abgelegt. Der Zeitpunkt wird dabei beim Start der Aktualisierungsanwendung festgelegt, aber erst am Ende des Aktualisierungslaufes in die Datei geschrieben. Damit ist sichergestellt, dass bei einem Abbruch der Aktualisierungsanwendung kein falscher Aktualisierungszeitpunkt vermerkt ist. Nach einem Abbruch ist es damit potentiell möglich, dass mehr Aktenstrukturen als unbedingt nötig aktualisiert werden, da der letzte Aktualisierungszeitpunkt nicht pro Akte einzeln verwaltet wird.

Der letzte Aktualisierungszeitpunkt wird dem Server übermittelt, so dass dieser nur die geänderten Akten liefern kann.

6.5.6.3 Aktualisierungsstrategie des lokalen Aktenstrukturcache

Die letzte Änderung des lokalen Aktenstrukturcaches beim eIP-Client ist pro Aktenstruktur bekannt. Diese kann entweder aufgrund der letzten Aktualisierung über den dezentralen Aktenstrukturcache oder aufgrund lokaler Änderungen / Aufrufe durchgeführt worden sein und ist über den Änderungszeitstempel der Cachedatei ableitbar.

Dieser Zeitstempel kann für den Zugriff auf den dezentralen Aktenstrukturcache pro Aktenstruktur genutzt werden. Als Voraussetzung ist hierbei, dass die Systemuhren der Clients und Cacheserver synchron laufen, um Änderungen korrekt erkennen zu können.

Jede zu aktualisierende Aktenstruktur (entweder neu oder bereits im lokalen Aktenstrukturcache vorhanden) wird beim Start des eIP-Clients im Hintergrund vom dezentralen Aktenstrukturcache geladen. Jede Aktenstruktur wird dabei einzeln aktualisiert. In diesem Fall kommt die Berechtigungssteuerung von eIP bereits zum Tragen, weil nur die Aktenstrukturen angefordert werden, für die der aktuelle Benutzer Rechte besitzt. Dabei werden neue als auch zu aktualisierende Aktenstrukturen automatisch geladen.

Aktenstrukturen, für der aktuelle Benutzer laut SecurityProvider keine Rechte hat, werden aus dem lokalen Aktenstrukturcache gelöscht. Dabei wird auch der Dokumentencache bereinigt und alle Dokumente dieser Akte inkl. Annotationen und Signaturen lokal gelöscht. Dieses Verhalten ist konfigurierbar.

Für den lokalen Aktenstrukturcache wird keine schwellwert-basierte Löschung von Strukturen umgesetzt. Dies ist aufgrund des geänderten Ladeverhaltens nicht notwendig.

Es werden Synchronisationspunkte eingebaut, um das Speichern im Rahmen von Hintergrundjobs sicherer vor doppelter Manipulation der Daten zu machen. Außerdem wird beim Beenden von eIP darauf geachtet, dass alle Hintergrundjobs korrekt abgearbeitet wurden, bevor das System beendet wird. Dadurch werden mögliche Inkonsistenzen in dem Aktenstrukturcache vermieden.

Beim Öffnen einer eAkte wird wie bisher immer eine Aktualisierung mittels Zugriff auf das DMS (über den *DigitalFilesService*) initiiert. Dabei werden nur die geänderten Objekte seit der letzten Aktualisierung abgefragt. Jede Änderung in der Akte verändert auch den zugehörigen Ordner (Parent), wodurch geänderte Objekte korrekt in den lokalen Cache einsortiert werden können, da der jeweilige Ordner mitgeliefert wird. Dadurch können auch gelöschte / verschobene Objekte erkannt werden.

6.5.6.4 Ablagestrategie im dezentralen Aktenstrukturcache

Die Ablage erfolgt im Dateisystem. Die Verwendung eines Datenbanksystems für den dezentralen Aktenstrukturcache wird als zu hohe Anforderung an die Infrastruktur im dezentralen Bereich angesehen.

Aufgrund der Entscheidung für eine passive dezentrale Cachingkomponente erfolgt die Ablage einer Aktenstruktur in einer Datei im XML-Format. Darin sind sämtliche Ordner, Dokumente sowie Annotationsdateien mit ihren jeweiligen Metadaten und Verknüpfungen hinterlegt.

Jede Aktenstrukturdatei wird als Einzeldatei abgelegt und verschlüsselt. Es kommt ein symmetrisches Verschlüsselungsverfahren zum Einsatz. Der Schlüssel wird in der eIP-Konfigurationsdatenbank hinterlegt. Sowohl die Aktualisierungsanwendung als auch der eIP-Client greifen auf die Konfigurationsdatenbank zu, um den Schlüssel auszulesen und die Strukturdateien zu verschlüsseln (Aktualisierungsanwendung) bzw. zu entschlüsseln (eIP-Client).

Eine zusätzliche Prüfsumme ist für die Inhalte des dezentralen Aktenstrukturcaches nicht notwendig, da keine Integritätsprüfung zwingend notwendig ist. Wenn eine korrekte Entschlüsselung stattfinden kann, ist ein korrekter Transport durchgeführt worden.

6.5.6.5 Ablagestrategie im lokalen Aktenstrukturcache

Die Ablagestrategie im lokalen Aktenstrukturcache muss nicht zwingend mit der im dezentralen Aktenstrukturcache übereinstimmen. Nachdem der lokale Aktenstrukturcache jederzeit aus dem dezentralen Aktenstrukturcache wieder aufgebaut werden kann, können hier auch alternative Formate betrachtet werden bzw. sogar überlegt werden, ob serialisierte Java-Objekte mit modifizierter Ablage nicht weiterhin sinnvoll eingesetzt werden könnten oder sollten. Dafür sind auch alternative Serialisierungsbibliotheken, die ggf. auf höhere Performance ausgelegt sind, nutzbar. Eine Verschlüsselung der Inhalte ist nicht notwendig, da die Dateien in einem nur für den Benutzer zugänglichen Bereich abgelegt werden sollten (abhängig vom Standardverzeichnis).

Die Nachteile der einzeldateibasierten Ablage der Umsetzung bis eIP 1.2.x (hohe I/O Last, Virenscanner) sind als gravierend anzusehen. Die Vorteile der einfacheren Aktualisierung werden als geringer gewichtet bewertet. Dies bedeutet, dass eine Ablagestrategie auf Basis einer Datei pro Akte umgesetzt wurde. Die Konsequenz daraus ist, dass das bei mehreren Veränderungen der eAkte durch einen Benutzer bei jeder Änderung die gesamte Aktenstruktur in den lokalen Aktenstrukturcache geschrieben werden muss. Deshalb wird ein optimiertes Vorgehen umgesetzt, bei dem ein Hintergrundjob den Cacheinhalt nach konfigurierbaren Intervallen in den lokalen Aktenstrukturcache schreibt und nicht bei jeder Änderung. Durch

die Ablage in einer Datei wird auch die Problematik des Virenscanners minimiert, der jeden Dateizugriff einzeln prüft.

Vom dezentralen Aktenstrukturcache bekommt der eIP-Client eine vollständige Aktenstruktur im XML-Format und lädt zusätzlich die OE-weite und private Handakte nach. Anschließend wird dieses in das interne Datenformat überführt und in den lokalen Aktenstrukturcache geschrieben.

Damit eine einheitliche Lösung über die verschiedenen Caches genutzt wird, wird auch hier das XML-Format genutzt. Die geringfügigen Performanceunterschiede beim Laden der XML-Dateien werden als weniger relevant eingestuft als die unterschiedlichen notwendigen Implementierungstechniken.

Im lokalen Aktenstrukturcache ist in einer XML-Datei die gesamte Akte für den aktuellen Benutzer inkl. Handakten abgebildet.

Eine Integritätsprüfung der Cachedateien im Aktenstrukturcache ist nicht notwendig, da durch Änderungen der Inhalte keine erweiterten Rechte o.ä. erlangt werden können. Es würden lediglich andere Daten im eIP-Client angezeigt werden.

Wenn die XML-Datei korrupt ist und nicht sinnvoll geladen werden kann (z. B. XML-Parserfehler), dann wird diese Datei aus dem Cache gelöscht und der Cache für diesen Bereich neu geladen (z. B. vom dezentralen Aktenstrukturcache oder vom DMS).

6.5.6.6 Fallback-Strategie bei Nicht-Erreichbarkeit des dezentralen Aktenstrukturcaches

Wenn der dezentrale Aktenstrukturcache nicht erreichbar ist, wird für die Aktualisierung des lokalen Aktenstrukturcache weiterhin auf das DMS (über den DigitalFilesService) zugegriffen. Dabei wird die Aktenstruktur vollständig geladen.

Beim Öffnen einer Akte wird grundsätzlich gegen das DMS die Aktualität der lokal gecachten Daten initiiert. Dabei wird lokal der letzte Aktualisierungszeitpunkt in der XML-Struktur vermerkt und für die Aktualisierungsprüfung gegen das DMS genutzt. Dadurch können unnötige Abfragen vermieden werden.

6.5.6.7 Suche nach Akten im Offline-Modus

Der Aktenstrukturcache wird nicht mehr vollständig in den Speicher geladen. Somit ist eine alternative Lösung für die Suche nach Akten im Offline-Modus notwendig, da der eIP-Client auch vollständig ohne Netzwerkverbindung in der Lage sein muss, den Offline-Cache zu durchsuchen. Ein Sichten sämtlicher Aktenstrukturdateien kommt aufgrund der Virenscan-

ner-Problematik nicht in Frage. Alternativ könnte aufgrund der Dateinamen der Aktenstrukturdateien Rückschlüsse auf das Aktenzeichen geführt werden. Dies ist aber bei sehr großen Gerichten (mehrere zehntausende Akten) ebenfalls unrealistisch.

Für jede Aktenstruktur, welche sich im Cache befindet, wird dies in einer separaten Datei (z. B. CachedDigitialFiles) zusätzlich abgelegt. In dieser Datei wird das Aktenzeichen in lesbarer Form zusammen mit der technischen ID der Aktenstruktur abgelegt. Durch dieses Tupel kann bei einem Suchtreffer das entsprechende Aktenzeichen angezeigt und das Öffnen der Akte initiiert werden.

Für die Speicherung der Daten im Dateisystem sind vier Varianten möglich, die auch in der Architekturentscheidung 4.5 (siehe 5.18) dokumentiert ist:

- Java-Properties-Dateien (`key=value`-Listen)
- XML
- JSON
- Serialisierte Java-Objekte (z. B. Map)

Java-Properties-Dateien scheiden aufgrund der Begrenzung auf einen Wert aus (ohne dass dafür wiederum eine eigene Struktur angelegt werden muss).

JSON ist sehr schnell und einfach zu handhaben, XML entsprechend schwergewichtiger.

Aus Gründen der Einheitlichkeit wird auch hier XML verwendet.

Diese Datei muss bei jeder Aktualisierung des Aktenstrukturcaches ebenfalls mit angepasst werden.

Diese Liste der Aktenstrukturen wird permanent im Speicher des Clients gehalten. Bei jeder Veränderung (neue oder gelöschte Aktenstruktur) wird diese Liste im Speicher und auf dem Dateisystem aktualisiert. Die entsprechenden schreibenden Zugriffe auf das Dateisystem werden entsprechend synchronisiert, um parallele Schreibvorgänge vermeiden zu können.

Eine Integritätsprüfung der Cachedatei ist nicht notwendig, da durch Änderungen der Inhalte keine erweiterten Rechte o. ä. erlangt werden können. Es würden lediglich andere Daten im eIP-Client angezeigt werden.

6.5.6.8 Darstellung des Gesamtablaufs

Das folgende Schaubild verdeutlicht die Ablagestrategie, die Zugriffswege sowie die verschiedenen involvierten Komponenten, die im Rahmen des Aktenstrukturcaches genutzt werden.

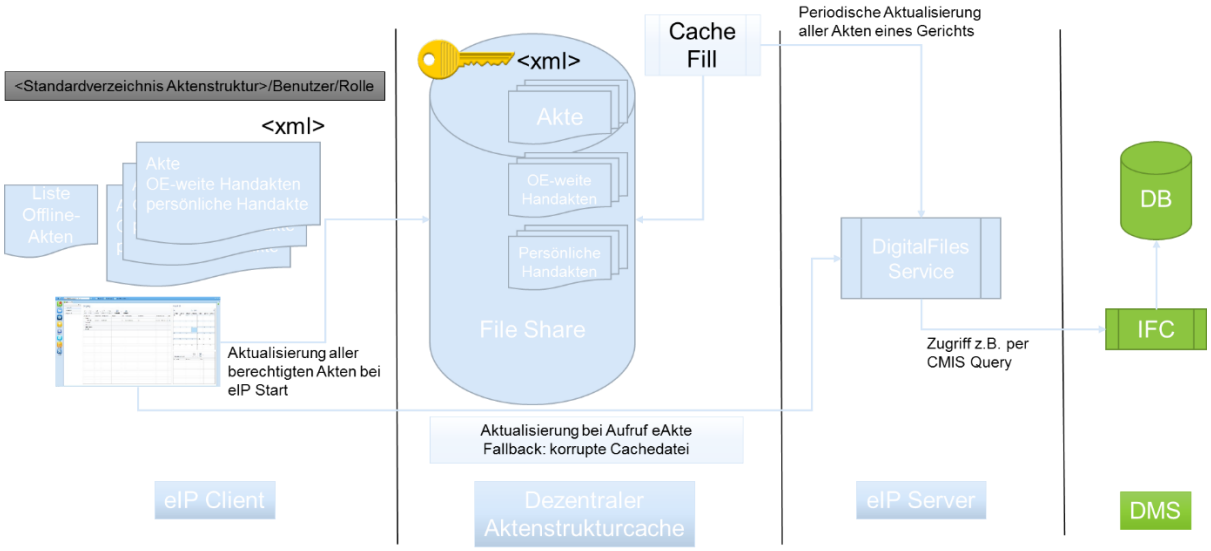

Abbildung 6-9: Gesamtablauf Aktenstrukturcache

6.5.7 Implizite Nutzung des Offline-Modus

Wenn beim Start oder während der laufenden Nutzung von eIP das Fachverfahren oder eine andere Serverkomponente (DMS, eIP Services / Datenbank) ausfällt bzw. nicht erreichbar ist, wird eIP implizit in den Offline-Modus für das ausgefallene System geschaltet und der Benutzer per Dialog darüber informiert. Bereits geöffnete Anwendungen und Fenster bleiben bestehen – es kann allerdings sein, dass damit geöffnete Fenster von aktuell ausgefallenen Systemen bei nächstem Zugriff interne Fehler liefern (z.B. Fachverfahren). Die eIP eigenen Anwendungsteile (Aktenbock, eAkte) schalten auch bei geöffneten Fenstern implizit in die korrekte Arbeitsweise für den Offline-Fall.

Die Erkennung eines Serverausfalls kann dabei je nach Komponente unterschiedlich korrekt erfolgen. Innerhalb von eIP ist eine Kategorisierung der möglichen Fehler erfolgt, anhand der erkannt werden kann, welcher Fehler zu einem Offline-Modus führen muss.

Die korrekte Arbeitsweise je nach Komponente ist außerhalb des Architekturkonzepts definiert (z.B. EIP-265) und legt fest, welche Funktionen in welchen Komponenten bei Ausfall der relevanten serverseitigen Komponenten genutzt werden können. Dies schließt u.a. die Deaktivierung der schreibenden Funktionen (inkl. Speicherung aus Office Produkten in der eAkte)

mit ein. Dazu gehören auch alternative Suchfunktionen (z.B. in den offline verfügbaren Akten).

6.5.8 Wechsel vom Offline-Modus in den Online-Modus

Ein Wechsel vom Offline-Modus in den Online-Modus ist über die Verbindungsübersicht in eIP möglich. In dieser Übersicht kann der Status jeder Komponente dargestellt sowie manuell geprüft werden. Ein Neustart von eIP ist dabei nicht notwendig.

6.5.9 Voraussetzung für die Nutzung des Offline-Modus

Der Benutzer muss mindestens einmal eine erfolgreiche Anmeldung an eIP und damit am SecurityProvider (Fachverfahren) durchgeführt haben. Erst nach diesem Zeitpunkt stehen die sicherheitsrelevanten Informationen im Cache zur Verfügung.

6.6 Mandantenfähigkeit von eIP

Mit eIP 1.3 wurde ein hierarchisches Konfigurationskonzept für eIP eingeführt, damit unterschiedliche Konfigurationswerte bei den verschiedenen Konfigurationsebenen möglich werden. Dies ist vor allem für den weiteren Rollout von eIP in den Ländern notwendig, da z.B. unterschiedliche Einstellungen in den Fachbereichen oder auch den Standorten möglich sein müssen.

Dabei wurden prinzipiell die folgenden Ebenen von der Justiz festgelegt, für die Einstellungen unterschiedlich möglich sein sollen:

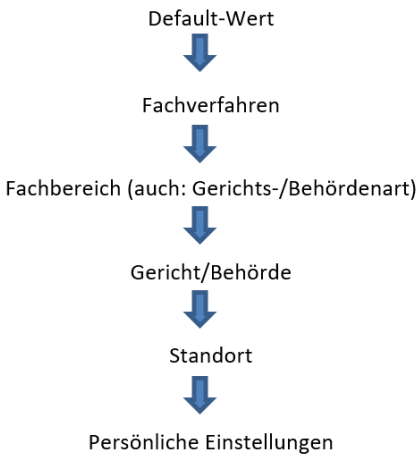

Abbildung 6-10: Hierarchieebenen der Konfiguration

Dies ergibt dann den folgenden Beispielausschnitt an möglichen Hierarchieausprägungen, der in Summe allerdings unvollständig ist:

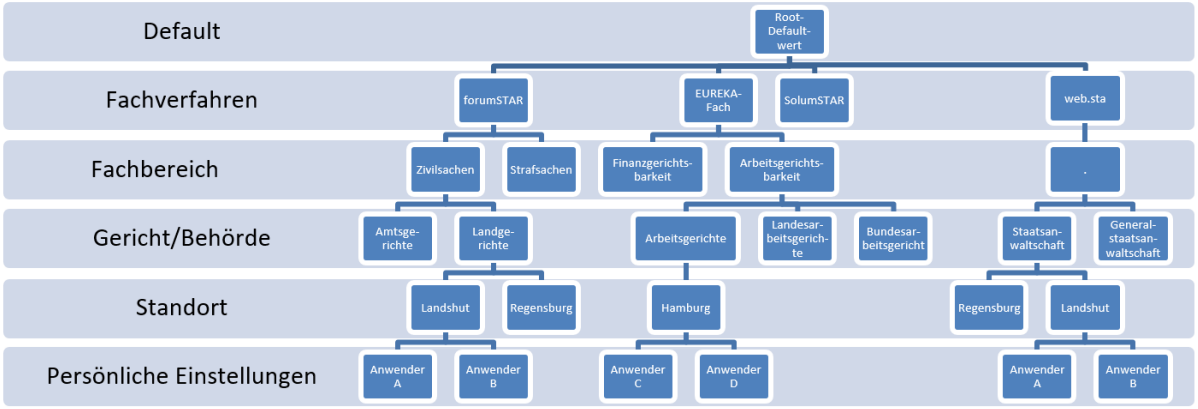

Abbildung 6-11: Beispiel der Hierarchieebenen der Konfiguration

Es sind somit Einstellungen auf jeder Hierarchieebene möglich. Dabei wurde folgendes festgelegt:

- Umgang mit Einzeleinstellungen (d.h. Einstellungen, bei denen es nur einen Wert geben kann)

  z.B. Erinnerungszeitraum, Breite Aktenbaum

  Überschreibend - Werte in unteren Hierarchiestufen überschreiben Werte aus oberen Hierarchiestufen

- Umgang mit Mehrfacheinstellungen (d.h. Einstellungen, die aus Listen bestehen)

  z.B. Kurztexte, Vorlagegründe, Stempel

  Ergänzend / additiv - Werte in unteren Hierarchiestufen ergänzen Werte aus oberen Hierarchiestufen

  Sichtbarkeit und Reihenfolge von einzelnen Einträgen der oberen Hierarchiestufen sind auf allen Ebenen änderbar

Im Rahmen der Entwicklung wurde mit der Justiz abgestimmt, welche Einstellungen in der ersten Umsetzungsstufe hierarchisch abgebildet werden sollen und welche nicht.

Die Ebene Fachverfahren ist derzeit nicht über die Hierarchieebenen abgebildet. Hier wurde derzeit entschieden, ein eigenes Datenbankschema pro Fachverfahren anzulegen. Somit besteht derzeit die Notwendigkeit, für jedes integrierte Fachverfahren einen eigenen eIP Server mit eigener eIP Datenbankanbindung umzusetzen.

Die Umsetzung der Hierarchie erfolgt mit Hilfe des bereits existierenden Attributs SCOPE in der Tabelle CONFIGURATION. Jede Hierarchieebene wird dabei durch einen Punkt („.“) von der nächsten Ebene getrennt. Wenn eine Hierarchieebene in einem Fachverfahren nicht benötigt wird, wird diese durch einen Leerstring repräsentiert – dies hat zur Folge, dass 2 Punkte einander folgen.

Mehrere Werte pro Konfigurationseintrag werden durch einen geeigneten Postfix in der Spalte KEY gelöst. Zusätzlich werden weitere Einschränkungen (Sichtbarkeit), die Sortierung und die Registerzeichenzuordnung über weitere Spalten in der Konfigurationstabelle gelöst.

Für die Umsetzung der hierarchischen Konfigurationsmöglichkeiten sind Stammdaten notwendig. Diese Stammdaten beschreiben die Hierarchieebenen, die für diese eIP Installation genutzt werden sollen. Die Stammdaten bestehen aus einem technischen Schlüssel, der im Scope-Attribut der Einstellungen als Hierarchieebene genutzt wird, sowie einem Anzeigenamen, der in den Oberflächen für die Darstellung der Ebene verwendet wird.

Die Stammdaten werden in entsprechenden Datenbanktabellen abgelegt (siehe 10.1).

Die Pflege der Stammdaten sowie der Konfigurationseinstellungen (bis auf persönliche Einstellungen) erfolgt mit einer separaten Administrationsanwendung. Persönliche Einstellungen sind über die Einstellungsdialoge im eIP Client konfigurierbar.

7 SERVERSEITIGE BASISDIENSTE VON EIP

Die serverseitigen Basisdienste von eIP stellen sich in der Architekturübersicht folgendermaßen dar:

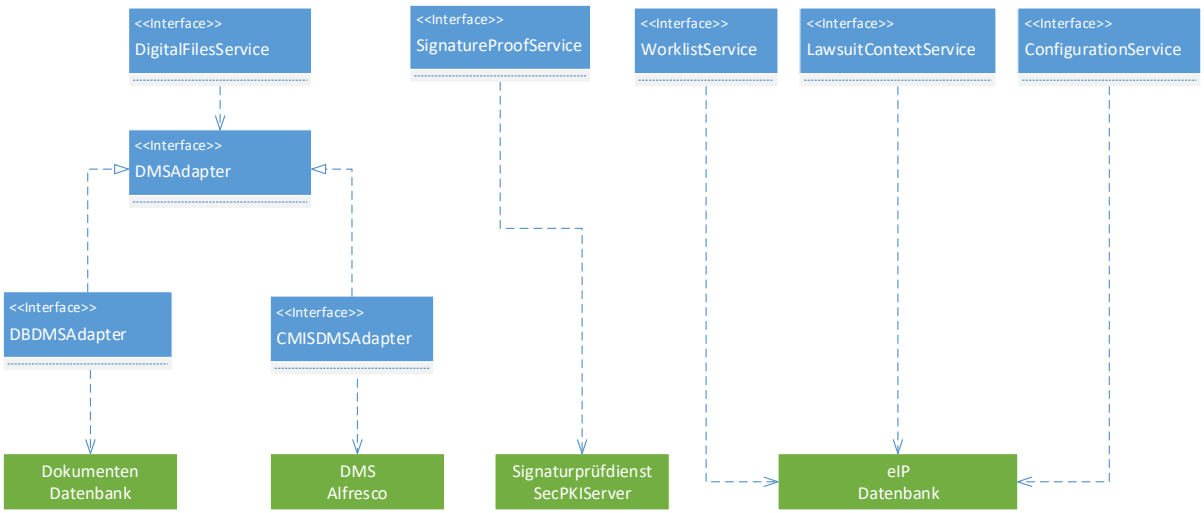

Abbildung 7-12: Komponentendiagramm der serverseitigen Basisdienste von eIP

7.1 Konfigurationsverwaltung (ConfigurationService)

Der *ConfigurationService* ist ein Service, der Konfigurationsdaten für alle Services und den Client bereitstellt. Dies können Schlüssel-Wert-Konfigurationsdaten je Komponente oder allgemeingültige/übergreifende sein. Bei den Schlüssel-Wert-Konfigurationsdaten wird der Komponentenname als Namespace genutzt.

In der Konfiguration werden dabei persönliche und allgemeine Einstellungen unterstützt sowie auch Binärdaten (z.B. Bilder, Texte o.ä.). Persönliche Einstellungen überlagern dabei allgemeine Einstellungen für den gleichen Schlüssel. Zusätzlich kann jede Einstellung einen Scope erhalten, in dem sie gültig ist. Dies ist vor allem bei unterschiedlichen Einsatzgebieten von eIP zwingend notwendig (z.B. Voreinstellungen je nach Fachverfahren oder Fachbereich).

Der Service bietet die Möglichkeit, Konfigurationsdaten zu schreiben, zu lesen und zu löschen.

7.2 Aufgabenverwaltung (WorklistService)

Der *WorklistService* ist der Service, der die eIP-eigenen Aufgaben verwaltet. Aufgaben können dabei erzeugt, gelöscht und geändert sowie Ordnern zugewiesen werden. Der Service stellt Methoden für diese Manipulationen zur Verfügung. Zusätzlich werden Methoden zur Zuweisung einer Aufgabe, die Abfrage nach Aufgaben und die Abfrage nach Aufgaben mit sofortiger Benachrichtigung des Adressaten angeboten. Die Aufgaben von eIP werden dabei in eigenen Datenbanktabellen gehalten. Das Datenmodell ist in Kapitel 10.1 beschrieben. Eine Aufgabe kann dabei einen Freitext oder einen zugewiesenen Vorlagegrund haben, der als Liste in einer eigenen Tabelle abgelegt und abgefragt werden kann.

Zusätzlich zur eigentlichen Aufgabenverwaltung ist das Thema der Protokollierung der Vertreterzugriff auf einen Aktenbock in diesem Service abgelegt. Zugriffe werden entsprechend in einer Datenbank protokolliert und ein Benutzer kann die Zugriffshistorie über die Oberfläche betrachten und löschen. Der Service stellt entsprechende Methoden zur Verfügung.

Es kommt keine Workflow Engine oder ähnliches zum Einsatz, um die verschiedenen Aspekte der Aufgabenverwaltung abzudecken.

Die fachliche Logik, wie z.B. das Verschieben von Aufgaben aus den Ausgängen anderer Aktenböcke in den Aktenbock des aktuellen Benutzers beim Nachladen der Aufgaben, ist dabei in diesem Service umgesetzt und wird vom Client lediglich verwendet. Der Client stellt eine reine Anzeige der Aufgaben zur Verfügung.

7.3 eAkte Dienste (DigitalFilesService)

Der *DigitalFilesService* ist ein Service, der die fachliche Logik der elektronischen Akte und die technische Anbindung an einen Dokumentenspeicher beinhaltet. Diese beiden Bereiche stellen unterschiedliche technische Komponenten innerhalb des Service dar.

Die fachliche Logik ist als externer Service als Remote EJB umgesetzt. Dabei existieren interne Serviceoperationen, d.h. nur für die eIP-eAkte internen Aktivitäten, als auch externe Serviceoperationen, die z.B. von Fachverfahren genutzt werden (siehe dabei auch Kapitel 7.6).

Interne Serviceoperationen stellen die Kernfunktionen dar, die in der eAkte-Komponente von eIP nutzbar sind. Darunter fallen unter anderen die folgenden Funktionen:

- Erzeugen, Aktualisieren, Verschieben oder Löschen einer Akte, eines Dokuments, einer Annotation oder eines Ordners
- Bereitstellen der Aktenbaumstruktur und Metadaten der Dokumente
- Dokument trennen

- Paginierung und Neupaginierung eines oder mehrerer Dokumente
- Kopieren von Dokumenten in die gleiche oder eine andere Akte
- Anlegen von Merkzetteln
- Festlegung eines Aktendeckeldokuments
- Erzeugen einer Kurzverfügung
- Erzeugen eines Fehlblatts
- Volltextsuche über eine, bestimmte oder alle berechtigten Akten

Weiterhin ist im *DigitalFilesService* die Anbindung der Konvertierung über das Scan Subsystem angesiedelt. Dieses wurde in diesem Service untergebracht, da die Konvertierung grundsätzlich auf einem Dokument basiert, welches bereits im Dokumentenspeicher abgelegt ist und damit serverseitig vorliegt. Für diesen Zweck sind mehrere Operationen vorhanden, u.a. sind dies:

- Start der Konvertierung für ein Dokument
- Prüfung der Konvertierungsstatus und Bereitstellung des konvertierten Dokuments

Die externen Serviceoperationen können je nach Fachverfahren unterschiedlich ausgestaltet sein (siehe dazu 7.6), decken aber grundsätzlich die folgenden fachlichen Aktionen ab:

- Anlegen einer elektronischen Akte
- Anpassen der Metadaten einer elektronischen Akte
- Trennen einer elektronischen Akte
- Verbinden von zwei elektronischen Akten
- Ablegen, Aktualisieren und Verschieben von Dokumenten in der elektronischen Akte

Zu verschiedenen Zwecken muss entweder ein PDF erzeugt (z.B. Fehlblatt, Kurzverfügung) oder ein PDF im Rahmen der verschiedenen fachlichen Aktionen verändert werden (z.B. Dokument trennen, Paginierung aufbringen usw.). Für diesen Zweck wird eine Open Source Bibliothek (Apache pdfBox) verwendet, um diese Veränderungen serverseitig zentral durchführen zu können. Dabei werden grundsätzlich nur PDF/A-1 bzw. PDF/A-2 Dokumente erzeugt.

Die technische Anbindung an den Dokumentenspeicher und die eingesetzte Abstraktionsebene sind in Kapitel 6.3.1 detailliert beschrieben. Die technische Umsetzung erfolgt unter Einsatz von lokalen Enterprise Java Beans, welche die technische Anbindung als interne

Komponente kapseln und verschiedenen fachlichen Services die Zugriffe zur Verfügung stellen. Die Identifikation des korrekten DMS-Adapters erfolgt unter Einsatz des Factory-Patterns, welches basierend auf den Konfigurationsdaten in der Datenbank die korrekte Implementierung des DMS-Adapters erzeugt und nutzt.

Die Anbindung von Alfresco über den CMIS Adapter ist als eigenständiges EAR mit Remote EJBs umgesetzt. Dies wurde mit der Unterstützung von Oracle Weblog 12.2 notwendig, da der von CMIS verwendete Webservice-Stack im WLS 12 nicht mehr nativ unterstützt wurde und die notwendigen Webservice-Bibliotheken auch die WLS eigenen Stack beeinflusst hat. Durch die Trennung der EARs sind somit Integration Webservice und CMIS Webservice Aufrufe sauber voneinander getrennt.

7.4 Arbeitskontextverwaltung (LawsuitContextService)

Der *LawsuitContextService* ist der Service, der die Historie der Arbeitskontexte (Verfahrenshistorie) der Benutzer verwaltet. Arbeitskontexte in der Historie können dabei erzeugt, abgefragt und gelöscht werden. Der Service stellt Methoden für diese Manipulationen zur Verfügung.

Der Service wird in der Maske zur Anzeige des Verfahrensverlaufs genutzt, um die geöffneten Arbeitskontexte der Vergangenheit anzuzeigen oder diese zu löschen. Wenn ein Arbeitskontext geöffnet wird, wird dieser in die Historie geschrieben.

7.5 Signaturprüfung (SignatureProofService)

Der *SignatureProofService* ist der Service für den Zugriff auf einen serverseitigen Signaturprüfdienst, der eine Signatur prüfen und ein Signaturprüfprotokoll erstellen kann. Dafür wird der SecPKIServer der Firma SecCommerce angesprochen.

7.6 Fachverfahrensspezifische Erweiterungen an den Basisdiensten

Der *DigitalFilesService* stellt einige Schnittstellen für Fachverfahren zur Verfügung, damit diese serverseitig genutzt werden können. Intern hat der Service noch wesentlich mehr Funktionen, die für die fachlichen Funktionen der eIP Clientinteraktion geschuldet sind.

Einige Funktionen, die hier als Basisdienste angeboten werden, werden in allen Fachverfahren identisch gehandhabt werden (z.B. Einlagern eines Dokuments in einen Ordner). Es gibt aber ggf. andere fachliche Funktionen, die identisch heißen werden, aber fachliche Unterschiedene bestehen. Beispielsweise wird bei der fachlichen Funktion „Verfahren trennen" eine

bestehende Akte dupliziert. Dies ist für das Fachverfahren forumSTAR spezifiziert worden und wird ggf. nicht auf alle Fachverfahren identisch zutreffen (z.B. kein Kopieren von Entwürfen o.ä.).

Grundsätzlich bestehen hier drei Alternativen:

- Keine fachverfahrensspezifischen Anpassungen der wiederverwendbaren Basisservices
    - Nicht realistisch durchhaltbar
- Umsetzung der fachverfahrensspezifischen Anpassungen innerhalb des Basisservice
    - Verlagerung von Fachverfahrenslogik in einen Basisservice erzwingt Verzweigungen im Code und verkompliziert die Wartung und Pflege
    - Kollateralschäden möglich
- Eigener EJB Service (z.B. ForumStarDigitalFilesService) für jedes Fachverfahren mit Umsetzung der spezifischen Funktionalität und Nutzung der internen Schnittstelle des eIP Services möglich für identische Funktionalität
    - Eigenes EJB als vorgeschaltete Instanz notwendig für jedes Fachverfahren mit maximaler Flexibilität
    - Keine Kollateralschäden
- Umsetzung über fachverfahrenspezifische Nachrichten in der Integrationsbibliothek und dem Intergration Webservice
    - Keine Kollateralschäden
    - Umsetzung in der Integrationsschicht des jeweiligen Service

Es wird die letzte Variante für die Umsetzung von fachverfahrensspezifischen Erweiterungen aufgrund der höheren Flexibilität und optimierter Wartungsmöglichkeiten ohne Kollateralschäden für andere Bereiche gewählt.

8 ARCHITEKTURSICHTEN

8.1 Systemkontext

Das folgende Systemkontext Diagramm stellt die beteiligten Systeme im Gesamtkontext dar. Dabei stellen die dünnen Linien Verbindungen dar, die nicht direkt von eIP aus gesteuert bzw. beeinflusst werden (lose Kopplung). Systeme, zu denen eIP eine direkte Verbindung hält, sind durch dicke Linien gekennzeichnet.

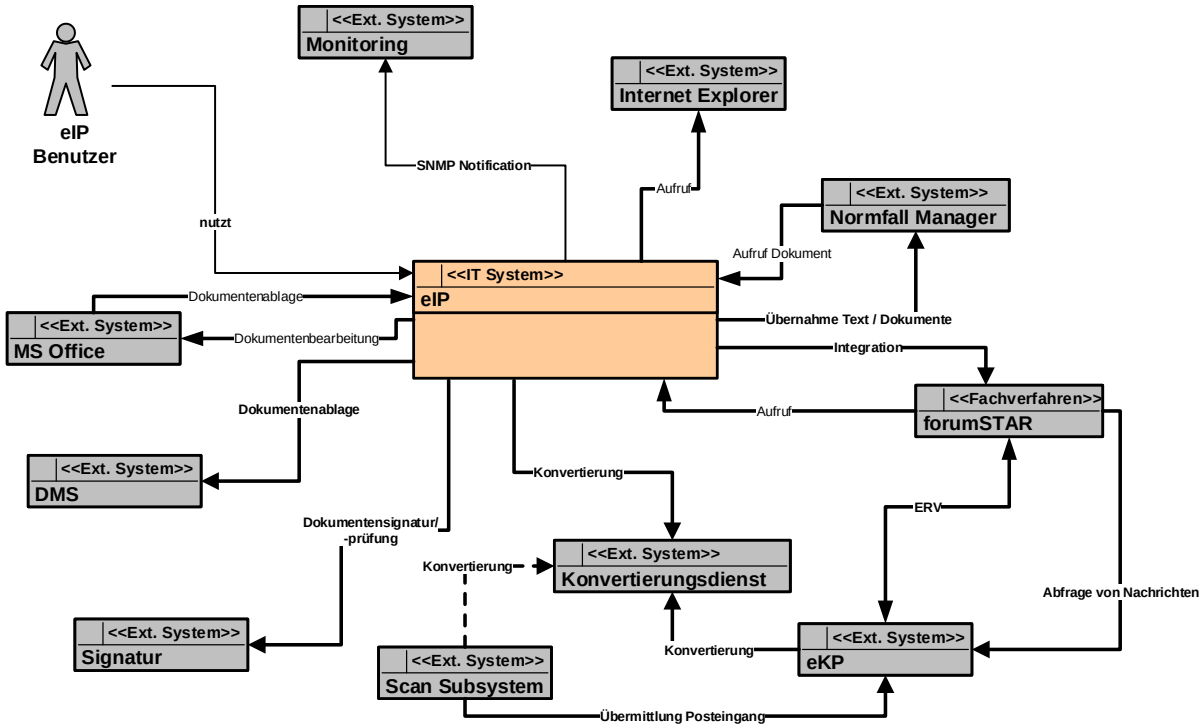

Abbildung 8-13: Systemkontext von eIP mit forumSTAR

Die eIP Services schreiben Log-Dateien, welche vom Monitoring ausgewertet werden können. Weiterhin werden SNMP Traps/Notification Events versendet, um das Monitoring aktiv über Probleme zu informieren. Deshalb besteht eine lose Kopplung zwischen eIP und dem **Monitoring**.

Der Systemkontext unterscheidet sich je nach Fachverfahrensintegration. Die folgenden beiden Beispiele zeigen dies anhand der EUREKA-Fach und SolumSTAR Integration auf.

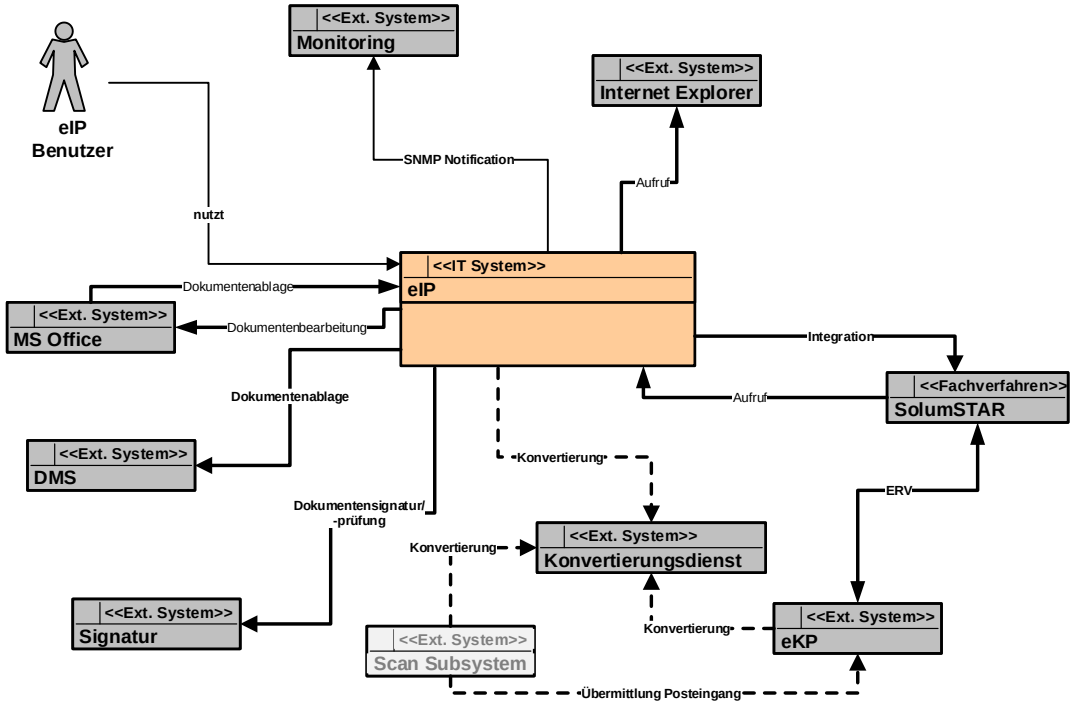

Abbildung 8-14: Systemkontext von eIP mit SolumSTAR

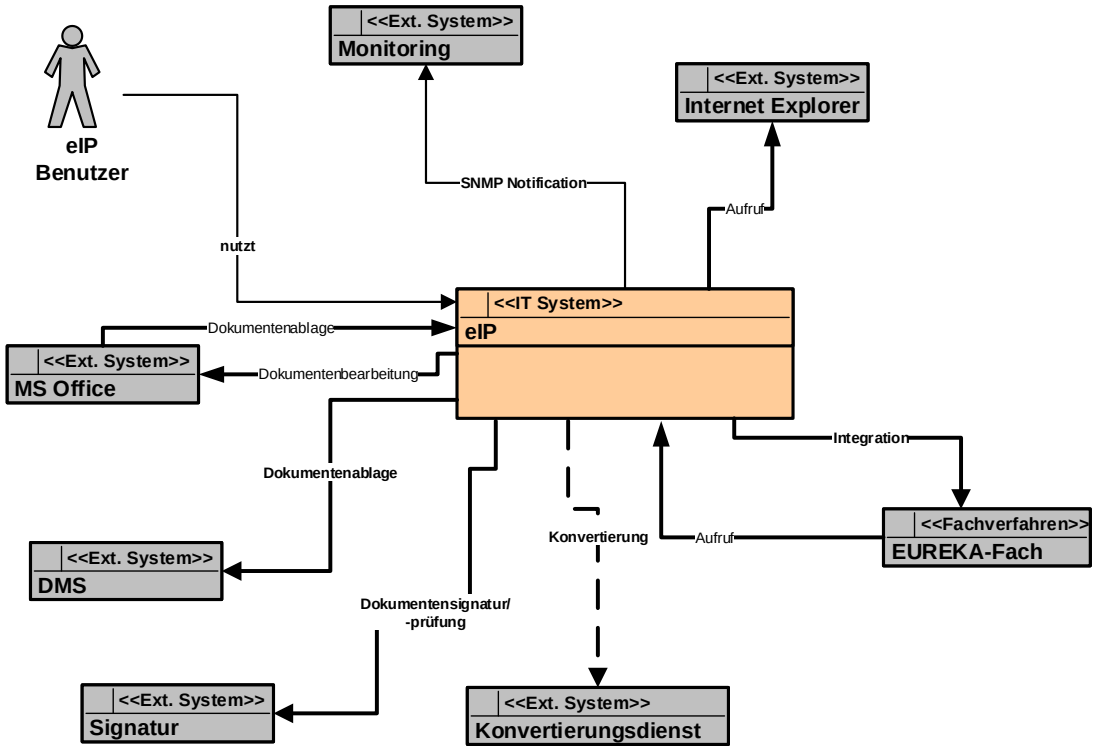

Abbildung 8-15: Systemkontexft von eIP mit EUREKA-Fach

8.2 Operationales Modell

In diesem Kapitel werden das operationale Modell und die Kommunikationswege von eIP kurz dargestellt, um den Gesamtzusammenhang besser verstehen zu können. Das folgende Schaubild zeigt dabei die umgesetzten und angebundenen Systeme.

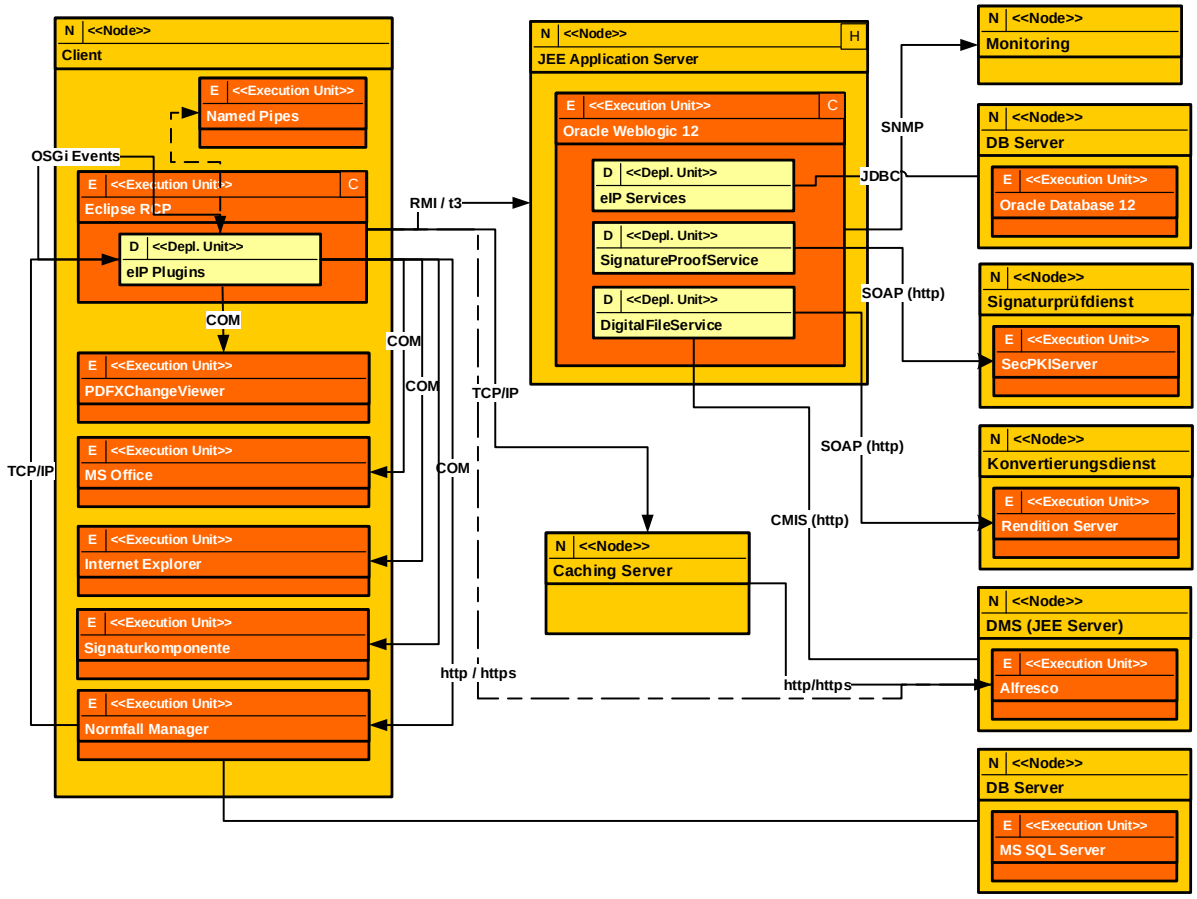

Abbildung 8-16: Operationales Modell von eIP

Die Application Server der Fachverfahren (z.B. forumSTAR) und die Kommunikationswege des Fachverfahrens sind in diesem Bild ausgespart, um die Komplexität zu verringern. Die Kommunikationswege des integrierten Fachverfahrens sind allerdings identisch zu einer klassischen Installation des Fachverfahrens ohne eIP-Integration. Ein vollständiges operationales Modell mit der forumSTAR-Anbindung ist beispielhaft im Integrationsleitfaden dargestellt.

Auf der linken Seite ist der eIP-Client dargestellt, im mittleren Bereich die Application Server und im rechten Bereich die Datenschicht bzw. Monitoring-Komponenten.

Die Kommunikationswege im eIP-Client sind in diesem Dokument schon angesprochen worden. Im Einzelnen sind dies:

- OSGi Events für die Kommunikation zwischen den verschiedenen eIP Plugins im Client
- Object Linking and Embedding (OLE) und COM für die Kommunikation eines Plugins mit einer nativen Windows Anwendung wie dem PDFXChangeViewer
- TCP/IP Kommunikation von einer nativen Anwendung (Normfall Manager) zum eIP Client
- Named Pipes für die Einbindung und Kommunikation von Fachverfahren (optionaler Integrationsmechanismus, abhängig von der Fachverfahrensintegration)

Der eIP-Client kommuniziert mit seinen Services über RMI (Standard EJB Kommunikation) unter der Nutzung des Protokolls t3 (Oracle WebLogic spezifisches Kommunikationsprotokoll).

Für den performanten Zugriff auf die Dokumente der Akte wird die WAN Leitung meist als limitierender Faktor gesehen. Somit ist der Einsatz eines Caching Servers im lokalen Gerichtsnetzwerk möglich.

eIP unterstützt einen Caching Server für die Dokumente, die im DMS (Alfresco) abgelegt sind. Der Caching Server stellt eine Eigenentwicklung der Firma it-novum dar und kann dabei dezentral im Gericht oder zentral betrieben werden (Kaskadierung der Server). Der Caching Server unterstützt dabei eine symmetrische Verschlüsselung der Dokumente. Der eIP Client fordert Dokumente per HTTP/HTTPS vom Caching Server an, entschlüsselt diese und legt sie in dem lokalen eIP Client Cache für die Anzeige ab. Wenn das Dokument noch nicht im Caching Server vorliegt, holt dieser das aktuelle Dokument per HTTP/HTTPS aus dem DMS (Alfresco) und legt dieses verschlüsselt bei sich ab. Sollte der Zugriff auf den Caching Server oder die Entschlüsselung der Dokumente im eIP Client scheitern, greift der eIP Client direkt per HTTP/HTTPS auf das DMS (Alfresco) zu, um das Dokument abzurufen. Der Caching Server ist optional. Dies ist in Abbildung 8-17: Unterstützung für zentrale und dezentrale DMS Caching Server dargestellt.

Der Normfall Manager nutzt eine zentrale MS SQL Server Datenbank als Projektspeicher. Entsprechend greift der Normfall Manager auf diesen zu. Der Normfall Manager ist optional.

Die eIP Server Komponenten (eIP Services bestehend u.a. aus *WorklistService* und *ConfigurationService*) greifen auf die eIP eigene Datenbank per JDBC unter Einsatz von JPA zu, um Konfigurationsdaten und Arbeitsaufträge auszulesen.

Die serverseitige Logik der elektronischen Akte (*DigitalFilesService*), welche die fachliche Logik der Aktenmanipulation sowie den Adapter zum DMS beinhaltet, greift unter Nutzung von CMIS (Content Management Interoperability Services) über HTTP/HTTPS auf das DMS (Alfresco) zu. Dabei können die Aktenstrukturen und Metadaten ausgelesen und verändert, Suchanfragen gestellt oder Dokumente hinzugefügt werden.

Zusätzlich kann über den *DigitalFilesService* die Konvertierung über einen Konvertierungsdienst (z.B. Luratech Rendition Service oder das Scan Subsystem) initiiert und die Ergebnisdokumente abgefragt werden. Dafür findet ein SOAP-Zugriff auf die Webservices des Konvertierungsdienstes statt.

Ein serverseitig vorhandener Signaturprüfdienst kann für die Erzeugung eines Signaturprüfprotokolls sowie zur Prüfung einer Signatur herangezogen werden. Dieser wird über den *SignatureProofService* per SOAP-Zugriff angesprochen. Es wird der SecPKIServer unterstützt.

Das Monitoring kann per SNMP v2c Notification Events über Fehlerfälle informiert werden.

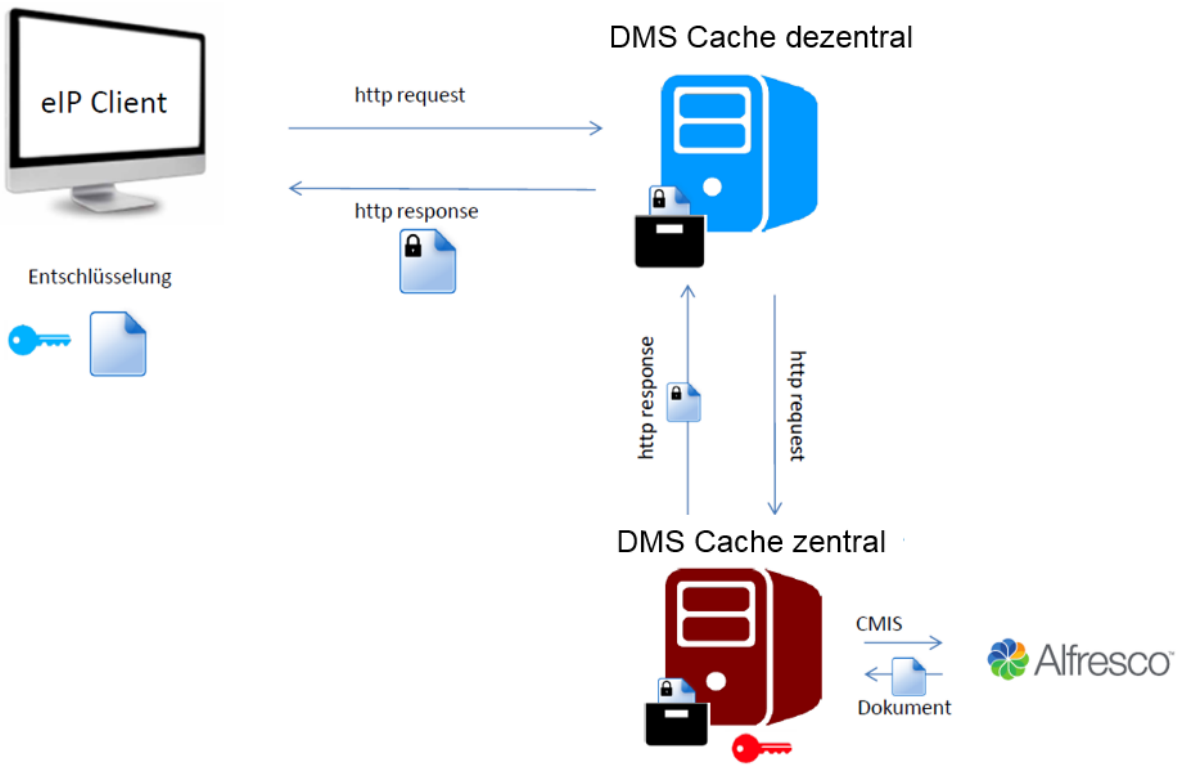

Abbildung 8-17: Unterstützung für zentrale und dezentrale DMS Caching Server

9 UNTERSTÜTZUNG DES BETRIEBS

Für einen ausfallsicheren Betrieb ist es unerlässlich, proaktiv die Verfügbarkeit der Services sicherzustellen und nicht erst bei einem Fehler aus dem Betrieb heraus zu reagieren.

Zu diesem Zweck muss der Betrieb die Basiskomponenten sowie Kommunikationswege ständig überwachen. Dies ist allerdings im Betriebskonzept zu definieren und liegt außerhalb des Architekturkonzeptes.

Um allerdings eine proaktive Verfügbarkeit der Services möglich zu machen, müssen die Services entsprechende standardisierte Schnittstellen bedienen.

Jeder Service muss dabei die beiden folgenden Methoden implementieren:

- `getVersion`

  Liefert die aktuelle Version des Service

- `performHealthCheck`

  Führt eine Funktionsprüfung des Service durch, in dem z.B. alle notwendigen Kommunikationswege geprüft und Testdaten geschrieben bzw. verarbeitet werden (z.B. Testzugriff auf die Datenbank)

Zusätzlich muss jeder Fehler, der dem Betrieb bekannt sein muss, eine SNMP-Notification v2c für das System Management Plattform absetzen.

Die Überwachungsmethoden der Services müssen aktiv von einem System Management Werkzeug (z.B. Nagios) gerufen werden. eIP ruft und überwacht diese Funktionen nicht selbständig. In eIP existiert allerdings eine Verbindungsübersicht, die die aktuell verfügbaren Verbindungen von diesem eIP Client aus darstellen und prüfen kann.

Für jeden Service werden mögliche Fehler im Fehlerhandbuch dokumentiert.

10 DATENMODELL

Innerhalb von eIP ist der Zugriff auf eine Datenbank für Konfigurations- und Aufgabendaten notwendig. Der Zugriff auf die Datenbank erfolgt mit den festgelegten Persistenzmechanismen unter Einsatz von Java Persistence API (JPA) auf Basis von Open JPA.

Weiterhin ist ein Zugriff auf einen Dokumentenspeicher notwendig, der die gesamte elektronische Akte von eIP darstellt. Die Metadaten und die Dokumente werden dabei vollständig in diesem Dokumentenspeicher gehalten.

Zwischen der Datenbankinhalten und dem Dokumentenspeicher besteht keinerlei zwingende Verbindung oder Abhängigkeit. Somit kann das Datenmodell für die Datenbank und den Dokumentenspeicher auch unabhängig voneinander betrachtet werden.

Das Datenmodell basiert auf dem Fachklassenmodell, welches im Rahmen der fachlichen Dokumentation ebenfalls erstellt wurde.

10.1 Datenbankmodell

Das Datenmodell von eIP besteht aus zehn Datenbanktabellen.

Tabellenname	Beschreibung
CONFIGURATION	Konfigurationstabelle mit den Konfigurationseinstellungen, die persönlich oder generell gültig sein können
WORKLIST_FOLDER	Ablage der möglichen Ordner im Aktenbock
WORKLIST_ENTRY	Speicherung der Aufgaben für Gruppen oder Personen
WORKLIST_ENTRY_ARCHIVE	Speicherung der archivierten/erledigten Aufgaben
WORKLIST_DELEGATE_ACCESS	Speicherung der Zugriffe eines Benutzers auf den Aktenbock eines anderen Benutzers (Vertreterzugriffe)
LAWSUIT_CONTEXT_ENTRY	Speicherung der geöffneten Arbeitskontexte (Historie)
XJUSTIZ_FOLDER_TYPE	Ablage der möglichen XJustiz Ordner (Teilakten) Typen
XJUSTIZ_DOCUMENT_TYPE	Ablage der möglichen XJustiz Dokumentenarten
CONFIGURATION_SCOPE_KEY_TYPE	Ablage der möglichen Schlüsseltypen (Hierarchieebenen) für die mandantenfähige Konfiguration
CONFIGURATION_SCOPE_KEY	Ablage der möglichen Schlüssel für die mandantenfähige Konfiguration mit Schlüssel- und Anzeigenamen (Stammdaten der Konfigurationshierarchie)
XJUSTIZ_VERSION	Ablage der unterstützten XJustiz-Versionen

Tabellenname	Beschreibung
XJUSTIZ_DOCUMENT_VERSIONS	Verknüpfungstabelle zwischen XJUSTIZ_DOCUMENT_TYPE und XJUSTIZ_VERSION für die Zuordnung der Dokumententypen zu einer XJustiz-Version
XJUSTIZ_FOLDER_VERSIONS	Verknüpfungstabelle zwischen XJUSTIZ_FOLDER_TYPE und XJUSTIZ_VERSION für die Zuordnung der Ordnertypen zu einer XJustiz-Version
XJUSTIZ_DOCUMENT_CATEGORIES	Verknüpfungstabelle zwischen CONFIGURATIO und den darin abgelegten Kategorien, XJUSTIZ_DOCUMENT_TYPE und XJUSTIZ_VERSION für die Zuordnung der Dokumententypen zu einer XJustiz-Version

Tabelle 10-2: Übersicht der Datenbanktabellen

Das ER[10]-Modell in diesem Kapitel zeigt die physische Sicht auf die Daten. Die Primärschlüssel werden durch technische Schlüssel abgebildet, die über eine Sequenz befüllt werden.

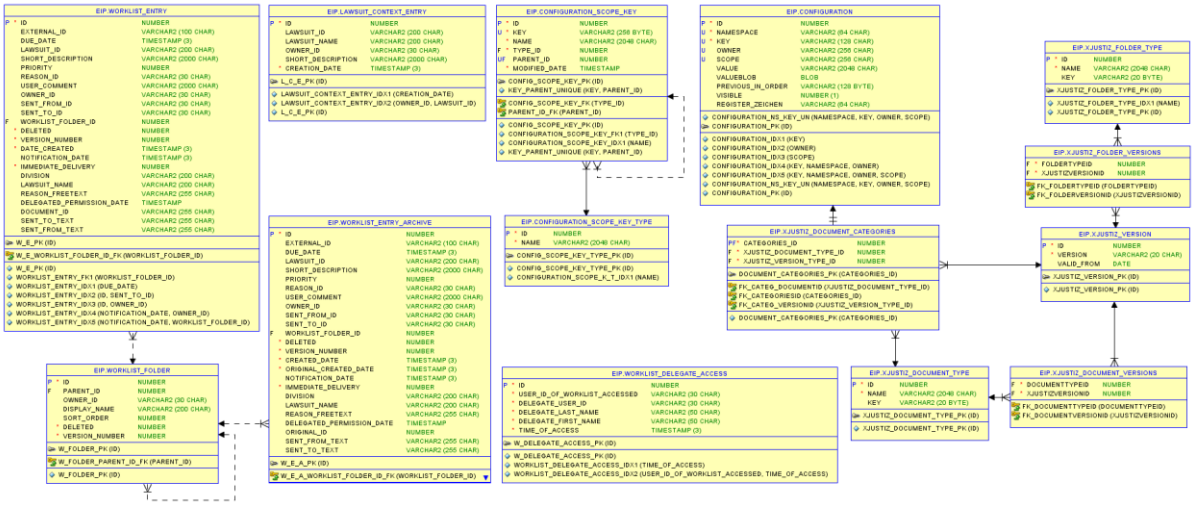

Abbildung 10-18: Datenbankmodell von eIP

10.2 Dokumentenspeichermodell

In der aktuellen Umsetzung von eIP wird ein Dokumentenspeicher auf Basis eines Standard-DMS-Produkts (Alfresco) eingesetzt. Der Zugriff erfolgt dabei per CMIS, womit die Mechanismen und Gestaltungsmöglichkeiten durch die CMIS-Implementierung in Alfresco beschränkt werden.

[10] Entity Relationship

Dabei wird die eAkte Struktur als Ordner im DMS abgebildet. Die folgende Struktur wird dabei genutzt:

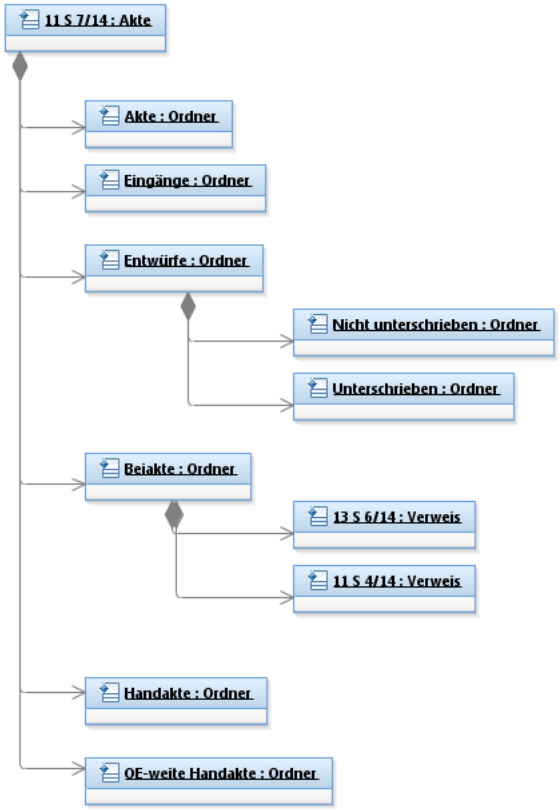

Abbildung 10-19: Struktur eAkte

- Ordner Ebene 1: Aktenzeichen
  - Ebene 2: Akte
  - Ebene 2: Eingänge
    - Ebene 3: Zeitstempel des Datenimports bzw. Nachrichteneingangs
  - Ebene 2: Entwürfe
    - Ebene 3: unterschrieben
    - Ebene 3: nicht unterschrieben
  - Ebene 2: Beiakten
  - Ebene 2: OE-weite Handakte

  - Ebene 2: Handakte

- Benutzerabhängige Dokumente sind in Unterverzeichnissen abgelegt
  - Beispiel: Handakte/RichterA

Die Ordner sowie Dokumente im DMS verwalten erweiterte Eigenschaften (z.B. Paginierungsinformation), die von der eAkte-Komponente genutzt werden. Es sind keine weiteren Datensenken für die Speicherung der Dokumenteninformationen notwendig.

Die Relation zwischen Dokumenten (z.B. XFDF → PDF) erfolgt über Attribute im DMS sowie unter Einsatz von CMIS-Relationships.

Es werden grundsätzlich zehn verschiedene Typen von Objekten im DMS verwaltet. Dabei sind die fachverfahrensspezifischen Metadaten in den jeweiligen Objekten enthalten:

Typ	Technischer Typ	Beschreibung
Dokument	eip:eipDocument	Ablagetyp für Dokumente
Ordner	eip:eipFolder	Ordner, welche wiederum Ordner oder Dokumente beinhalten können
Annotation	eip:eipAnnotation	Ablagetyp für Annotationen (Anmerkungen)
EakteNoteDokument	eip:eipEakteNoteDocument	Notizdokument an einer Akte (Notizzettel, Post-it)
EakteCoverDokument	eip:eipEakteCoverDocument	Stammdatenblatt bzw. Deckblattdokument für eine Akte
Signatur	eip:eipSignature	Ablagetyp für Signaturdateien
SignaturPruefprotokollDokument	eip:eipSignatureVerificationDocument	Ablagetyp für Signaturprüfprotokolle
TransfervermerkDokument	eip:eipTransferNoteDocument	Ablagetyp für Transfervermerke von ERV-Nachrichten
ReferenzOrdner	eip:eipReferenceFolder	Ablagetyp für Referenzen auf Beiakten
Verweisungsvermerk auf Dokumente	eip:eipDocumentLink	Ablagetyp für einen Verweisungsvermerk auf eine Akte, einen Ordner oder ein Dokument
Verweisungsvermerk auf Ordner und Akten	eip:eipFolderLink	Ablagetyp für einen Verweisungsvermerk auf eine Akte oder einen Ordner
XJustizNachricht	xJustiz:message	Ablagetyp für eine XJustiz-Nachricht

Die folgenden Tabellen zeigen dabei die möglichen Attribute je nach Typ auf.

Attributname	Kurzbeschreibung	Beschreibung	Werteliste	Attributtyp	Erforderlich
cmis:createdBy	Ersteller	Der Ersteller des Objekts im DMS	N/A	String	Ja
cmis:creationDate	Erstelldatum	Das Erzeugungsdatum dieses Objekts im DMS	N/A	Date	Ja
cmis:lastModificationDate	Änderungsdatum	Das Datum der letzten Änderung des Objekts	N/A	Date	Ja
cmis:lastModifiedBy	Änderungsbenutzer	Der Benutzer, der die letzte Änderung durchgeführt hat	N/A	String	Ja
cmis:name	Name	Name des Objekts	N/A	String	Ja
cmis:objectId	Objekt Id	Eindeutige Objekt-Id	N/A	String	Ja
cmis:objectTypeId	Objekt Typ	Id des Objekt-Typs	OBJECT_TYPE_ID	String	Ja
cmis:parentId	Vater Objekt Id	Die Objekt Id des Vaterelements in der hierarschischen Struktur	N/A	String	Ja
cmis:path	Pfad	Der vollqualifizierte Pfad zu dem Ordner	N/A	String	Ja
dabag:gericht	Gericht in dabag	Zur Bildung des Aktenzeichens bei Grundakten	N/A	String	Nein
dabag:bezirk	Bezirk in dabag	Zur Bildung des Aktenzeichens bei Grundakten	N/A	String	Nein
dabag:blattnummer	Blattnummer in dabag	Zur Bildung des Aktenzeichens bei Grundakten	N/A	String	Nein
dabag:anzeigename	Anzeigename in dabag	Für Akten: Aktenzeichen als Name für Anzeige im Aktenbaum des Aktenviewers (Anzeigename) oder Firmenname für Handelsregisterakte oder für Unterordner: Ordnungsnummer oder Vorgangsnummer im Anzeigeformat	N/A	String	Nein
dabag:abteilung	Abteilung in dabag	oder für Unterordner: Ordnungsnummer oder Vorgangsnummer im Anzeigeformat	N/A	String	Nein
dabag:jahr	Jahr in dabag	Zur Bildung des AR-Aktenzeichens	N/A	String	Nein
dabag:laufendeNummer	Laufende Nummer in dabag	Zur Bildung des AR-Aktenzeichens	N/A	String	Nein
dabag:papierakteVorhanden	Papierakte Vorhanden	Falls null, kann davon ausgegangen werden, dass es sich um eine Papierakte oder Akte aus SolumStar handelt. Für dabag und AuRegis Standardwert "Unbekannt". Andere Werte sind ja und nein. Bei komplett neuen Akten können die Systemfassaden von dabag und AuRegis „nein“ vergeben.	N/A	Boolean	Nein
dabag:folderOrdnungsnummer	Folder Ordnungsnummer	Laufende Nummer / Ordnungsnummer in dabag; Vorgangsnummer in AuRegis	N/A	String	Nein
dabag:eingangsdatumVorgang	Eingangsdatum Vorgang in dabag	Datum mit Uhrzeit des Eingangs des ersten Antrags zum Vorgang	N/A	Date	Nein
dabag:eingangsordner	Eingangs Ordner in dabag	Beschreibt, ob es sich um einen Eingangsordner handelt; evtl. auch Ermittlung über Vorlage	N/A	Boolean	Nein
dabag:entwurfsordner	Entwurfs Ordner in dabag	Beschreibt, ob es sich um einen Entwurfsordner handelt; evtl. auch Ermittlung über Vorlage	N/A	Boolean	Nein
dabag:vorgangsId	Vorgangs Id in dabag	Vorgangs ID aus dem FV falls von dort aus angelegt	N/A	String	Nein
eip:folderCreator	Ordnerersteller	Der Erzeuger des Dokuments aus Sicht der Fachanwendung	N/A	String	Ja
eip:folderDoPaging	Paginierungsart des Ordners	Flag für die Aktivierung / Deaktivierung der Paginierung in dem Ordner	true/false(null)	Boolean	Ja
eip:folderGroupOwner	Benutzergruppe	Die Benutzergruppe, der das Objekt zugeordnet ist	N/A	String	Nein
eip:folderLawsuitId	Eindeutige Verfahrenskennung	Die eindeutige Verfahrenskennung, der der Ordner zugeordnet ist Dies stellt nicht das Aktenzeichen dar, da dieses nicht eindeutig im Fachverfahren ist.	N/A	String	Nein
eip:folderName	Ordnername	Anzeigename des Ordners für die Darstellung im Aktenbaum	N/A	String	Ja
eip:folderOwner	Dokumenteneigentümer	Benutzerkennung, welche das Dokument verändern darf	N/A	String	Nein
eip:folderType	Ordnertyp	Der Typ eines Ordners	ORDNERTYP	Long	Ja
eip:folderChildrenOrder	Ordnerreihenfolge	Die Reihenfolge Liste aller Dokumente und Unterordner	N/A	Long	Ja
eip:folderArchivShortestRetenti	Kürzeste Aufbewahrungsfrist	Datum kürzeste Aufbewahrungsfrist der Dokumente im Ordner	N/A	Date	Nein
eip:folderArchivLongestRetenti	Längste Aufbewahrungsfrist	Datum längste Aufbewahrungsfrist der Dokumente im Ordner	N/A	Date	Nein
eip:folderArchivState	Archivierung der Akte	Informationen über Archivierung	ARCHIVTYP	String	Nein
solumstar:bezirk	Bezirk der Grundakte	Zur Bildung des Aktenzeichens bei Grundakten	N/A	String	Nein
solumstar:blattnummer	Blattnummer der Grundakte	Zur Bildung des Aktenzeichens bei Grundakten	N/A	String	Nein
solumstar:gericht	Gericht der Grundakte	Zur Bildung des Aktenzeichens bei Grundakten	N/A	String	Nein
solumstar:onr	Ordungsnummer der Grundakte	Zur Bildung des Aktenzeichens bei Grundakten	N/A	String	Nein
eip:folderDigitalFileId	ID der e-Akte	Eindeutige ID der e-Akte. Muss nicht der Verfahrens-ID entsprechen. Bei der Grundakte z.B. gibt es keine direkte Zuordnung von Verfahren / Fällen zu Akten	N/A	String	Nein
eip:folderXJustizType	Xjustiz Ordnertyp	Liste von Xjustiz spezfischen Ordnertypen	N/A	String	Nein
eip:folderNewDigitalFile	Digital File Kennzeichnung	Kennzeichnung ob Digital File noch nicht geöffnet wurde. Default ist false.	N/A	Boolean	Nein
eip:folderGrantExternalAccess	Kennzeichnung externer Zugriff	Kennzeichnung ob der Ordner außerhalb des Gerichts zugreifbar ist. Default ist true.	N/A	Boolean	Nein
eip:folderDigitalFileName	Aktenzeichen	Aktenzeichen der e-Akte	N/A	String	Nein
eip:folderDigitalFileState	Aktenstatus	Neues Metadatum „status“ auf Ebene der Akte mit einer Werteliste	AKTENSTATUS	String	Nein
eip:folderLawsuitState	Lawsuitstatus	Neues Metadatum „status“ auf Ebene der Akte	N/A	String	Nein
eip:folderInformationBlock	Auskunftssperre	Ein Bereich mit Auskunftssperre darf externen Einsichtsnehmern o.ä. nicht zugänglich gemacht werden.	true/false(null)	Boolean	Nein
eip:folderRestrictedAccess	Sperrkennzeichen	Boolean-Flag erweitert, welches kennzeichnet, ob es sich bei dem alle Dokumente im Ordner um ein gesperrtes oder nicht gesperrtes.	true/false(null)	Boolean	Nein

Abbildung 10-20: DMS Attribute Ordner

Attributname	Kurzbeschreibung	Beschreibung	Werteliste	Attributtyp	Erforderlich
cmis:contentStreamFileName	Content Stream Dateiname	Der Dateiname für das Streaming des Inhalts	N/A	String	Ja
cmis:contentStreamId	Content Stream Id	Die Id des Streams	N/A	String	Ja
cmis:contentStreamLength	Content Stream Länge	Die Größe der Datei / des Dokuments	N/A	Long	Ja
cmis:contentStreamMimeType	Content Stream Mime Type	Der Mime-Type des Dokuments	Offizielle Mime Typen	String	Ja
cmis:createdBy	Ersteller	Der Ersteller des Objekts im DMS	N/A	String	Ja
cmis:creationDate	Erstelldatum	Das Erzeugungsdatum dieses Objekts im DMS	N/A	Date	Ja
cmis:lastModificationDate	Änderungsdatum	Das Datum der letzten Änderung des Objekts	N/A	Date	Ja
cmis:lastModifiedBy	Änderungsbenutzer	Der Benutzer, der die letzte Änderung durchgeführt hat	N/A	String	Ja
cmis:name	Name	Name des Objekts	N/A	String	Ja
cmis:objectId	Objekt Id	Eindeutige Objekt-Id	N/A	String	Ja
cmis:objectTypeId	Objekt Typ	Id des Objekt-Typs	OBJECT_TYPE_ID	String	Ja
cmis:versionSeriesId	Version Id	Die Versionskennzeichnung des Objekts	N/A	String	Ja
eip:docCreator	Dokumentenersteller	Der Erzeuger des Dokuments aus Sicht der Fachanwendung	N/A	String	Ja
eip:docOwner	Dokumenteneigentümer	Benutzerkennung, welche das Dokument verändern darf	N/A	String	Nein
eip:docGroupOwner	Benutzergruppe	Die Benutzergruppe, der das Objekt zugeordnet ist	N/A	String	Nein
eip:docContentChecksum	Dokumentinhalt Checksumm	MD5-Prüfsumme für die Dateien (im Voraus berechneten). Ein Benutzer kann die Prüfsumme der heruntergeladenen Datei zu vergleichen	N/A	String	Nein
eip:docExtension	Dokumentextension	Dateiendung des Dokuments	N/A	String	Ja
eip:annotationDocId	Verknüpftes Dokument Id	Die Id des verknüpften Dokuments	N/A	String	Nein
eip:annotationKind	Annotationstyp	Art der Annotation. Default ist persoenlich.	ANNOTATIONART	String	Nein

Abbildung 10-21: DMS Attribute Annotationen

Attributname	Kurzbeschreibung	Beschreibung	Werteliste	Attributtyp	Erforderlich
cmis:contentStreamFileName	Content Stream Dateiname	Der Dateiname für das Streaming des Inhalts	N/A	String	Ja
cmis:contentStreamId	Content Stream Id	Die Id des Streams	N/A	String	Ja
cmis:contentStreamLength	Content Stream Länge	Die Größe der Datei / des Dokuments	N/A	Long	Ja
cmis:contentStreamMimeType	Content Stream Mime Type	Der Mime-Type des Dokuments	Offifzielle Mime Typen	String	Ja
cmis:createdBy	Ersteller	Der Ersteller des Objekts im DMS	N/A	String	Ja
cmis:creationDate	Erstelldatum	Das Erzeugungsdatum dieses Objekts im DMS	N/A	Date	Ja
cmis:lastModificationDate	Änderungsdatum	Das Datum der letzten Änderung des Objekts	N/A	Date	Ja
cmis:lastModifiedBy	Änderungsbenutzer	Der Benutzer, der die letzte Änderung durchgeführt hat	N/A	String	Ja
cmis:name	Name	Name des Objekts	N/A	String	Ja
cmis:objectId	Objekt Id	Eindeutige Objekt-Id	N/A	String	Ja
cmis:objectTypeId	Objekt Typ	Id des Objekt-Typs	OBJECT_TYPE_ID	String	Ja
cmis:versionSeriesId	Version Id	Die Versionskennzeichnung des Objekts	N/A	String	Ja
eip:docCreator	Dokumentenersteller	Der Erzeuger des Dokuments aus Sicht der Fachanwendung	N/A	String	Ja
eip:docOwner	Dokumenteneigentümer	Benutzerkennung, welche das Dokument verändern darf	N/A	String	Nein
eip:docGroupOwner	Benutzergruppe	Die Benutzergruppe, der das Objekt zugeordnet ist	N/A	String	Nein
eip:docContentChecksum	Dokumentinhalt Checksumm	MD5-Prüfsumme für die Dateien (im Voraus berechneten). Ein Benutzer kann die Prüfsumme der heruntergeladenen Datei zu vergleichen	N/A	String	Nein
eip:docExtension	Dokumentextension	Dateiendung des Dokuments	N/A	String	Ja
eip:docDigitalFileId	Eindeutige Verfahrenskennung	Die eindeutige Verfahrenskennung, der das Dokument zugeordnet ist Dies stellt nicht das Aktenzeichen dar, da dieses nicht eindeutig im Fachverfahren ist.	N/A	String	Nein
eip:docDigitalFileName	Verfahrensname	Das Aktenzeichen in lesbarer Form für die Anzeige	N/A	String	Nein
eip:docName	Dokumentenname	Anzeigename des Dokuments für die Darstellung im Aktenbaum	N/A	String	Ja
eip:docOriginalName	Originalname des Dokuments	Der unveränderliche Originalname des Dokuments bei der Erstellung	N/A	String	Ja
eip:docCategory	Kategorie	Kommaseparierte Liste mit dem Dokument zugeordnete Kategorien, frei erweiterbar, somit keine feste Werteliste	EAKTE_CATEGORY	List<String>	Nein
eip:docConversionState	Konvertierungsstatus	Der Konverierungsstatus des Dokuments aufgrund einer möglichen Konvertierung nach PDF/A	KONVERTIERUNGSSTATUS	Long	Ja
eip:docOriginalDocumentId	Originaldokument ID	Referenzierte Dokument-ID, die aktuelle Dokument zu erstellen verwendet wurde	N/A	String	Nein
eip:docSourceSystemDocumentId	ID eines Dokuments aus einem Quellsyste	Dokumente, die aus einem anderen System kommen können dort eine ID besitzen. Diese kann hier abgelegt werden, um das ursprüngliche Dokument im Quell-System identifizieren zu können. z.B. externe Referenzkennung auf ein Entwurfsdokument (forumSTAR Text)	N/A	String	Nein
ekp:docAttachmentId	EKP Attachment ID	Eindeutige Kennung eines Anhangsdokuments in der eKP	N/A	Long	Nein
ekp:docMessageId	EKP Message ID	Eindeutige Nachrichtenkennung in der eKP, aus der dieses Dokument stammt	N/A	Long	Nein
ekp:docMessageType	EKP Message Typ	EKP Message Typ	EKPMESSAGETYP	String	Nein
eip:docErvAttachmentId	ERV Attachment ID	Eindeutige Kennung eines Anhangsdokuments im Fachverfahren	N/A	Long	Nein
eip:docErvMessageId	ERV Message ID	Eindeutige Kennung einer Nachricht im Fachverfahren	N/A	Long	Nein
eip:docIsNew	Neumarkierung	Markierung ob ein Dokument als neu angezeigt werden soll	true/false(null)	Boolean	Ja
eip:docPagesCount	Seitenzahl	Anzahl an Seiten in diesem Dokument	N/A	Long	Nein
eip:docPagesInfo	Seiteinfo	Seitenummerinformation fuer jede Seite	N/A	List<String>	Nein
eip:docPagingMethod	Paginierungsart	Gewählte Paginierungsart für dieses Dokument	PAGINIERUNGSART	String	Nein
eip:docPagingInformations	Seiteinfo-Historie	Seitenummerinformation-Historie fuer jede Seite	N/A	List<String>	Nein
eip:docSignatureState	Signaturstatus	Status der Signatur an diesem Dokument	SIGNATURSTATUS	String	Nein
eip:docType	Dokument Typ	Dokument Typ	DOKUMENTTYP	List<String>	Nein
eip:docMoveToAkteTimestamp	Verschiebungsdatum in die Akte	Zeitstempel des Datums, wann das Dokument in ie Akte verschoben ist	N/A	Date	Nein
eip:docBundleCreationDate	Gruppeerstellungsdatum	Erstellungsdatum einiger Dokumentengruppe . Alle Dokumente zu dieser Gruppe gehören, haben den gleichen Wert.	N/A	Date	Nein
eip:docDeletionDate	Löschdatum	Löschdatum des Dokuments	N/A	Date	Nein
eip:docArchivLongestRetained	Längste aufzuhebende Schriftstücke	Markierung ob ein Dokument als längste aufzuhebend werden soll	true/false(null)	Boolean	Nein
eip:docSecurityPassword	Sicherheitspasswort	Sicherheitspasswort, zum Öffnen der PDF-Datei	N/A	String	Nein
eurekafach:docAddress	Adresse des Absenders oder Empfängers c	Adresse des Absenders oder Empfängers des Dokuments. Attribut wird von Eeureka-Fach-Anwendung verwendet.	N/A	String	Nein
eurekafach:docType	Dokument typ von Eureka-Fach	Dokument typ von Eureka-Fach	N/A	List<String>	Nein
eip:docDigitalFileId	ID der e-Akte	Eindeutige ID der e-Akte, der das Dokument zugeordnet ist.	N/A	String	Nein
eip:docBundleRestoreFromTrashDate	Wiederherstellungszeitpunkt	Datum und Zeit wann das Dokument aus dem Abfalleimer wiederhergestellt wurde.	N/A	Date	Nein
eip:docGrantExternalAccess	Akteneinsicht	Dokument unterliegt der Akteneinsicht	N/A	Boolean	Nein
eip:docInformationBlock	Auskunftssperre	Ein Bereich mit Auskunftssperre darf externen Einsichtsnehmern o.ä. nicht zugänglich gemacht werden.	true/false(null)	Boolean	Nein
eip:docSortDate	Sortierdatum	Als Sortierdatum welches vom Fachverfahren beim Speichern der Dokumente gesetzt werden kann.	N/A	Date	Nein

Abbildung 10-22: DMS Attribute Dokument

Attributname	Kurzbeschreibung	Beschreibung	Werteliste	Attributtyp	Erforderlich
cmis:contentStreamFileName	Content Stream Dateiname	Der Dateiname für das Streaming des Inhalts	N/A	String	Ja
cmis:contentStreamId	Content Stream Id	Die Id des Streams	N/A	String	Ja
cmis:contentStreamLength	Content Stream Länge	Die Größe der Datei / des Dokuments	N/A	Long	Ja
cmis:contentStreamMimeType	Content Stream Mime Type	Der Mime-Type des Dokuments	Offifzielle Mime Typen	String	Ja
cmis:createdBy	Ersteller	Der Ersteller des Objekts im DMS	N/A	String	Ja
cmis:creationDate	Erstelldatum	Das Erzeugungsdatum dieses Objekts im DMS	N/A	Date	Ja
cmis:lastModificationDate	Änderungsdatum	Das Datum der letzten Änderung des Objekts	N/A	Date	Ja
cmis:lastModifiedBy	Änderungsbenutzer	Der Benutzer, der die letzte Änderung durchgeführt hat	N/A	String	Ja
cmis:name	Name	Name des Objekts	N/A	String	Ja
cmis:objectId	Objekt Id	Eindeutige Objekt-Id	N/A	String	Ja
cmis:objectTypeId	Objekt Typ	Id des Objekt-Typs	OBJECT_TYPE_ID	String	Ja
cmis:versionSeriesId	Version Id	Die Versionskennzeichnung des Objekts	N/A	String	Ja
eip:docCreator	Dokumentenersteller	Der Erzeuger des Dokuments aus Sicht der Fachanwendung	N/A	String	Ja
eip:docOwner	Dokumenteneigentümer	Benutzerkennung, welche das Dokument verändern darf	N/A	String	Nein
eip:docGroupOwner	Benutzergruppe	Die Benutzergruppe, der das Objekt zugeordnet ist	N/A	String	Nein
eip:docContentChecksum	Dokumentinhalt Checksumm	MD5-Prüfsumme für die Dateien (im Voraus berechneten). Ein Benutzer kann die Prüfsumme der heruntergeladenen Datei zu vergleichen	N/A	String	Nein
eip:docExtension	Dokumentextension	Dateiendung des Dokuments	N/A	String	Ja
eip:signatureRefDocId	Verknüpftes Dokument Id	Die Id des verknüpften Dokuments	N/A	String	Ja
eip:signatureFormatType	Signaturtyp	Art der Signatur	SIGNATUREFORMATTYPE	Long	Ja

Abbildung 10-23: DMS Attribute Signatur

Attributname	Kurzbeschreibung	Beschreibung	Werteliste	Attributtyp	Erforderlich
cmis:contentStreamFileName	Content Stream Dateiname	Der Dateiname für das Streaming des Inhalts	N/A	String	Ja
cmis:contentStreamId	Content Stream Id	Die Id des Streams	N/A	String	Ja
cmis:contentStreamLength	Content Stream Länge	Die Größe der Datei / des Dokuments	N/A	Long	Ja
cmis:contentStreamMimeType	Content Stream Mime Type	Der Mime-Type des Dokuments	Offifzielle Mime Typen	String	Ja
cmis:createdBy	Ersteller	Der Ersteller des Objekts im DMS	N/A	String	Ja
cmis:creationDate	Erstelldatum	Das Erzeugungsdatum dieses Objekts im DMS	N/A	Date	Ja
cmis:lastModificationDate	Änderungsdatum	Das Datum der letzten Änderung des Objekts	N/A	Date	Ja
cmis:lastModifiedBy	Änderungsbenutzer	Der Benutzer, der die letzte Änderung durchgeführt hat	N/A	String	Ja
cmis:name	Name	Name des Objekts	N/A	String	Ja
cmis:objectId	Objekt Id	Eindeutige Objekt-Id	N/A	String	Ja
cmis:objectTypeId	Objekt Typ	Id des Objekt-Typs	OBJECT_TYPE_ID	String	Ja
cmis:versionSeriesId	Version Id	Die Versionskennzeichnung des Objekts	N/A	String	Ja
eip:docCreator	Dokumentenersteller	Der Erzeuger des Dokuments aus Sicht der Fachanwendung	N/A	String	Ja
eip:docOwner	Dokumenteneigentümer	Benutzerkennung, welche das Dokument verändern darf	N/A	String	Nein
eip:docGroupOwner	Benutzergruppe	Die Benutzergruppe, der das Objekt zugeordnet ist	N/A	String	Nein
eip:docContentChecksum	Dokumentinhalt Checksumm	MD5-Prüfsumme für die Dateien (im Voraus berechneten). Ein Benutzer kann die Prüfsumme der heruntergeladenen Datei zu vergleichen	N/A	String	Nein
eip:docExtension	Dokumentextension	Dateiendung des Dokuments	N/A	String	Ja
eip:docDigitalFileId	Eindeutige Verfahrenskennung	Die eindeutige Verfahrenskennung, der das Dokument zugeordnet ist Dies stellt nicht das Aktenzeichen dar, da dieses nicht eindeutig im Fachverfahren ist.	N/A	String	Nein
eip:docDigitalFileName	Verfahrensname	Das Aktenzeichen in lesbarer Form für die Anzeige	N/A	String	Nein
eip:docName	Dokumentenname	Anzeigename des Dokuments für die Darstellung im Aktenbaum	N/A	String	Ja
eip:docOriginalName	Originalname des Dokuments	Der unveränderliche Originalname des Dokuments bei der Erstellung	N/A	String	Ja
eip:coverDocSourceSystemDocumentId	Schriftstück Id	Externe Referenzkennung auf ein Entwurfsdokument (forumSTAR Text)	N/A	String	Nein

Abbildung 10-24: DMS Attribute Deckblatt

Attributname	Kurzbeschreibung	Beschreibung	Werteliste	Attributtyp	Erforderlich
cmis:contentStreamFileName	Content Stream Dateiname	Der Dateiname für das Streaming des Inhalts	N/A	String	Ja
cmis:contentStreamId	Content Stream Id	Die Id des Streams	N/A	String	Ja
cmis:contentStreamLength	Content Stream Länge	Die Größe der Datei / des Dokuments	N/A	Long	Ja
cmis:contentStreamMimeType	Content Stream Mime Type	Der Mime-Type des Dokuments	Offifzielle Mime Typen	String	Ja
cmis:createdBy	Ersteller	Der Ersteller des Objekts im DMS	N/A	String	Ja
cmis:creationDate	Erstelldatum	Das Erzeugungsdatum dieses Objekts im DMS	N/A	Date	Ja
cmis:lastModificationDate	Änderungsdatum	Das Datum der letzten Änderung des Objekts	N/A	Date	Ja
cmis:lastModifiedBy	Änderungsbenutzer	Der Benutzer, der die letzte Änderung durchgeführt hat	N/A	String	Ja
cmis:name	Name	Name des Objekts	N/A	String	Ja
cmis:objectId	Objekt Id	Eindeutige Objekt-Id	N/A	String	Ja
cmis:objectTypeId	Objekt Typ	Id des Objekt-Typs	OBJECT_TYPE_ID	String	Ja
cmis:versionSeriesId	Version Id	Die Versionskennzeichnung des Objekts	N/A	String	Ja
eip:docCreator	Dokumentenersteller	Der Erzeuger des Dokuments aus Sicht der Fachanwendung	N/A	String	Ja
eip:docOwner	Dokumenteneigentümer	Benutzerkennung, welche das Dokument verändern darf	N/A	String	Nein
eip:docGroupOwner	Benutzergruppe	Die Benutzergruppe, der das Objekt zugeordnet ist	N/A	String	Nein
eip:docContentChecksum	Dokumentinhalt Checksumm	MD5-Prüfsumme für die Dateien (im Voraus berechneten). Ein Benutzer kann die Prüfsumme der heruntergeladenen Datei zu vergleichen	N/A	String	Nein
eip:docExtension	Dokumentextension	Dateiendung des Dokuments	N/A	String	Ja
eip:docDigitalFileId	Eindeutige Verfahrenskennung	Die eindeutige Verfahrenskennung, der das Dokument zugeordnet ist Dies stellt nicht das Aktenzeichen dar, da dieses nicht eindeutig im Fachverfahren ist.	N/A	String	Nein
eip:docDigitalFileName	Verfahrensname	Das Aktenzeichen in lesbarer Form für die Anzeige	N/A	String	Nein
eip:docName	Dokumentenname	Anzeigename des Dokuments für die Darstellung im Aktenbaum	N/A	String	Ja
eip:docOriginalName	Originalname des Dokuments	Der unveränderliche Originalname des Dokuments bei der Erstellung	N/A	String	Ja

Abbildung 10-25: DMS Attribute Notizzettel

Attributname	Kurzbeschreibung	Beschreibung	Werteliste	Attributtyp	Erforderlich
cmis:contentStreamFileName	Content Stream Dateiname	Der Dateiname für das Streaming des Inhalts	N/A	String	Ja
cmis:contentStreamId	Content Stream Id	Die Id des Streams	N/A	String	Ja
cmis:contentStreamLength	Content Stream Länge	Die Größe der Datei / des Dokuments	N/A	Long	Ja
cmis:contentStreamMimeType	Content Stream Mime Type	Der Mime-Type des Dokuments	Offifzielle Mime Typen	String	Ja
cmis:createdBy	Ersteller	Der Ersteller des Objekts im DMS	N/A	String	Ja
cmis:creationDate	Erstelldatum	Das Erzeugungsdatum dieses Objekts im DMS	N/A	Date	Ja
cmis:lastModificationDate	Änderungsdatum	Das Datum der letzten Änderung des Objekts	N/A	Date	Ja
cmis:lastModifiedBy	Änderungsbenutzer	Der Benutzer, der die letzte Änderung durchgeführt hat	N/A	String	Ja
cmis:name	Name	Name des Objekts	N/A	String	Ja
cmis:objectId	Objekt Id	Eindeutige Objekt-Id	N/A	String	Ja
cmis:objectTypeId	Objekt Typ	Id des Objekt-Typs	OBJECT_TYPE_ID	String	Ja
cmis:versionSeriesId	Version Id	Die Versionskennzeichnung des Objekts	N/A	String	Ja
eip:docCreator	Dokumentenersteller	Der Erzeuger des Dokuments aus Sicht der Fachanwendung	N/A	String	Ja
eip:docOwner	Dokumenteneigentümer	Benutzerkennung, welche das Dokument verändern darf	N/A	String	Nein
eip:docGroupOwner	Benutzergruppe	Die Benutzergruppe, der das Objekt zugeordnet ist	N/A	String	Nein
eip:docContentChecksum	Dokumentinhalt Checksumm	MD5-Prüfsumme für die Dateien (im Voraus berechneten). Ein Benutzer kann die Prüfsumme der heruntergeladenen Datei zu vergleichen	N/A	String	Nein
eip:docExtension	Dokumentextension	Dateiendung des Dokuments	N/A	String	Ja
eip:docDigitalFileId	Eindeutige Verfahrenskennung	Die eindeutige Verfahrenskennung, der das Dokument zugeordnet ist Dies stellt nicht das Aktenzeichen dar, da dieses nicht eindeutig im Fachverfahren ist.	N/A	String	Nein
eip:docDigitalFileName	Verfahrensname	Das Aktenzeichen in lesbarer Form für die Anzeige	N/A	String	Nein
eip:docName	Dokumentenname	Anzeigename des Dokuments für die Darstellung im Aktenbaum	N/A	String	Ja
eip:docOriginalName	Originalname des Dokuments	Der unveränderliche Originalname des Dokuments bei der Erstellung	N/A	String	Ja

Abbildung 10-26: DMS Attribute Signaturprüfprotokoll

Attributname	Kurzbeschreibung	Beschreibung	Werteliste	Attributtyp	Erforderlich
cmis:contentStreamFileName	Content Stream Dateiname	Der Dateiname für das Streaming des Inhalts	N/A	String	Ja
cmis:contentStreamId	Content Stream Id	Die Id des Streams	N/A	String	Ja
cmis:contentStreamLength	Content Stream Länge	Die Größe der Datei / des Dokuments	N/A	Long	Ja
cmis:contentStreamMimeType	Content Stream Mime Type	Der Mime-Type des Dokuments	Offifzielle Mime Typen	String	Ja
cmis:createdBy	Ersteller	Der Ersteller des Objekts im DMS	N/A	String	Ja
cmis:creationDate	Erstelldatum	Das Erzeugungsdatum dieses Objekts im DMS	N/A	Date	Ja
cmis:lastModificationDate	Änderungsdatum	Das Datum der letzten Änderung des Objekts	N/A	Date	Ja
cmis:lastModifiedBy	Änderungsbenutzer	Der Benutzer, der die letzte Änderung durchgeführt hat	N/A	String	Ja
cmis:name	Name	Name des Objekts	N/A	String	Ja
cmis:objectId	Objekt Id	Eindeutige Objekt-Id	N/A	String	Ja
cmis:objectTypeId	Objekt Typ	Id des Objekt-Typs	OBJECT_TYPE_ID	String	Ja
cmis:versionSeriesId	Version Id	Die Versionskennzeichnung des Objekts	N/A	String	Ja
eip:docCreator	Dokumentenersteller	Der Erzeuger des Dokuments aus Sicht der Fachanwendung	N/A	String	Ja
eip:docOwner	Dokumenteneigentümer	Benutzerkennung, welche das Dokument verändern darf	N/A	String	Nein
eip:docGroupOwner	Benutzergruppe	Die Benutzergruppe, der das Objekt zugeordnet ist	N/A	String	Nein
eip:docContentChecksum	Dokumentinhalt Checksumm	MD5-Prüfsumme für die Dateien (im Voraus berechneten). Ein Benutzer kann die Prüfsumme der heruntergeladenen Datei zu vergleichen	N/A	String	Nein
eip:docExtension	Dokumentextension	Dateiendung des Dokuments	N/A	String	Ja
eip:docDigitalFileId	Eindeutige Verfahrenskennung	Die eindeutige Verfahrenskennung, der das Dokument zugeordnet ist Dies stellt nicht das Aktenzeichen dar, da dieses nicht eindeutig im Fachverfahren ist.	N/A	String	Nein
eip:docDigitalFileName	Verfahrensname	Das Aktenzeichen in lesbarer Form für die Anzeige	N/A	String	Nein
eip:docName	Dokumentenname	Anzeigename des Dokuments für die Darstellung im Aktenbaum	N/A	String	Ja
eip:docOriginalName	Originalname des Dokuments	Der unveränderliche Originalname des Dokuments bei der Erstellung	N/A	String	Ja

Abbildung 10-27: DMS Attribute Transfervermerk

Attributname	Kurzbeschreibung	Beschreibung	Werteliste	Attributtyp	Erforderlich
cmis:createdBy	Ersteller	Der Ersteller des Objekts im DMS	N/A	String	Ja
cmis:creationDate	Erstelldatum	Das Erzeugungsdatum dieses Objekts im DMS	N/A	Date	Ja
cmis:lastModificationDate	Änderungsdatum	Das Datum der letzten Änderung des Objekts	N/A	Date	Ja
cmis:lastModifiedBy	Änderungsbenutzer	Der Benutzer, der die letzte Änderung durchgeführt hat	N/A	String	Ja
cmis:name	Name	Name des Objekts	N/A	String	Ja
cmis:objectId	Objekt Id	Eindeutige Objekt-Id	N/A	String	Ja
cmis:objectTypeId	Objekt Typ	Id des Objekt-Typs	OBJECT_TYPE_ID	String	Ja
cmis:parentId	Vater Objekt Id	Die Objekt Id des Vaterelements in der hierarschischen Struktur	N/A	String	Ja
cmis:path	Pfad	Der vollqualifizierte Pfad zu dem Ordner	N/A	String	Ja
eip:folderLawsuitId	Eindeutige Verfahrenskennung	Die eindeutige Verfahrenskennung, der der Ordner zugeordnet ist Dies stellt nicht das Aktenzeichen dar, da dieses nicht eindeutig im Fachverfahren ist.	N/A	String	Nein
eip:folderDigitalFileId	ID der e-Akte	Eindeutige ID der e-Akte. Muss nicht der Verfahrens-ID entsprechen. Bei der Grundakte z.B. gibt es keine direkte Zuordnung von Verfahren / Fällen zu Akten	N/A	String	Nein
eip:folderRelatedDigitalFileId	ID der Ziel e-Akte	Eindeutige ID der Ziel e-Akte aus der Beziehung	N/A	String	Ja

Abbildung 10-28: DMS Attribute Beiaktenreferenz

Attributname	Kurzbeschreibung	Beschreibung	Werteliste	Attributtyp	Erforderlich
cmis:contentStreamFileName	Content Stream Dateiname	Der Dateiname für das Streaming des Inhalts	N/A	String	Ja
cmis:contentStreamId	Content Stream Id	Die Id des Streams	N/A	String	Ja
cmis:contentStreamLength	Content Stream Länge	Die Größe der Datei / des Dokuments	N/A	Long	Ja
cmis:contentStreamMimeType	Content Stream Mime Type	Der Mime-Type des Dokuments	Offifzielle Mime Typen	String	Ja
cmis:createdBy	Ersteller	Der Ersteller des Objekts im DMS	N/A	String	Ja
cmis:creationDate	Erstelldatum	Das Erzeugungsdatum dieses Objekts im DMS	N/A	Date	Ja
cmis:lastModificationDate	Änderungsdatum	Das Datum der letzten Änderung des Objekts	N/A	Date	Ja
cmis:lastModifiedBy	Änderungsbenutzer	Der Benutzer, der die letzte Änderung durchgeführt hat	N/A	String	Ja
cmis:name	Name	Name des Objekts	N/A	String	Ja
cmis:objectId	Objekt Id	Eindeutige Objekt-Id	N/A	String	Ja
cmis:objectTypeId	Objekt Typ	Id des Objekt-Typs	OBJECT_TYPE_ID	String	Ja
dabag:aufnahmeDatum	Aufnahmedatum des Dokuments	Eigenschaft Aufnahmedatum des Dokuments	N/A	Date	Nein
dabag:docOrdnungsnummer	Ordnungsnummer des Ordners	Eigenschaft Ordnungsnummer des Ordners	N/A	String	Nein
dabag:docVerweisAteilId	Verweis Aktenteil id	Eigenschaft Verweis Aktenteil id	N/A	String	Nein
dabag:dokumentIdFv	Dokument ID FV	Eigenschaft Dokument ID FV	N/A	String	Nein
dabag:dokumentenart	Dokumentenart	Eigenschaft Dokumentenart	N/A	String	Nein
dabag:dokumentenherkunft	Dokumentenherkunft	Eigenschaft Dokumentenherkunft	N/A	String	Nein
dabag:eingangszeitpunkt	Eingangszeitpunkt	Eigenschaft Eingangszeitpunkt	N/A	Date	Nein
dabag:empfaenger	Empfängerdaten	Eigenschaft Empfängerdaten	N/A	String	Nein
dabag:erstellungszeitpunkt	Erstellungszeitpunkt	Eigenschaft Erstellungszeitpunkt	N/A	Date	Nein
dabag:fremdgz	Fremd-GZ	Eigenschaft Fremd-GZ	N/A	String	Nein
dabag:name	Original-Dateiname	Eigenschaft Original-Dateiname	N/A	String	Nein
dabag:signaturdatei	Signaturdatei	Eigenschaft Signaturdatei	N/A	String	Nein
dabag:urkundennummer	Urkundennummer	Eigenschaft Urkundennummer	N/A	String	Nein
dabag:versandtAm	Versandt am	Eigenschaft Versandt am	N/A	Date	Nein
dabag:vpsMsgId	VPS-MSGID	Eigenschaft VPS-MSGID	N/A	String	Nein
dabag:zustellungsDatum	Zustelldatum	Eigenschaft Zustelldatum	N/A	Date	Nein
eip:docContentChecksum	Dokumentinhalt Checksumm	MD5-Prüfsumme für die Dateien (im Voraus berechneten). Ein Benutzer kann die Prüfsumme der heruntergeladenen Datei zu vergleichen	N/A	String	Nein
eip:docCreator	Dokumentenersteller	Der Erzeuger des Dokuments aus Sicht der Fachanwendung	N/A	String	Nein
eip:docDigitalFileId	Eindeutige Verfahrenskennung	Die eindeutige Verfahrenskennung, der das Dokument zugeordnet ist Dies stellt nicht das Aktenzeichen dar, da dieses nicht eindeutig im Fachverfahren ist.	N/A	String	Nein
eip:docDigitalFileName	Verfahrensname	Das Aktenzeichen in lesbarer Form für die Anzeige	N/A	String	Nein
eip:docExtension	Dokumentextension	Dateiendung des Dokuments	N/A	String	Nein
eip:docGroupOwner	Benutzergruppe	Die Benutzergruppe, der das Objekt zugeordnet ist	N/A	String	Nein
eip:docLawsuitId	Id vom Gerichtsverfahren	Die Id von dem Gerichtsverfahren	N/A	String	Nein
eip:docLinkType	Art der Verknüpfung	Art der Verknüpfung, abhängig vom Typ des Zielobjekts	DOCUMENT	String	Ja
eip:docName	Dokumentenname	Anzeigename des Dokuments für die Darstellung im Aktenbaum	N/A	String	Ja
eip:docOriginalName	Originalname des Dokuments	Der unveränderliche Originalname des Dokuments bei der Erstellung	N/A	String	Nein
eip:docOwner	Dokumenteneigentümer	Benutzerkennung, welche das Dokument verändern darf	N/A	String	Nein
eip:documentRestrictions	Einschränkungsliste	Liste der Einschränkungen für das Dokument	noDelete, noRename, noReorder	String	Nein

Abbildung 10-29: DMS Attribute Verweisungsvermerk Dokument

Attributname	Kurzbeschreibung	Beschreibung	Werteliste	Attributtyp	Erforderlich
cmis:createdBy	Ersteller	Der Ersteller des Objekts im DMS	N/A	String	Ja
cmis:creationDate	Erstelldatum	Das Erzeugungsdatum dieses Objekts im DMS	N/A	Date	Ja
cmis:lastModificationDate	Änderungsdatum	Das Datum der letzten Änderung des Objekts	N/A	Date	Ja
cmis:lastModifiedBy	Änderungsbenutzer	Der Benutzer, der die letzte Änderung durchgeführt hat	N/A	String	Ja
cmis:name	Name	Name des Objekts	N/A	String	Ja
cmis:objectId	Objekt Id	Eindeutige Objekt-Id	N/A	String	Ja
cmis:objectTypeId	Objekt Typ	Id des Objekt-Typs	OBJECT_TYPE_ID	String	Ja
cmis:parentId	Vater Objekt Id	Die Objekt Id des Vaterelements in der hierarschischen Struktur	N/A	String	Ja
cmis:path	Pfad	Der vollqualifizierte Pfad zu dem Ordner	N/A	String	Ja
dabag:abteilung	Abteilung in dabag	oder für Unterordner: Ordnungsnummer oder Vorgangsnummer im Anzeigeformat	N/A	String	Nein
dabag:anzeigename	Anzeigename in dabag	Für Akten: Aktenzeichen als Name für Anzeige im Aktenbaum des Aktenviewers (Anzeigename) oder Firmenname für Handelsregisterakte oder für Unterordner: Ordnungsnummer oder Vorgangsnummer im Anzeigeformat	N/A	String	Nein
dabag:bezirk	Bezirk in dabag	Zur Bildung des Aktenzeichens bei Grundakten	N/A	String	Nein
dabag:blattnummer	Blattnummer in dabag	Zur Bildung des Aktenzeichens bei Grundakten	N/A	String	Nein
dabag:eingangsdatumVorgang	Eingangsdatum Vorgang in dabag	Datum mit Uhrzeit des Eingangs des ersten Antrags zum Vorgang	N/A	Date	Nein
dabag:eingangsordner	Eingangs Ordner in dabag	Beschreibt, ob es sich um einen Eingangsordner handelt; evtl. auch Ermittlung über Vorlage	N/A	Boolean	Nein
dabag:entwurfsordner	Entwurfs Ordner in dabag	Beschreibt, ob es sich um einen Entwurfsordner handelt; evtl. auch Ermittlung über Vorlage	N/A	Boolean	Nein
dabag:folderOrdnungsnummer	Folder Ordnungsnummer	Laufende Nummer / Ordnungsnummer in dabag; Vorgangsnummer in AuRegis	N/A	String	Nein
dabag:folderVerweisAteilId	Verweis Aktenteil id	Eigenschaft Verweis Aktenteil id		String	Nein
dabag:gericht	Gericht in dabag	Zur Bildung des Aktenzeichens bei Grundakten	N/A	String	Nein
dabag:jahr	Jahr in dabag	Zur Bildung des AR-Aktenzeichens	N/A	String	Nein
dabag:laufendeNummer	Laufende Nummer in dabag	Zur Bildung des AR-Aktenzeichens	N/A	String	Nein
dabag:papierakteVorhanden	Papierakte Vorhanden	Falls null, kann davon ausgegangen werden, dass es sich um eine Papierakte oder Akte aus SolumStar handelt. Für dabag und AuRegis Standardwert "Unbekannt". Andere Werte sind ja und nein. Bei komplett neuen Akten können die Systemfassaden von dabag und AuRegis „nein“ vergeben.	N/A	Boolean	Nein
dabag:vorgangsId	Vorgangs Id in dabag	Vorgangs ID aus dem FV falls von dort aus angelegt	N/A	String	Nein
eip:folderDigitalFileId	ID der e-Akte	Eindeutige ID der e-Akte. Muss nicht der Verfahrens-ID entsprechen. Bei der Grundakte z.B. gibt es keine direkte Zuordnung von Verfahren / Fällen zu Akten	N/A	String	Nein
eip:folderLawsuitId	Eindeutige Verfahrenskennung	Die eindeutige Verfahrenskennung, der der Ordner zugeordnet ist Dies stellt nicht das Aktenzeichen dar, da dieses nicht eindeutig im Fachverfahren ist.	N/A	String	Nein
eip:folderLinkType	Art der Verknüpfung	Art der Verknüpfung, abhängig vom Zielobjekt	DIGITAL_FILE, FOLDER	String	Ja
eip:folderName	Ordnername	Anzeigename des Ordners für die Darstellung im Aktenbaum	N/A	String	Nein
eip:folderRestrictions	Einschränkungsliste	Für einen Ordner können diverse Einschränkungen gelten. Diese werden hiermit benannt. So kann ein Ordner z. B. vor dem Löschen geschützt sein.	noDelete, noRename, noReorder	String	Nein
eip:folderDigitalFileName	Aktenzeichen	Aktenzeichen der e-Akte	N/A	String	Nein

Abbildung 10-30: DMS Attribute Verweisungsvermerk Ordner / Akte

Details zu dem Modell werden in dem Fachklassenmodell beschrieben.

Die Relationen zwischen den verschiedenen Typen lässt sich folgendermaßen darstellen.

Quelleobjekt	Fachliche Richtung	Richtung in Alfres	Zielobjekt
cm:referencing	n to n	n to n	cm:content
eip:eipDocument	n to 1	n to n	eip:eipDocument
eip:eipDocument	1 to n	n to n	eip:eipDocument
eip:eipDocument	1 to n	n to n	eip:eipAnnotation
eip:eipFolder	n to 1	n to 1	eip:eipFolder
eip:eipFolder	1 to n	n to n	eip:eipFolder
eip:eipDocument	1 to 1	n to n	eip:eipSignatureVerificationDocument
eip:eipDocument	1 to n	n to n	eip:eipDocument
eip:eipFolder	n to 1	n to 1	eip:eipFolder
eip:eipReferenceFolder	n to 1	n to n	eip:eipFolder
eip:eipReferenceFolder	n to 1	n to n	eip:eipFolder
eip:eipDocument	n to n	n to n	eip:eipTransferNoteDocument
eip:eipBaseDocument	1 to n	n to n	eip:eipBaseDocument

Abbildung 10-31: DMS-Abbildung der Referenzen

.

11 NICHT-FUNKTIONALE ANFORDERUNGEN

11.1 Softwarequalität

Softwarequalität bezeichnet die Gesamtheit der Merkmale und Merkmalswerte eines Softwareproduktes, die sich auf dessen Eignung beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen.

Um Softwarequalität sicherzustellen sind geeignete konstruktive und analytische Maßnahmen durchzuführen.

Analytische QS-Maßnahmen sind im Rahmen des Projekts vorgesehen.

- Interne QS- und Peer-Reviews für Produkte
    - Prüfung von Dokumenten (Spezifikationen, Architekturdokumente, etc.)
    - formale Prüfung der Dokumente auf Fehler, Inkonsistenzen und Unvollständigkeiten.
- Design Review
    - Überprüfung der Gesamtspezifikation und der fachlichen und technischen Feinkonzeptionen.
    - Prüfung der Integrierbarkeit der technischen Lösung in das Gesamtsystem.
    - Identifikation und Prüfung von technischen Risiken, zum Beispiel Technologien und Standards, Eindeutigkeit zwischen Anforderungen und Akzeptanzkriterien
- Code Reviews durch erfahrene Softwareentwickler
- Statische Code-Analyse (unter Verwendung der unter konstruktiven QS eingeführten Regelsätze)
    - Einsatz von Qualitätssicherungswerkzeugen bereits bei der Codeentwicklung. Für die Codeanalyse kommen folgende Werkzeuge zum Einsatz:
        - PMD[11]: Erkennung von ineffizientem Code

[11] http://pmd.sourceforge.net/

  - Checkstyle[12]: Überprüfung der Einhaltung von syntaktischen Kodierrichtlinien
  - Findbugs[13]: Ermittlung von kritischen Codestellen
  - SonarQube[14]: als Integrationsplattform zur Darstellung und Auswertung der Ergebnisse einzelner Tools (PMD, Checkstyle (WhiteBox), FindBugs (BlackBox) u.a.)

- Tests
  - Prüfung der auszuliefernden Software in mehreren Phasen
  - Entwicklertests
  - Modultests
  - Oberflächentests
  - Integrationstests / Systemtests
  - Regressionstests
  - Automatisierte Tests
  - Entwicklungsbegleitende Vortests durch den Auftraggeber

Das Erreichen von Softwarequalität wird durch den Einsatz geeigneter technischer Werkzeuge unterstützt. IBM setzt dabei für das Projekt auf die bereits erfolgreich im forumSTAR und eKP Projekt eingesetzten Werkzeuge zur Qualitätssicherung:

- Rational Functional Tester (RFT)

  Tool, das automatische Testfunktionalität für Funktions- und Regressionstests, Tests der grafischen Benutzerschnittstelle und datengesteuerte Tests bereitstellt.

- Codeanalysewerkzeuge

[12] http://checkstyle.sourceforge.net/

[13] http://findbugs.sourceforge.net/

[14] http://www.sonarqube.org/

Für die Codeanalyse kommen die Werkzeuge PMD, Checkstyle und Findbugs zum Einsatz. Diese sind über das Produkt SonarQube (Integrationsplattform zur Darstellung und Auswertung der Ergebnisse einzelner Analyse-Werkzeuge) integriert,

IBM bietet auch entwicklungsbegleitende Tests durch den Auftraggeber an. Dabei werden nicht nur Vortests bei der IBM genutzt werden, sondern auch eine Terminalserver-basierte Testumgebungen, die IBM für den Auftraggeber bereitstellt und betreibt, eingesetzt, um Entwicklungsstände in einer frühen Phase testen zu können.

11.2 Performance

Ziel ist es, mögliche Performance-Probleme so früh wie möglich im Softwareentwicklungsprozess zu identifizieren bzw. zu vermeiden.

Grundsätzlich sind die Performance-Themen im eIP Umfeld mehrschichtig zu sehen, da diese oft von den angebundenen Fachverfahren und den entsprechend gelieferten Daten abhängen.

Die Konfigurationsverwaltung sowie die Aufgabenverwaltung von eIP nutzen sehr wenige Daten und ein einfaches Datenmodell, wodurch Performanceengpässe nicht zu erwarten sind. Anders stellt sich der Zugriff auf die elektronische Akte dar, welche performant entsprechend viele Dokumente und Strukturen liefern muss. In dem Umfeld ist das Antwortzeitverhalten als kritisch anzusehen.

Folgende Maßnahmen etabliert:

- Berücksichtigung von Performanceaspekten während der Konzeption und des Anwendungsdesigns

  Bereits während der Anforderungsspezifikation bzw. Fachfeinkonzeption müssen im Hinblick auf die Performance nicht-funktionale Anforderungen (z.B. Antwortzeitverhalten, Aufrufhäufigkeit einer fachlichen Funktion und Mengengerüste) berücksichtigt werden. Die nicht-funktionalen Anforderungen haben eine direkte Auswirkung auf das Anwendungsdesign

- Performante Implementierung von neuem Programmcode

  Im Zuge der regulären Qualitätssicherungsmaßnahmen werden Programmiervorgaben verwendet, welche auch Vorgaben für performanten Programmcode beinhalten. Diese Programmiervorgaben stellen Regeln für jeden Programmierer dar, die bei der Erstellung bzw. Anpassung von Programmcode einzuhalten sind. Dazu zählen unter Performanceaspekten beispielsweise die Berücksichtigung des JAVA-Speichermodells und

dessen effiziente Nutzung. Dadurch können Speicherlecks (engl. Memory Leaks) aktiv verhindert werden.

Performancekritische Programmstellen können außerdem mit Hilfe von Profiling genauer untersucht und damit konkrete Schwachstellen aufgedeckt werden (z.B. mehrfach geschachtelte Schleifen), die meist in Verbindung mit großen Datenmengen Performanceprobleme verursachen.

Die Überprüfung der Einhaltung der definierten Programmiervorgaben erfolgt werkzeuggestützt. Dabei kommen Werkzeuge wie PMD, Checkstyle und Findbugs zur statischen Quellcodeanalyse zum Einsatz, die sich nahtlos in die Entwicklungsumgebung integrieren lassen und deren Verwendung für die Entwickler verpflichtend sind.

Neben der werkzeuggestützten, statischen Überprüfung der Programmiervorgaben ist ein Code-Review-Prozess nach dem 4-Augen-Prinzip etabliert. Dabei wird neu erstellter bzw. angepasster Programmcode von einem Teamleiter der Entwicklung visuell kontrolliert. Dies beinhaltet auch die Kontrolle unter Performanceaspekten.

- Test performancekritischer Funktionen (z.B. in Hinblick auf Antwortzeiten bzw. Durchsatz)

  Der Test erfolgte bisher i.d.R. mit einem angepassten Testdatenbestand. Damit lassen sich Performanceprobleme aktiv meist nicht feststellen. Aus diesem Grund soll für den Test von performancekritischen Bereichen ein anonymisierter Testdatenbestand sowohl vom Fachverfahren als auch von Dokumenten verwendet werden.

Performanceverbesserungen sind durch Maßnahmen über alle Phasen des Softwarelebenszyklus hinweg zu erreichen. Diese lassen sich die folgenden Aspekte zusammenfassen.

- Anforderungsspezifikation und Fachfeinkonzeption
  - Analyse und Dokumentation nichtfunktionaler Anforderungen (Antwortzeitverhalten, Aufrufhäufigkeit einer fachlichen Funktion, Mengengerüste)
  - Ermittlung von Aufrufhäufigkeiten der (Endanwender-) Funktionen
  - Prüfung
- Entwicklung
  - Vorsehen von Caching-Mechanismen bei einer hohen Zahl von zu verarbeitenden Daten bzw. kurzes Antwortzeitverhalten
  - Nutzung von asynchrone Hintergrundaktivitäten

- Vorgaben für die Erstellung von performanten Programmcode (Berücksichtigung des JAVA Speichermodells)
- Verhinderung von Speicher Lecks (Memory Leaks)
- Durchführung von Code Reviews (Kontrolle von Performanceaspekten)

- Test
  - Durchführung dedizierter Tests für performancekritische Bereiche
  - Verwendung von anonymisierten Echtdatenbeständen bei Regressionstests

11.3 Testvorgehen und Auswirkungen auf Testautomatisierung

eIP hat als weiteres Ziel, die Testbarkeit der Anwendung für einen höheren Automatisierungsgrad zu erreichen. Ziel der Testautomatisierung ist die Effizienzsteigerung in der Testdurchführung und die Verringerung von Testaufwänden vor der Auslieferung von Releases sowie die Beschleunigung der Abnahmetests beim Auftraggeber.

11.3.1 Automatisierte Oberflächentests

Prinzipiell können alle über die Oberfläche ansteuerbaren Funktionen automatisiert getestet werden. IBM setzt dazu im Projekt forumSTAR bereits heute IBM Rational Functional Tester (RFT) erfolgreich ein. Der Auftraggeber nutzt RFT ebenfalls für die automatisierten Tests von forumSTAR.

RFT ist ein Tool, das automatische Testfunktionalität für Funktions- und Regressionstests, Tests der grafischen Benutzeroberfläche und datengesteuerte Tests bereitstellt.

Als Programmiersprache stellt RFT unter anderem Java zur Verfügung, was einen modularen Aufbau der Testskripte erlaubt. Mit RFT bieten sich folgende Möglichkeiten:

- Datengetriebene Tests können durch die Verwendung von Variablen oder regulären Ausdrücken anstelle von fest kodierten Werten im Skript erreicht werden. Externe Daten können in verschiedenen Formaten (Text, Tabelle, Datenbank) bereitgestellt werden. Der Zugriff ist über Java möglich.
- Ergebnisprüfungen können als integrierte Prüfpunkte (Verification Points) während der Testaufzeichnung als auch nachträglich in das Skript eingefügt werden. Dabei können Objektdaten (zum Beispiel Tabelleninhalte), Objekteigenschaften (zum Beispiel Status

einer Schaltfläche) oder Bilder (Vergleich von Bitmaps) geprüft werden. Darüber hinaus steht ein Rahmenwerk für die Prüfpunkte über die Programmierschnittstelle zur Verfügung, so dass eine Erweiterung der Ergebnisprüfung möglich ist.

- Wiederholungen und Varianten der Testskripte können durch die Verwendung von Schleifenoperationen oder Bedingungen, die von Java zur Verfügung gestellt werden, erreicht werden.

- Das Verhalten bei Fehlern ist zum einen über Optionen steuerbar, zum anderen kann für jeden erfassbaren Fehler ein Error Handler erstellt werden.

Durch eine Anpassung der Objekterkennung des IBM Rational Functional Testers kann die Aufzeichnung der Testfälle mit den normalen Funktionalitäten des Werkzeugs erfolgen.

Im Rahmen der Architekturentscheidung (Kapitel „5.1 Architekturentscheidung Clienttechnologie“) wurde die automatisierte Testbarkeit ebenfalls mit betrachtet. Dabei wurde bereits herausgestellt, dass die nativen Windows Controls, wie sie von SWT verwendet werden, für die Testautomation Vorteile bieten können.

Es kommt der Mechanismus Capture & Replay zum Einsatz. Das von forumSTAR bekannte Testframework wird nicht verwendet.

11.3.2 Automatisierte Servicetests

Für die automatisierte Prüfung von Services bieten sich Testframeworks wie z.B. JUnit oder Soap-UI an, um automatisierte Tests ohne Benutzeroberfläche durchführen zu können. Diese können auch zu sogenannten Testsuites zusammengefügt werden, um aggregierte Tests durchzuführen.

Im Rahmen der eKP-Entwicklung wurden bereits JUnit-Tests für verschiedene Servicetests eingesetzt, die auch periodisch ausgeführt werden. Automatisierte Servicetests sind für synchrone Abläufe und Serviceaufrufe gut umsetzbar.

Der Umfang der serverseitigen Dienste ist in eIP gering. Es werden rudimentäre JUnit-Testfälle für die automatisierte Prüfung der Services umgesetzt. Diese werden auch dem Auftraggeber ausgeliefert und mit einer entsprechenden Dokumentation für Ausführungen in der Testumgebung des Auftraggebers versehen.

11.3.3 Herstellung der Datenbasis für automatisierte Tests

Automatisierte Tests können nur sinnvoll ablaufen, wenn sie auf bekannten Datenkonstellationen aufsetzen oder diese im Vorfeld selbständig aufbauen. Es sind verschiedene Ansätze für

dieses Grundproblem vorhanden. Unter anderem sind die folgenden Ansätze im Rahmen von eIP denkbar:

- Bereitstellung von entsprechend aufbereiteten Datenbank-Dumps und DMS Inhalten und Einspielen der Datenbank-Dumps / DMS Inhalte für die automatisierten Tests
- Hierarchisch aufgebaute Testfälle, wobei zuvor ausgeführte Testfälle die Datenkonstellation für die nachfolgenden Testfälle herstellen
- Bereitstellung in sich abgeschlossener Testfälle ohne Datenanforderung

Die verschiedenen Ansätze haben jeweils Vor- und Nachteile, welche sich darüber hinaus je nach Testvorhaben (automatisierte Oberflächentests für Regressionstests, Servicetests und Last-/Performancetests) unterschiedlich darstellen.

Details zu diesen Vorgehensweisen müssen im Rahmen eines Testkonzepts erarbeitet und dargestellt werden.

11.4 Historisierung / Versionsmanagement

11.4.1 Fachliche Historisierung

Unter Historisierung wird die Speicherung der zeitlichen Entwicklung der Daten verstanden. Typisches Beispiel ist die Speicherung der Preisentwicklung eines Produkts oder die Änderungen, welche an den Daten einem Dokument im Laufe der Zeit vorgenommen worden sind.

Mit diesen historisierten Daten kann der Zustand der Daten zu jedem Zeitpunkt ermittelt werden (Vergangenheit wie Zukunft). Wird auch der Aktor mit bei den historisierten Daten vermerkt, dann kann auch die Frage beantwortet werden „Wer hat wann welche Daten geändert?“.

Aktuell existieren keine Anforderungen an eIP, dass eIP eine vollumfängliche fachliche Historisierung unterstützen muss.

Konkret müssen innerhalb von eIP die verschiedenen Daten, die verwaltet werden, betrachtet werden:

- Konfigurationswerte
- Aufgaben
- Dokumente / Ordner der eAkte

Wird diese Anforderung zu einem späteren Zeitpunkt relevant, so muss die dann umgesetzte Historisierung auf jeden Fall folgende Anforderungen erfüllen:

- **Richtigkeit der Information:** Zu einem gegebenen Zeitpunkt darf nur ein einziger Objektzustand gültig sein.
- **Nachvollziehbarkeit:** Alle Daten, die abgespeichert und geändert wurden, müssen nachvollziehbar sein.
- **Nachweisbarkeit:** Die Nachvollziehbarkeit der Systemreaktionen ist nur möglich, wenn nachgewiesen werden kann, welche Objektzustände in der Datenbank zu einem gegebenen Zeitpunkt bekannt waren.
- **Erhöhung der Verfügbarkeit der Daten:** Alle Zustände eines Objekts (frühere, gegenwärtige und zukünftige) sollen in der Datenbank gespeichert und auf die gleiche Art verarbeitet werden können.

Die Historisierung soll transparent für die Anwendung erfolgen, um diese nicht mit den Belangen der Historisierung zu belasten. Ist zudem ein Zugriff der Fachanwendung auf die historisierten Daten notwendig, so ist ein Historisierungsdienst innerhalb der Persistenzschicht vorzuziehen.

Sollen aus Revisionsgründen ausschließlich die Datenänderungen protokolliert werden und ist kein direkter Zugriff aus der Fachanwendung auf die historisierten Daten gefordert, so können jegliche Datenänderungen über Datenbanktrigger in Spiegeltabellen kopiert werden. Dies ist dann ein Lösungsansatz, welcher jederzeit ohne Anpassungen der Fachanwendung auf einige oder alle Daten der Fachanwendung angewendet werden kann.

11.4.2 Technische Versionierung von Services

Jeder Service erhält eine eindeutige Versions- und Buildnummer. Diese kann über eine spezielle Methode des Service (`getVersion`) abgerufen werden (siehe dazu auch Kapitel 9).

Zusätzlich werden diese Informationen im MANIFEST.MF innerhalb des gepackten Archivs (EAR) abgelegt, so dass diese ohne Aufrufe des Codes kontrolliert werden können.

Die Informationen werden automatisch vom Buildprozess in diese beiden Bereiche eingefügt.

Somit hat der Betrieb verschiedene Möglichkeiten, die korrekten Versionen zu kontrollieren, was die Betreibbarkeit der Anwendung erhöht.

11.4.3 Technische Versionierung der Datenbank

In eIP wird eine Versionsnummer der Datenbank in einer Tabelle abgelegt. Diese wird von den Aktualisierungsskripten der Datenbank genutzt, um herauszufinden, welche Skripte für die aktuell zu installierende Version aufzurufen sind. Nach der Aktualisierung wird die Datenbankversionsnummer auf die aktuellste Version gesetzt. Die Aktualisierungsskripte sind so aufgebaut, dass sie auch mehrfach hintereinander ohne Fehler aufrufbar sind. Der gleiche Mechanismus kommt ebenfalls bei eKP zum Einsatz und hat sich bewährt.

Eine weiterführende Prüfung des Datenbankschemas erfolgt in eIP aufgrund des einfachen Datenmodells derzeit nicht.

Die Prüfung des Datenmodells kann in Zukunft im Rahmen der Verfügbarkeitsprüfung (`performHealthCheck`) der Services umgesetzt und somit vom Betrieb aktiv in das Betriebsmonitoring eingebunden werden. Zur Laufzeit bzw. während der Nutzung der Services durch einen Benutzer werden aus Performancegründen keine Konsistenzprüfungen mit der Datenbank durchgeführt – d.h. es wird nicht bei jedem Zugriff geprüft, ob eine zur Serviceversion passende Datenbankversion installiert ist.

11.4.4 Technische Versionierung des DMS Modells

Die Verwaltung des DMS-Modells obliegt dem jeweiligen DMS Produkt. Über CMIS ist es allerdings nicht möglich, eine Versionsnummer o.ä. in Bezug auf das DMS Modell abzufragen. Somit kann mit den Standardmitteln keine Versionsprüfung stattfinden.

Die CMIS-basierte Implementierung für den Zugriff auf das DMS prüft beim ersten Aufruf, ob die zentral benötigten Ordner (z.B. DMS für die Akten, EKP für die elektronischen Nachrichteneingänge) vorhanden sind.

Eine weiterführende Prüfung des DMS Modells (z.B. notwendige Attribute je Entität vorhanden) erfolgt in eIP derzeit nicht.

Die Prüfung des Datenmodells kann in Zukunft im Rahmen der Gesundheitsprüfung (`performHealthCheck`) der Services umgesetzt und somit vom Betrieb aktiv in das Betriebsmonitoring eingebunden werden. Zur Laufzeit bzw. während der Nutzung der Services durch einen Benutzer werden aus Performancegründen keine Konsistenzprüfungen mit der Datenbank durchgeführt.

11.4.5 Versionsprüfung zwischen den Services

Eine serviceorientierte Architektur zeichnet sich durch ihre Aufteilung und Kapselung der einzelnen Aufgaben in einzelne Services aus. Jeder Service bietet somit Operationen an, die

einen dedizierten Zweck erfüllen. Nach außen bieten Services Schnittstellen für den Zugriff an. Die Schnittstelle eines Service zeichnet sich durch die Signatur sowie durch die übermittelten Datenstrukturen aus. Sobald ein Service von anderen Services genutzt wird, muss bei einer Aktualisierung des Service auf die Einhaltung der Schnittstelle besonderen Wert gelegt werden. Sobald sich die Schnittstelle ändert, müssen auch alle abhängigen Komponenten aktualisiert werden.

Im Rahmen von eIP ist eine Verwendung der Services untereinander sehr eingeschränkt. Es wird lediglich der *ConfigurationService* wiederverwendet. Somit wird auf eine weiterführende Versionsprüfung der Services verzichtet.

11.4.6 Versionsprüfung zwischen Client und Server

Zwischen Client- und Serverversionen existieren meist identische Abhängigkeitsprobleme, da der Client bestimmte Versionen der genutzten Services voraussetzt.

Aufgrund der geringen Abhängigkeiten ist in eIP aktuell keine Versionsprüfung zwischen Client und Server geplant.

Im Rahmen von eIP kann eine Prüfung zwischen Client- und Serverversion umgesetzt werden. Nachdem der Client aber aus verschiedenen Modulen bestehen kann, die wiederum unterschiedliche Serverkomponenten nutzen, gelten für den Client in Zukunft ähnliche Voraussetzungen wie für die Serverseite und der dort beschriebenen Abhängigkeitsmatrix.

Die jeweiligen Clientmodule werden auf Aufforderung die Versionen der notwendigen Services prüfen können. Dabei kommen ähnliche Ansätze wie bei der Serverprüfung zum Einsatz.

12 PRODUKTENTSCHEIDUNGEN

Es kommen verschiedene Produkte in eIP zum Einsatz, da eine Eigenentwicklung in diesen Bereichen aus verschiedenen Gründen nicht sinnvoll, zeitlich leistbar oder zu kostenintensiv ist.

12.1 PDF Renderer: PDFXChangeViewer

Für die Anzeige der Dokumente der eAkte ist ein PDF Viewer / Renderer notwendig, der eingebettet in eIP genutzt werden kann. Dabei sind die folgenden Kriterien maßgeblich für die Auswahl eines entsprechenden Produkts:

- Mehrfache Instanziierung des Viewers
    - Innerhalb von eIP können beliebig viele Instanzen der eAkte geöffnet sein, somit muss der Viewer dazu in der Lage sein
- Ausblendbarkeit der Bedienelemente
    - Es müssen sämtliche Bedienelemente im Viewer ausblendbar sein (Menüs, Toolbars, Kontextmenüs)
- Integrierbarkeit des Viewers in eine andere Anwendung (in unserem Fall eIP)
    - Native Anwendungen müssen dabei als OLE Control (ActiveX Control) vorliegen
    - Java Anwendungen müssten für eine Integrierbarkeit speziell geprüft werden
    - Alternativ kann ein Bearbeitungsfenster mittlerweile auch eingefangen werden - dann müssen trotzdem die Bedienelemente usw. ausblendbar und eine externe Schnittstelle aufrufbar sein.
- Unterstützung der fachlich gewünschten Funktionen
- Aufruf sämtlicher fachlicher Funktionen des Viewers (die innerhalb von eIP genutzt werden sollen) über eine extern zugängliche Schnittstelle
    - eIP ist in der Lage, Java Schnittstellen oder COM Schnittstellen aufzurufen
    - Es ist nicht notwendig, dass die Schnittstellen spezifisch für jede Funktion einzeln ausgeprägt vorliegen. Beispielsweise sind im PDFXChangeViewer manche Funktio-

nen über generische JavaScript Funktionen aufrufbar. Dies ist z.B. auch bei Acrobat Professional so umgesetzt.

- Export und Import von Annotationen als XFDF Format
  - Annotationen werden als XFDF Dateien im DMS verwaltet und bei Änderungen aus dem Viewer exportiert, zerlegt (Unterstützung der Sichtbarkeitsregeln) und im DMS gespeichert.
  - Der Viewer muss somit beliebig viele XFDF Dateien für ein Dokument importieren und zur Anzeige bringen können. Aktuell werden bis zu vier XFDF Dateien (freigegebene, OE-weite, persönliche, Normfall Manager Annotationen) importiert.
- Lizenzpolitik und Preis

Als mögliche Produkte wurden bisher

- PDFXChangeViewer
- Jadice
- Acrobat Acrobat Professional
- Nuance Power PDF Advanced
- Nitro PDF Viewer
- Foxit Phantom PDF

betrachtet.

Der Auftraggeber hatte ursprünglich den Einsatz des Adobe Professional aufgrund der hohen Kosten grundsätzlich abgelehnt. Im Rahmen der eIP 1.3.0 Entwicklung wurde Adobe Acrobat Pro DC als weiterer PDF Renderer integriert. Diese Integration scheiterte am Ende aufgrund aktueller Produktlimitierungen und fehlender Detailschnittstellen.

PDFXChangeViewer wurde in verschiedenen Großprojekten öffentlicher Kunden als auch im Normfall Manager als PDF Viewer eingesetzt und hat damit seine Integrationsfähigkeit unter Beweis gestellt. Somit wurde dieses Produkt für die Pilotierungsumsetzung gewählt. Notwendige Verbesserungen und Optimierung bei der Unterstützung eines barrierefreien Zugangs zu den Inhalten eines PDFs sind beim Produkthersteller adressiert.

Der PDFXChangeViewer wird aktuell nicht mehr weiterentwickelt. Das Folgeprodukt (PDFXChangeEditor) bietet im Umfeld der Barrierefreiheit noch keine nennenswerten Neue-

rungen und hat zudem eine andere Schnittstelle als der PDFXChangeViewer. Eine Alternative ist derzeit nicht vorhanden.

12.2 Strukturierungswerkzeug: Normfall Manager

Es gibt verschiedene Strukturierungswerkzeuge am Markt (z.B. Normfall Manager, NeBis von Fallsoft und weitere). Der Normfall Manager wird bereits an verschiedenen Stellen in Bayern eingesetzt, weshalb dieses Produkt in eIP integriert werden sollte. Die Integration sollte aber folgend den Architekturvorgaben lose gekoppelt umgesetzt werden, um einen späteren Austausch zu ermöglichen. Die Anbindung ist somit optional zu sehen und keine zwingende Voraussetzung für den Betrieb von eIP mit und ohne eAkte.

12.3 DMS: Alfresco

Zu Beginn der prototypischen Umsetzung von eIP wurde die Produktneutralität, wie auch in den Prämissen dokumentiert, stark hervorgehoben. Aus diesem Grund hat der Auftraggeber den Wunsch geäußert, eine Open Source Lösung anzubinden, um nicht den Anschein einer Produkt-/Herstellerverbundenheit zu signalisieren.

IBM hat in Abstimmung mit dem Auftraggeber aufgrund des folgenden, kurzen Anforderungsprofils einige wenige Alternativen beleuchtet und am Ende die Entscheidung für Alfresco für die prototypische Umsetzung getroffen. Aktuell wird Alfresco auch im Pilotbetrieb in Bayern eingesetzt, wobei die Entscheidung für die weitere produktive Nutzung noch nicht getroffen wurde.

Folgende Kriterien wurden kurz betrachtet

- Open Source Projekt
- Lizenzvereinbarungen
- CMIS Unterstützung
- Volltextsuche
- Art des Content Repositories
- Unterstützte Datensenken z.B. Datenbanken
- Anpassungsfähigkeit der Metadatenstrukturen eines Dokuments
- Unterstützung mehrerer Betriebssysteme

- Betrieb in der IBM Testumgebung auf Oracle Enterprise Linux
- Umsetzungstechnologie (z.B. Java)

Die folgenden Umsetzungen wurden betrachtet:

- Alfresco
- Jackrabbit
- Ametys
- Nuxeo

Eine erste Evaluationsimplementierung auf Basis von OpenCMIS Chemistry und Jackrabbit scheiterte an den begrenzten Möglichkeiten der Metadatenstrukturen.

In Abstimmung mit dem Auftraggeber wird Alfresco genutzt, wenn auch der Funktionsumfang für eIP viel zu groß ist.

13 PROGRAMMIERMODELL

Ein einheitliches Programmiermodell ist essentiell wichtig für die Pflege, Wartung und Weiterentwicklung einer Software. Nur wenn einheitliche Vorgaben gemacht und auch eingehalten werden, ist eine Pflege und Wartung bei komplexen Großprojekten in einem großen Team sinnvoll möglich.

In diesem Kapitel wurden nur einige wenige grobe Richtungen definiert. Ein vollständiges Programmiermodell im Sinne eines Programmierleitfadens wird als separates Dokument erstellt, da an dieser Stelle nur ein rudimentärer Einblick vermittelt werden soll.

Neben Themen wie Fehlerbehandlung sowie Logging und Tracing, die in weiteren Kapiteln detailliert werden, muss ein Programmiermodell auch generelle Vorgaben beinhalten sowie spezifische Vorgaben für die jeweilige Client- und Servertechnologie.

13.1 Dokumentation

Die Dokumentationssprache für weiterführende Dokumentation ist Deutsch. Entwicklungsnahe Dokumentationen wie z.B. die technischen Vorgaben für eine Fachverfahrensintegration werden in Englisch geschrieben.

Die Sprache innerhalb der technischen Umsetzung (z.B. JavaDoc, Kommentare etc.) ist Englisch. Dabei müssen sämtliche öffentliche Methoden und Konstruktoren kommentiert werden, so dass eine Verwendung der API ohne Codekenntnis möglich ist. Weiterhin müssen private Methode kommentiert werden, sofern diese komplexe Sachverhalte abbilden. Inline-Kommentare können dies unterstützen. Implementierungen von Interfaces müssen die Dokumentation nicht duplizieren.

OpenSource Bibliotheken und das Java Runtime Environment werden vom Auftraggeber beigestellt. IBM führt für die gesamte Anwendung die verwendeten Bibliotheken und Frameworks inkl. Versionsnummer und Anwendungsbereiche in einer Liste. Diese wird bei jeder Lieferung aktualisiert mit ausgeliefert.

13.2 Programmiersprache

Die Programmiersprache ist Java. Es wird mindestens Java 1.7.0_75 64-bit eingesetzt. Dies gilt sowohl für die Client- als auch für die Serverkomponenten.

13.3 Namenskonventionen

Klassen und Methoden werden Englisch benannt, um internationale Standards einzuhalten.

Deutsche Begriffe aus der fachlichen Domäne (Justizumfeld, Datenmodell), zu denen keine treffenden englischen Begriffe existieren, sind zulässig. Es wird eine einheitliche Übersetzungstabelle erstellt, mit der fachliche Begriffe als englische Begriffe abgebildet sind.

Detaillierte Vorgaben für Klassen, Interfaces, Packages werden im Rahmen des Programmierleitfadens erstellt. Grundsätzlich wird allgemeinen Konventionen[15] gefolgt.

Grundlegend wird von dem Package-Prefix `de.justiz.eip` ausgegangen.

Alle Projekte im eIP Umfeld werden entsprechend des Eclipse RCP Standards analog zu den Java Packages benannt (z.B. `de.justiz.eip.digitalfile.viewer.ui`). Dies betrifft sowohl Projekte für den Server als auch für den Client.

13.4 Client

Jede fachlich zu integrierende Komponente wird als ein oder mehrere eigene Plugins ausgeprägt. Dabei ist es unerheblich, ob Plugins sichtbare oder nicht sichtbare Komponenten beinhalten.

Die konkreten Programmiervorgaben für den Client inkl. der einzusetzenden Programmiermuster werden im Programmierleitfaden für Eclipse RCP mit SWT detailliert.

Es kommt Eclipse RCP 4.3.1 (Kepler) zum Einsatz.

13.5 Server

Sämtliche Java Implementierungen müssen der JEE und damit der EJB Spezifikation genügen. Es kommt JEE 5 mit EJB 3.0 zum Einsatz. Nach einer Umstellung auf die Oracle FusionMiddleware 12 kann über den Einsatz von EJB 3.1 nachgedacht werden.

Die weiteren Programmiervorgaben werden im Programmierleitfaden definiert.

[15] Code Conventions for the Java™ Programming Language, z.B. Version vom 20.04.1999 unter https://www.oracle.com/technetwork/java/codeconvtoc-136057.html

13.6 Annotationen

Es sind die Annotation `@NonNull` und `@Nullable` für Methodenargumente und -rückgabewerte oder semantische Äquivalente einzusetzen.

13.7 Fehlerbehandlung

Fehlerbehandlung und -verarbeitung ist ein sehr wichtiges Thema bei jeder Applikationsentwicklung. Der Anwendungsentwickler sollte nicht mit der Verarbeitung von konkreten Fehlern belastet werden. Dies sollte durch klare Vorgaben und eine entsprechenden Software-Architektur abgehandelt werden, um eine konsistente Implementierung mit vorhersehbarem Verhalten zu erhalten.

Die Reaktion auf einen Fehler muss extern konfigurierbar sein, um z.B. den Schweregrad eines Fehlers, den Fehlertext o.ä. ändern zu können. Ansonsten können Änderungen in diesen Bereichen nur über Anpassungen im SourceCode erfolgen. In der aktuellen Umsetzung von eIP erfolgt die Fehlerkonfiguration in XML- und Propertiesdateien außerhalb des SourceCodes.

Der gesamte Mechanismus basiert im wesentliche auf dem Java Exception Mechanismus. Dieser Mechanismus ist stabil, bewährt und standardisiert. Ein Fehler wird demzufolge mit einer Exception und nicht mit Rückgabewerten an den Aufrufer kommuniziert.

Jede Exception kann weitere Informationen über den Fehler enthalten.

IBM nutzt einen zentralen ErrorHandler, welcher von einer Komponente bei Auftreten eines Fehlers gerufen wird. Anhand einer Fehlerkennung wird ein Fehlerobjekt erzeugt, das Logging automatisch aufgrund konfigurierbarer Attribute des Fehlerobjektes durchgeführt und das Fehlerobjekt für die weitere Verarbeitung in einer Exception an den Aufrufer übergeben. Damit ist eine zentrale Instanz für die Fehlerbehandlung vorhanden, welche die korrekte Beschreibung und eine angemessene Reaktion auf den Fehler ermöglicht. Die Eigenschaften des Fehlerobjektes definieren auch, ob eine SNMP-Notification abgesetzt werden soll. Wenn ein entsprechender Fehler auftritt, löst der zentrale ErrorHandler einen entsprechenden Event aus.

Die Konfiguration der Fehler erfolgt über XML für die Fehlerkonfiguration (z.B. Schweregrad, Applikations-Komponente, SNMP-Trap) und Java Properties-Dateien für die sprachabhängigen Fehlertexte.

13.8 Logging und Tracing

Logging und Tracing sind integraler und wichtiger Bestandteil einer Anwendungsarchitektur und müssen von Beginn an vorgesehen und integriert werden.

Logging wird für die Ausgabe von Fehlerzuständen genutzt. Anhand von Log-Dateien können Fehler lokalisiert und, unter Voraussetzung eines sinnvoll gepflegten Fehlerhandbuchs, möglicherweise direkt behoben werden (z.B. Nicht-Verfügbarkeit einer Datenbank). Ein Logger wird innerhalb einer Applikation verwendet, um Fehler, die beim Ablauf innerhalb dieser Applikation auftreten, zu protokollieren.

Ein Tracer wird in einer Applikation verwendet, um den Applikationsfluss (Entry/Exit von Methoden) abzubilden. Die Ausgabe wird für das Debugging einer Applikation genutzt, um einen Fehler zu lokalisieren und den Ablauf zu klären. Beim Entry müssen dabei die übergebenen Parameter beim Exit die Rückgabewerte protokolliert werden.

Es ist sinnvoll, dass die Datensenken für Logger und Tracer frei konfigurierbar sind. Datensenken können dabei Dateien, Datenbanken, Events für Monitoring o.ä. sein. Weiterhin ist es oft notwendig, mehrere Datensenken parallel mit gefilterten Events zu versorgen. Beispielsweise sollen nur Fehler an ein Monitoringsystem gemeldet werden, alle anderen Events sollen z.B. in Dateien abgelegt werden.

Die Komponenten loggen sämtliche Fehler in eigene Fehlerdateien sowie in eine zentrale Fehlerdatei pro Clusterknoten. Dabei werden pro Laufzeitumgebung eigene Logs geschrieben.

Es kommt Apache Log4J als Logging-Framework zum Einsatz. Innerhalb der Konfigurationsdateien für Log4J können verschiedenste Datensenken (über so genannte Appender) definiert werden. Die Inhalte werden über Filter-Elemente eingegrenzt.

Zusätzlich dazu kann ein Trace aktiviert werden, welcher die Aktivitäten inkl. Parameter innerhalb der Komponenten detailliert in eine eigene Datei schreibt. Analog zu den Log-Dateien können pro Komponente eigene Traces in eigene Dateien geschrieben werden. Die Standardkonfiguration von eIP schreibt alle Traceausgaben aller Komponenten in eine Datei.

Die vordefinierten Werte sind als globale Konstanten (z.B. Severity-Texte) implementiert, die Kennung der Komponente ist als Konfigurationseinstellung des Layouters konfigurierbar.

13.8.1 Vorgaben für Logging

Logging ist bei jedem Fehlerfall implizit einzubauen. Dabei spielt es keine Rolle, ob der Fehler durch eine Exception oder durch eine logische Prüfung festgestellt wurde. Durch den Einsatz des zentralen ErrorHandlers erfolgt das Logging automatisch.

13.8.2 Vorgaben für Tracing

Grundsätzlich ist in allen Methoden ein ENTRY und EXIT als Trace-Anweisung einzufügen. Am Client kommt hierzu AspectJ zum Einsatz, so dass die Programmierung dies am Client nicht vorsehen muss. Dabei ist zu beachten, dass auch im Fehlerfall ein EXIT geschrieben werden muss – die Trace-Anweisung muss dabei ein EXIT_FAIL beinhalten.

In den folgenden Ausnahmen sind keine ENTRY/EXIT-Anweisungen notwendig:

- Einfache get-/set-Methoden
- Leermethoden inkl. leeren Konstruktoren
- Methoden mit trivialer Logik (analog get-/set-Methoden)

Es kann vorkommen, dass in Spezialfällen das Tracing in einzelnen Methoden deaktiviert werden muss, um die Lesbarkeit des Trace sicherzustellen. Dies ist dann der Fall, wenn Methoden z.B. in Schleifen extrem oft gerufen werden.

13.9 Zugriff auf Konfigurationsdaten

Konfigurationsdaten werden in einer Datenbank abgelegt. Der Zugriff erfolgt dabei durch einen Service (ConfigurationService).

Auf der Serverseite wird dieser Service von den anderen Services gerufen. Dabei kommen EJB Annotations zum Einsatz, die die Auffindbarkeit des Service in der Installation durch den JEE Container einsetzen.

Auf der Clientseite steht eine Erweiterung des Eclipse Preferences Mechanismus zur Verfügung. Dieser liest die Einstellungen über den *ConfigurationService* aus der Datenbank aus, stellt diese (teilweise) für die Bearbeitung in Konfigurationsmasken zur Verfügung und regelt die Speicherung in der Datenbank. Dabei kann es je nach Komponente veränderbare und nicht veränderbare Konfigurationsparameter geben.

Am Client sind weiterhin Konfigurationsparameter über die Systemumgebung notwendig. Diese werden in hierarchisch organisierten Konfigurationsdateien (z.B. app.conf) abgelegt. Details zu den Konfigurationsmöglichkeiten sind im Installationshandbuch zu finden.

14 ANHANG

14.1 Weiterführende Aspekte der Ausfallsicherheit

Die folgenden Kapitel beschreiben mögliche Erweiterungen der Ausfallsicherheitsarchitektur von eIP, die derzeit noch nicht umgesetzt sind. Diese stellen nur eine Themensammlung dar, ohne dass abschließende Konzepte dafür vorliegen.

14.1.1 Aktuelle Nachteile des Ausfallsicherheitskonzepts

Änderungen an der Autorisierung des Benutzers (berechtigte Verfahren) werden nicht unmittelbar genutzt, solange der Offlinemodus besteht.

Es erfolgt keine Prüfung im Hintergrund, ob z.B. der SecurityProvider (und damit das Fachverfahren) wieder verfügbar ist. Diese Prüfung kann relativ lange dauern und ressourcenintensiv sein. Es ist allerdings möglich, über die Verbindungsübersicht die Prüfung aktiv anzustoßen.

Ungespeicherte Änderungen können bei Ausfall des jeweiligen Severbestandteils verloren gehen – abhängig von der Funktionalität der jeweiligen Anwendungskomponente.

Bei Ausfall des lokalen Netzes im Gericht ist eine Nutzung des Offline-Modus nur am Arbeitsgerät mit entsprechend gefüllten lokalen Client Caches möglich.

14.1.2 Unterstützung für schreibende Zugriffe im Offline-Fall

In bestimmten Szenarien sind schreibende Zugriffe im Offline-Fall ohne massive Auswirkungen umsetzbar.

Sämtliche persönliche Anmerkungen, Dokumente in der persönlichen Handakte und persönliche Aufgaben sind nur für den aktuell angemeldeten Benutzer sichtbar. Somit können hier Veränderungen vorgenommen werden, die sich nicht unmittelbar auf andere Benutzer auswirken und die damit zu keinem Konflikt bei schreibenden Zugriffen führen würden. Ein späteres Rückschreiben, sobald das System wieder online ist, wäre somit möglich.

Zu beachten ist dabei grundsätzlich, dass ein Benutzer auch mehrmals mit unterschiedlichen Geräten angemeldet sein kann. Damit können potentiell Konfliktsituationen entstehen, die organisatorisch zu regeln sind. Aus Sicht von eIP würde in dem Fall die letzte Änderung maßgeblich sein.

Die folgenden Anwendungsfälle könnten in Betracht kommen:

- Anbringen / Verändern von persönlichen Anmerkungen in der elektronischen Akte

  Es sind nur persönliche Anmerlungen vom aktuellen Benutzer sichtbar, diese können somit gefahrlos verändert und später synchronisiert werden. Es ist darauf zu achten, dass trotzdem nicht-persönliche Anmerkungen angesehen, aber diese nicht verändert werden dürfen. Diese Veränderung kann im PDFXChangeViewer aber nicht explizit deaktiviert werden. Somit würde dies dazu führen, dass Änderungen in nicht-persönlichen Anmerkungen nicht gespeichert werden.

- Veränderungen in der persönlichen Handakte

  Sämtliche Dokumente in der persönlichen Handakte sind nur für den aktuell angemeldeten Benutzer sichtbar. Somit können hier Veränderungen vorgenommen werden, die keinen Einfluss auf andere Benutzer haben (Hinzufügen, Löschen, Verändern von Dokumenten sowie Anbringen von persönlichen Annotationen).

- Strukturierung der elektronischen Akte in persönlichen Normfall Manager Projekten

  Die Markierungen von Textstellen, die in ein Normfall Manager Projekt übernommen werden, werden als persönliche Annotationen gespeichert. Somit gelten die gleichen Bedingungen wie bei persönlichen Anmerkungen. Dies ist nur bei Verfügbarkeit des zentralen Normfall Manager Projektspeichers möglich.

- Erledigen / Verschieben von persönlichen Aufgaben im Aktenbock

  Grundsätzlich können persönliche Aufgaben im Aktenbock ebenfalls durch den Endbenutzer verändert werden (z.B. das Erledigen von Aufgaben oder das Verschieben in den Bereich „Aufgaben“). Dabei ist allerdings zu beachten, dass andere Benutzer über den Vertretungsaktenbock ebenfalls diese Aufgaben sehen und ggf. übernehmen können.

Die aktuellen Anwendungsfälle gehen in vielen Fällen bei der Veränderung von der Verfügbarkeit des *DigitalFilesService* aus. Somit müssten hier alternative Speicherwege geschaffen werden.

14.2 Weiterführende Aspekte der Sicherheitsarchitektur

Die folgenden Kapitel beschreiben mögliche Erweiterungen der Sicherheitsarchitektur von eIP, die derzeit noch nicht umgesetzt sind. Diese stellen nur eine Themensammlung dar, ohne dass abschließende Konzepte dafür vorliegen.

14.2.1 Mischarbeitsplätze mit mehreren Fachverfahren

Bei Mischarbeitsplätzen wären mehrere Plugins für Fachverfahren aktiv, die wiederum alle einen SecurityProvider implementieren müssten. In dem Fall wird beim Start von eIP die Anmeldung an jedem Fachverfahren durchgeführt und ggf. mehrere Benutzeranmeldedaten in eIP verwaltet. Entsprechend muss bei Öffnen der Akte mehrere Fachverfahren nach der Berechtigung für das Verfahren gefragt werden, wenn eIP keine Logik für die Zuordnung eines Aktenzeichens zu einem Fachverfahren besitzt. Der Arbeitskontext in eIP kennt den entsprechenden SecurityProvider für dieses Verfahren und kann darüber die relevanten Abfragen stellen. In den eIP Aufgaben sind die entsprechenden Informationen über den relevanten SecurityProvider zu hinterlegen.

Vor allem aus betrieblichen Gründen wird ein eIP Mischarbeitsplatz mit mehreren Fachverfahren innerhalb von eIP als nicht tragfähig angesehen. Dieses Vorgehen wird derzeit nicht weiter verfolgt.

14.2.2 Unterstützung von SAFE für die Anmeldung (Authentisierung)

Eine Anmeldung an einer SAFE-Domäne (SAFE IdentityProvider) stellt aus Sicht von eIP keine Änderung der Sicherheitsarchitektur dar. Es muss ein eIP SecurityProvider für die Anmeldung gegen den SAFE IdentityProvider umgesetzt werden.

Für die Anmeldung an den Fachverfahren muss in den Fachverfahren die Möglichkeit bestehen, eine bestehende SAFE Anmeldung zu übernehmen bzw. eine SingleSignOn Lösung zu nutzen. Die Übernahme einer bestehenden Anmeldung (Übernahme des SAML-Tokens) ist für Rich Client Anwendungen derzeit im Rahmen des SAFE Konzeptes noch nicht berücksichtigt (Stichwort „on behalf" Authentisierung). Grundsätzlich bedeutet dies, dass die Fachverfahren dafür vorbereitet sein müssen. Solange dies nicht gegeben ist, ist eine vollumfängliche Umsetzung des SAFE Konzeptes nicht möglich.

Die Autorisierung verbleibt auch bei einer Anmeldung über SAFE im Fachverfahren und wird aus Sicht von eIP über den SecurityProvider des Fachverfahrens weiterhin zur Verfügung gestellt.

Dieses Konzept erfordert eine Erweiterung des eIP SecurityProvider-Konzeptes, da Authentisierung und Autorisierung von unterschiedlichen Providern durchgeführt werden müssen.

Allerdings ist generell der Umgang mit einer SAFE-Anmeldung in einer Rich Client Anwendung noch weitestgehend ungeklärt. In einem Gutachten (Hr. Streckel / dataport) wird die Verwendug von SAFE für eIP als nicht sinnvoll angesehen.

Die Verwendung von serverseitigen Diensten, die per SAFE abgesichert sind, ist ebenfalls denkbar. Auch hier ist die Thematik „on behalf" Authentisierung noch nicht in SAFE umge-

setzt, wodurch lediglich eine Autorisierung auf Basis der Dienstkennung möglich wäre (technischer Benutzer).

14.2.3 Absicherung der Kommunikation einer nativen Anwendung mit eIP

Die Kommunikation einer nativen Anwendung mit eIP erfolgt über die Integrationsbibliothek (siehe 5.11) oder über COM/OLE. Für COM/OLE ist keine zusätzliche Absicherung der Kommunikation notwendig und meist auch nicht möglich.

Bei der Kommunikation über die Integrationsbibliothek kommen Named Pipes zum Einsatz. Die Kommunikation zwischen eIP und Fachverfahren über Named Pipes ist quasi ungesichert und die Sicherheit kann über Betriebssystemmittel noch erhöht werden.

Somit besteht in dieser clientseitigen Kommunikationstechnik noch Optimierungspotential, um Angriffe zu vermeiden.

14.2.4 Erweiterte Datensichtbarkeit bei Aktenabgabe / Akteneinsicht

Im Rahmen der externen Akteneinsicht werden Akten und deren Inhalte für Externe zugänglich gemacht. Dabei müssen z.B. freigegebene Annotationen auf das PDF in der Akte aufgebracht und ggf. Schwärzungen vorgenommen werden. Erst diese veränderten Dokumente werden dann zur externen Akteneinsicht verwendet. Die Publikation bzw. Bereitstellung dieser Dokumente ist aktuell noch nicht geklärt. Grundsätzlich ist dafür das Akteneinsichtsportal konzipiert worden, die entsprechenden Umsetzungen in den Ländern und abschließende Freigabe sind noch nicht abgeschlossen.

Entsprechende Mechanismen für eine externe Akteneinsicht fehlen in der aktuellen Umsetzung. Diese sollen im Rahmen des Projektes „Akteneinsicht“ zukünftig erarbeitet werden.

Aktuell unterstützt eIP einen Export der Akte in das Dateisystem. Die Dokumente werden beim Export für die Akteneinsicht in einen ZIP-Container verpackt und digital signiert. Die Dokumente selbst sind nicht geschützt, damit diese sofort in einem PDF Viewer betrachtet werden können.

14.2.5 Erweiterungen der Rechtesteuerung

Erweiterungen im Rahmen der Rechtesteuerung wären insofern denkbar, als das bei der Anmeldung am Fachverfahren die Rechte für die elektronische Akte an eIP übergeben werden und eIP dann (client-seitig) eine Aktivierung/Deaktivierung der jeweiligen Funktionen in der eAkte umsetzt.

14.2.6 Verschlüsselte Ablage der Dokumente

Die Dokumente können verschlüsselt im DMS abgelegt (abhängig von dem DMS Produkt und dessen Konfiguration) und auch verschlüsselt transportiert werden. Dabei kommen moderne Verschlüsselungsverfahren (z.B. AES mit 256-bit Schlüssellänge) zum Einsatz, die zur Entschlüsselung die Kenntnis des jeweiligen Schlüssels voraussetzen.

Der Client, welcher die Dokumente anzeigen möchte, muss die Dokumente entschlüsseln. Bei eIP müssen die Dokumente entschlüsselt auf der Festplatte des Arbeitsplatzrechners (bzw. einem Netzwerklaufwerk) abgelegt werden (Stichwort Client Cache), damit diese vom PDF Viewer angezeigt werden können.

Der für die Entschlüsselung notwendige Schlüssel muss dem eIP Client vorliegen und kann aus der eIP Datenbank ausgelesen und vom Client abgerufen werden. Alle Dokumente werden mit dem identischen Schlüssel verschlüsselt.

Ohne gültigen Schlüssel können somit die Dokumente nicht geöffnet werden und eine Kenntnis der URL (inkl. Basic Authentication Header Daten) bringt einem potentiellen Angreifer des Systems keinen Vorteil.

Sobald die Dokumente entschlüsselt auf dem Arbeitsplatz-PC zwischengespeichert werden, sind diese entsprechend ungeschützt. Der Zugriff auf den Arbeitsplatz-PC muss entsprechend abgesichert werden, um diese Lücke zu schließen.

14.2.7 Erweiterter Schutz beim Zugriff auf die eIP Datenbank

Als Alternative zum umgesetzten Schutz beim Zugriff auf die DataSource kann eine DataSource auch ohne Benutzerdaten konfiguriert werden. In dem Fall muss die Anwendung Benutzerkennung und Passwort bei Anforderung einer Datenbankverbindung über die DataSource mit zur Verfügung stellen. Dies bedeutet, dass die Verwaltung der Anmeldedaten an der Datenbank in die Hoheit der Anwendung überführt wird. In diesem Fall könnten dann auch für jeden fachlichen Benutzer unterschiedliche Datenbankbenutzer zum Einsatz kommen. Innerhalb der Oracle Datenbank müssen dann die jeweiligen Benutzer gepflegt und die Rechte für die Tabellenzugriffe erhalten. Somit ist ein Synchronisationsmechanismus mit der Benutzerverwaltung umzusetzen, was der Prämisse 2 aus Kapitel 3.6 widerspricht.

14.3 Sicherheitsarchitektur - forumSTAR-spezifische Erläuterungen

14.3.1 Technische Umsetzung des SecurityProviders für forumSTAR

Im Fall von forumSTAR implementiert der SecurityProvider einen Großteil der Funktionen auf dem Server, wobei jedoch der Anmeldekontext des aktuellen Benutzers (GerichtspersonSession-Information im Stateful Session Bean [FacadeSession] von forumSTAR) implizit genutzt wird. Deshalb ist eine Verwendung dieser Funktionen für serverseitige Aufrufe ohne weitere Anpassungen nicht möglich.

14.3.2 Unterstützte Anmeldetechniken (Authentisierungsverfahren)

forumSTAR unterstützt drei Anmeldemechanismen (Test-Login mit/ohne Angabe einer vordefinierten Benutzerkennung (GPE_KEY_ID), passwort-basierte Anmeldung, Single-Sign-On basierend auf der Windows-Anmeldung (Kerberos-Login)), welche durch eine entsprechende Konfiguration direkt auch mit eIP nutzbar sind.

14.3.3 Abbildung der Rechtezuordnung in forumSTAR

In forumSTAR besteht ein komplexes Rechtevergabesystem basierend auf Rollen, Rechten sowie Zuordnungen zu Gerichten und Organisationseinheiten sowie dem entsprechend zugewiesenen Geltungsbereich. In diesem kurzen Kapitel werden die Administrationsmöglichkeiten und Begriffe nicht umfassend erklärt, da dies den Rahmen sprengen würde. Entsprechend soll dieses Kapitel dem forumSTAR-Versierten das Verständnis vermitteln, wie die Stufen in forumSTAR realisiert wurden. Ein umfassendes, vollständiges Konzept des forumSTAR Rollen- und Rechtekonzepts existiert leider nicht.

Stufe	Beschreibung
Freigegeben	Freigegebene Inhalte sind von allen Benutzern, welche ein Verfahren öffnen können, zu sehen und zu verändern.
OE-weit	OE-weite Inhalte wurden aus Sicht von forumSTAR so definiert, dass ein Benutzer einer Organisationseinheit (OE) explizit zugeordnet sein muss, um Inhalte dieser OE sehen zu können. Dies bedeutet, dass Benutzer, die ein Verfahren einer anderen OE aufgrund des Geltungsbereichs (abteilungsweite / gerichtsweite Sicht) zwar sehen und öffnen können, die OE-weiten Inhalte nicht sehen können (außer sie sind der OE zugeordnet). Dabei ist es unerheblich, ob der Benutzer der OE in der aktuellen Rolle oder in einer anderen Rolle zugeordnet ist. In beiden Fällen liefert forumSTAR ein TRUE zurück.
Persönlich	Persönliche Inhalte werden anhand des eindeutigen Schlüssels GPE_KEY_ID des Benutzers identifiziert.

Das folgende Schaubild verdeutlicht die Rechtezuordnung in forumSTAR im Zusammenhang mit eIP nochmals.

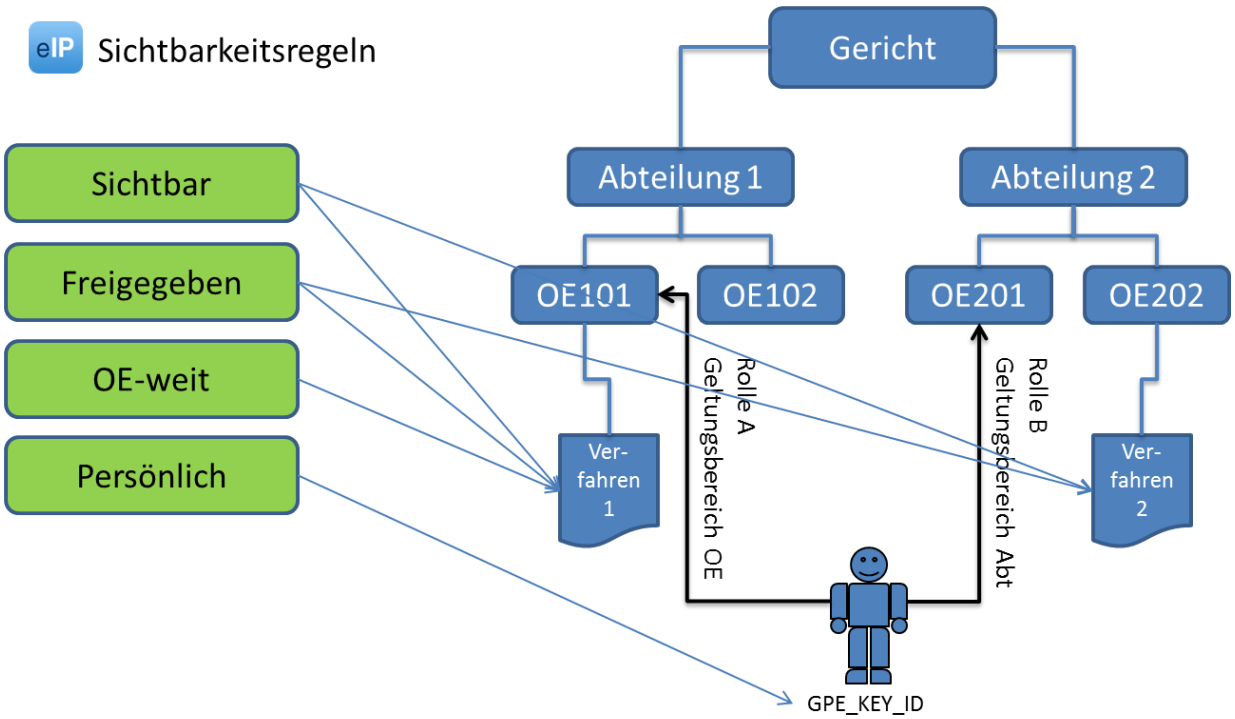

Abbildung 14-32: Organisationstruktur in forumSTAR und eIP Sichtbarkeitsregeln

.

15 ABBILDUNGS- UND TABELLENVERZEICHNIS

16 LITERATURVERZEICHNIS

Da die Projekt-Dokumente zurzeit teilweise in Bearbeitung sind und während der Laufzeit des Vorhabens "Pflege und Weiterentwicklung von forumSTAR und eJustice Basisdienste (eIP und eKP)" möglicherweise weiteren Aktualisierungen unterliegen, werden diese Dokumente ohne Version und Ausgabedatum aufgelistet. Für diese Dokumente ist immer die jeweils aktuelle akzeptierte Version zu verwenden.

/1/ Vertrag Los 2 "Pflege und Weiterentwicklung eJustice Basisdienste (eIP und eKP)", Freistaat Bayern vertreten durch das Bayerische Staatsministerium der Justiz, vom 24. Februar 2015

/2/ Ergänzende Vertragsbedingungen und Leistungsbeschreibung, Anlage 1 zu den Vergabebestimmungen „Pflege und Weiterentwicklung forumSTAR und eJustice Basisdienste (eIP und eKP)“, Freistaat Bayern vertreten durch das Bayerische Staatsministerium der Justiz, IT 5401 - 41/13

/3/ Vorgehensmodell V-Modell XT, Version 1.4

/4/ Abkürzungsverzeichnis und Glossar für das Vorhaben "Pflege und Weiterentwicklung von forumSTAR und eJustice Basisdienste (eIP und eKP)"