



Technische Spezifikation Konzeption VIDaL

BU BOS IT-Lösungen

Version	1.4
Stand	16. Okt. 2024
Status	Freigegeben
Autor	T-Systems Information Services GmbH

Schutzklasse: zur Veröffentlichung



Impressum

Herausgeber

T-Systems Information Services GmbH
BU BOS IT-Lösungen
10587 Berlin, Pascalstraße 11

Dateiname	Dokumentennummer	Dokumentenbezeichnung
ExpForum ViDaL - Technische Spezifikation V1.4.docx		Technische Spezifikation
Version	Stand	Status
1.4	16.10.2024	Freigegeben
Autor	Inhaltlich geprüft von	Freigegeben von
T-Systems Information Services GmbH	Alexander Kryukov	Frank Rottländer
Ansprechpartner	Telefon / Fax	E-Mail
Frank Rottländer	+49 228 181 73531	Frank.rottlaender@t-systems.com

Vorwort

Das Land Nordrhein-Westfalen plant den Aufbau einer hochverfügbaren Lageplattform zum automatisierten Informationsaustausch von meldepflichtigen Ereignissen und bei Großeinsatzlagen zwischen den Kreisen und den kreisfreien Städten, den Bezirksregierungen sowie dem Ministerium des Innern in Nordrhein-Westfalen. Damit sollen Daten digitalisiert und automatisiert ausgetauscht werden, die Meldewege vereinfacht und verkürzt werden. Somit soll die Verfügbarkeit eines aktuellen und umfassenden Lagebildes auf allen Ebenen des Landes NRW (Kreise, kreisfreie Städte, Regierungsbezirke, Ministerium des Innern) geschaffen werden. Das Ziel ist die Erhöhung der Effektivität und Effizienz im Brand- und Katastrophenschutz.

Dazu soll die Plattform ViDaL (Vernetzung von Informationen zur Darstellung der Landeslage) aufgebaut werden, um die Informations- und Berichtswege in der nicht-polizeilichen Gefahrenabwehr zu digitalisieren und zu automatisieren. Zugleich sollen die bisherigen Informationsinhalte um innovative Elemente mit dynamischen Daten ergänzt werden.

Mit dem Gesamtprojekts ViDaL werden folgende zentrale Komponenten konzipiert und aufgebaut:

- Lageplattform (LPF)
- Lagemodule
- Lagedatenbank (LDB)
- Lagedokumentationsdienst (LDD)

Inhaltsverzeichnis

1	Einleitung	11
2	Übergreifende Konzepte	12
2.1	Konzept AAA	12
2.1.1	Authentifizierung	12
2.1.2	Autorisierung.....	13
2.1.3	Einrichten eines Client-SSL-Zertifikats	14
2.2	Konzept E2E Verschlüsselung	15
2.2.1	Key Management.....	15
2.2.2	Anwendungsfälle.....	16
2.3	Konzept Adressierung	16
2.3.1	OID-Hierarchie	17
2.3.2	Gruppenadressen	17
2.3.3	OID-Typen	18
2.3.4	VIDaL OID-Nomenklatur	18
2.3.5	Identitätsmanagement.....	20
2.3.6	Ansatz Integration externer Systeme	20
2.4	Konzept IP Adressen und Netze	23
2.4.1	IP-Adresskonzept.....	23
2.5	Konzept IP Routing und Firewall	24
2.5.1	Routing	24
2.5.2	Firewall	24
2.5.3	Lageplattform	24
2.5.4	Systemmanagement	24
2.5.5	LDB Lagedatenbank	25
2.5.6	LDD Legendokumentationsdienst	26
2.5.7	Lagemodul	26
2.5.8	Lagemodul HA (generisch für den Einsatz im Rechenzentrum).....	27
3	Architekturkonzept	28
3.1	Softwarearchitektur	28
3.1.1	Systemübersicht	28
3.1.2	Vermittlungsebene	29
3.1.3	Anwendungsebene	29
3.2	Operatives Systemmanagement	30
3.2.1	Administrative Domänen	31
3.2.2	Administrative Aufgaben	31
3.2.3	Anwendungskonfiguration	31
3.3	Architekturentscheidungen.....	31

3.3.1	Schnittstelle für Kommunikationsteilnehmer	32
4	Spezifikation ViDaL-Anwendungen	34
4.1	Schemata – allgemeine Festlegungen	35
4.1.1	Einleitung	35
4.1.2	Namenkonvention	35
4.1.3	JSON-Objekte.....	35
4.1.4	Datentypeinschränkungen	36
4.1.5	Zusammenhängende Meldungen (Request/Response)	38
4.1.6	Fehlermeldungen	38
4.1.7	Validierung von Meldungen.....	38
4.1.8	Schema -Fehlermeldung – v1.0	42
4.2	Spezifikation Meldungen	43
4.2.1	Schema – Lageinformation – v1.0.....	43
4.2.2	Schema – Lageinformations-Abfrage -v1.0	55
4.2.3	Schema – Lageinformationsfehler – v1.0	56
4.3	Spezifikation Einheiten.....	58
4.3.1	Schema – Einheit – v1.0	58
4.3.2	Schema – Einheit-Referenz – v1.0.....	63
4.3.3	Schema – Einheitenfilter – v1.0.....	64
4.3.4	Schema – Einheitenübersicht – v1.0	65
4.3.5	Schema – Einheitenfehler – v1.0	67
4.4	Spezifikation Ressourcen.....	68
4.4.1	Schema – Ressource – v1.0	69
4.4.2	Schema – Ressource-Referenz – v1.0.....	80
4.4.3	Schema – Ressourcenfilter – v1.0.....	81
4.4.4	Schema – Ressourcenübersicht – v1.0	82
4.4.5	Schema – Ressourcenfehler – v1.0	85
4.5	Spezifikation Kontinuierliche Einsatzstatistik	86
4.5.1	Schema – Einsatzstatistik – v1.0.....	87
4.5.2	Schema – Einsatzstatistikfilter – v1.0	93
4.5.3	Schema – Einsatzstatistiken – v1.0.....	94
4.6	Spezifikation Adhoc-Information.....	96
4.6.1	Schema – Adhoc-Information – v1.0	96
4.7	Spezifikation Organisationen.....	97
4.7.1	Schema – Organisation – v1.0	97
4.7.2	Schema – Organisations-Referenz – v1.0.....	103
4.7.3	Schema – Organisationsfilter – v1.0.....	104
4.7.4	Schema – Organisationsübersicht – v1.0	105
4.7.5	Schema – Organisationsfehler – v1.0	107
4.8	Codelisten.....	108

4.8.1	Zusammenhang zwischen Meldung, Nachrichtenschema und Codeliste	109
4.8.2	Beispiel	110
4.8.3	Eingebettete Codelisten	112
4.8.4	Handhabung von komplexeren Codelisten.....	114
4.8.5	Verwaltung von Codelisten durch den Schema-Service	115
5	Spezifikation Lagemodul	117
5.1	Einleitung	117
5.2	LM ViDaL API v1.0	117
5.2.1	Spezifikation	117
5.2.2	Berechtigungskonzept.....	118
5.2.3	OpenAPI	118
5.2.4	ViDaL Registry API	118
5.2.5	ViDaL Schema API	119
5.2.6	ViDaL Group Management API.....	120
5.2.7	ViDaL Messaging API	121
5.3	LM im Netzwerk des KT	124
5.3.1	Einbau der Lagemodul Appliance.....	125
5.3.2	Netzwerkintegration	126
5.3.3	Informationen und Beistellungen des KT	126
5.3.4	Integration in das vorhandene Standortnetzwerk	126
5.4	LM Connection Test	129
5.4.1	Status der ViDaL Services	129
5.4.2	E2E Kommunikationstest	130
5.4.3	Kommunikation des lokalen Admins mit dem Connection Test	130
6	Spezifikation Lageplattform	131
6.1	LPF Einleitung.....	131
6.1.1	Bausteinsicht	131
6.2	LPF KT Register	131
6.2.1	LPF Registry Service	132
6.3	LPF Nachrichtenschema	135
6.3.1	LPF Schema Service	135
6.4	LPF Gruppenmanagement.....	137
6.4.1	LPF Group Service	138
6.5	LPF-Messaging.....	140
6.5.1	Spezifikation Kontinuierliche Einsatzstatistik	141
6.6	LPF Systemmeldungen	141
6.6.1	Schema – Systemmeldung	144
6.7	Operatives Systemmanagement	145
6.7.1	Administrative ViDaL-Funktionen	145

6.7.2	Authentifizierung / Autorisierung	146
6.7.3	Systemmeldungen	146
7	Spezifikation Lagedokumentationsdienst	147
7.1	Einleitung	147
7.1.1	Bausteinsicht	147
7.2	LDD Berechtigungskonzept.....	147
7.2.1	Zugriffsberechtigung für regulären KT	148
7.2.2	Zugriffsberechtigung für LDD-Administrator	148
8	Spezifikation Lagedatenbank	149
8.1	LDB Einleitung	149
8.1.1	Bausteinsicht	149
8.2	LDB Datenmodell - Einheit.....	150
8.2.1	ER-Modell	150
8.3	LDB Datenmodell – Resource.....	150
8.3.1	ER-Modell	150
8.4	LDB Datenmodell - Einsatzstatistik	151
8.4.1	ER-Modell	151
8.5	LDB Datenmodell - Organisation.....	152
8.5.1	ER-Modell	152
	Abkürzungsverzeichnis.....	154
	Änderungshistorie / Release Notes	155

Abbildungsverzeichnis

Abbildung 2.1: Konzept AAA Authentifizierung Anwendungsebene	12
Abbildung 2.2: Key Management.....	16
Abbildung 2.3: OID-Hierarchie.....	17
Abbildung 2.4: ViDaL OID-Nomenklatur	18
Abbildung 2.5: Synchronisation über Gateway	22
Abbildung 2.6: Synchronisation als ViDaL-Anwendung	23
Abbildung 2.7: Modulares IP-Layout.....	23
Abbildung 2.9: Lageplattform.....	24
Abbildung 2.10: Systemmanagement	25
Abbildung 2.11: LDB Lagedatenbank	25
Abbildung 2.12: LDD Lagedokumentationsdienst	26
Abbildung 2.13: Lagemodul.....	27
Abbildung 2.14: Lagemodul HA	27
Abbildung 3.1: Adapter-Muster	28
Abbildung 3.2: Überblick Vermittlungsebene	29
Abbildung 3.3: ViDaL Anwendung.....	30
Abbildung 3.5: Schnittstelle für Kommunikationsteilnehmer.....	32
Abbildung 4.1: Zusammenhang zwischen Meldung, Nachrichtenschema und Codeliste	109
Abbildung 5.1: Bausteinsicht Lagemodul.....	117
Abbildung 5.2: LM im Netzwerk des KT.....	125
Abbildung 5.3: Integration in das vorhandene Standortnetzwerk	127
Abbildung 5.4: Verwendung eines dedizierten WAN-Anschlusses.....	128
Abbildung 5.5: Anordnung ohne DMZ-Infrastruktur.....	128
Abbildung 5.7: LM Connection Test.....	129
Abbildung 6.1: Admin Webanwendung.....	131
Abbildung 7.1: Bausteinsicht LDD	147
Abbildung 8.1: Bausteinsicht LDB	149
Abbildung 8.2: ER-Modell Einheit	150
Abbildung 8.3: ER-Modell Resource.....	151
Abbildung 8.4: ER-Modell Einsatzstatistik.....	151
Abbildung 8.5: ER-Modell Organisation.....	152

Tabellenverzeichnis

Tabelle 2.1: Zertifikatsanfrage Zertifikatsattribute	15
Tabelle 2.2: OID-Typen	18
Tabelle 2.3: OID-Einzeladresse	19
Tabelle 2.4: OID-Gruppenadresse	20
Tabelle 4.1: Schema-Definition für das Beispiel	40
Tabelle 4.2: Beispiel Auswirkung der Schema-Definition-Regeln	41
Tabelle 4.3: Schema Fehlermeldung	42
Tabelle 4.4: Schema Lageinformation	52
Tabelle 4.5: Schema Lageinformations-Abfrage	56
Tabelle 4.6: Schema Lageinformationsfehler	57
Tabelle 4.7: Fehler-Codes Lageinformationsfehler	57
Tabelle 4.8: Schema Einheit	62
Tabelle 4.9: Schema Einheit-Referenz	64
Tabelle 4.10: Schema Einheitenfilter	65
Tabelle 4.11: Schema Einheitenübersicht	66
Tabelle 4.12: Schema Einheitenfehler	67
Tabelle 4.13: Fehler-Codes Einheitenfehler	68
Tabelle 4.14: Schema Ressource	79
Tabelle 4.15: Schema Ressource-Referenz	80
Tabelle 4.16: Schema Ressourcenfilter	82
Tabelle 4.17: Schema Ressourcenübersicht	82
Tabelle 4.18: Schema Ressourcenfehler	85
Tabelle 4.19: Fehler-Codes Ressourcenfehler	86
Tabelle 4.20: Schema Einsatzstatistik	89
Tabelle 4.21: Schema Einsatzstatistikfilter	94
Tabelle 4.22: Schema Einsatzstatistiken	94
Tabelle 4.23: Schema Adhoc-Information	96
Tabelle 4.24: Schema Organisation	102
Tabelle 4.25: Schema Organisationsreferenz	104
Tabelle 4.26: Schema Organisationsfilter	105
Tabelle 4.27: Schema Organisationsübersicht	106
Tabelle 4.28: Schema Organisationsfehler	107
Tabelle 4.29: Fehler-Codes Organisationsfehler	108
Tabelle 5.1: ViDaL Registry API für die Rolle KT	119
Tabelle 5.2: ViDaL-Schema API	120
Tabelle 5.3: ViDaL Group Management API	121
Tabelle 5.4: ViDaL Messaging API	123

Tabelle 5.5: Schema Zustellquittung.....	124
Tabelle 5.6: Status der ViDaL Services	129
Tabelle 5.7: E2E Kommunikationstest	130
Tabelle 6.1: LPF Registry API	135
Tabelle 6.2: LPF Schema API	136
Tabelle 6.3: Systemmeldungen im Zusammenhang mit dem Schemamanagement	137
Tabelle 6.4: Gruppenmanagement Berechtigungskonzept	138
Tabelle 6.5: Group API	140
Tabelle 6.6: Systemmeldungen Gruppenmanagement	140
Tabelle 6.7: LPF Systemmeldungen	144
Tabelle 6.8: Schema Systemmeldung	145

1 Einleitung

In diesem Dokument werden einzelne Systemkomponenten und Funktionen des ViDaL-Systems spezifiziert. Ausgehend von funktionalen und nicht-funktionalen Projektanforderungen werden systemübergreifende Entscheidungen getroffen, spezifische Eigenschaften von einzelnen Systemkomponenten festgelegt und Anwendungsprotokolle für ViDaL-Anwendungen spezifiziert.

2 Übergreifende Konzepte

Hier werden übergreifende, generelle Prinzipien und Lösungsansätze festgelegt, die in vielen Teilen der Systemarchitektur einheitlich benutzt werden. Konzepte beziehen sich oft auf mehrere Bausteine. Hier findet man Themen wie Domänenmodelle, Architekturmuster und -stile, Regeln zur Nutzung bestimmter Technologiestacks, etc.

2.1 Konzept AAA

2.1.1 Authentifizierung

2.1.1.1 Anwendungsebene

Die Kommunikation zwischen VIDaL-Subsystemen wird in beide Richtungen sowohl für fachliche Zugriffe als auch für das operative Systemmanagement mittels mTLS authentifiziert:

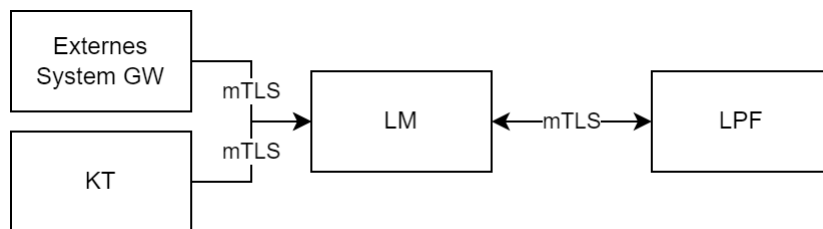


Abbildung 2.1: Konzept AAA Authentifizierung Anwendungsebene

Das Lagemodul (LM) erhält sein Client Zertifikat und den PublicKey des Servers (der Lageplattform - LPF) direkt beim Systemaufsetzen durch den Betreiber des VIDaL-Systems.

Zertifikate für Kommunikationsteilnehmer (KT, z.B. ELS/LMS) bzw. Gateways für externe Systeme werden auf Basis einer [Zertifizierungsanfrage](#) über die zentrale PKI (private key infrastructure) der LPF signiert. Der Flow ist:

1. T-Systems (als der Betreiber der Plattform) stellt die Funktionsmailbox Support_VIDaL@t-systems.com für das Onboarding der KT zur Teilnahme am VIDaL Verbund NRW zur Verfügung (FMB Support VIDaL).
2. Über diese FMB kann eine Anfrage geschickt werden unter Angabe folgender Informationen (ggf. identisch mit den Infos für die Ticketing System API):
 - a. Organisation
 - b. Ansprechpartner (Vor- und Zuname, dienstliche Mailadresse, dienstliche Anschrift, Telefonnummer für Rückfragen)
3. Nach der Prüfung der Anfrage richtet T-Systems im System einen KT-Account mit zugewiesener Test/Pilot-OID (das Projekt VIDaL hat später eine feste Basis-OID) ein. Die zugewiesene OID wird dem Antragsteller über E-Mail mitgeteilt.
4. Der Antragsteller erstellt eine [Zertifizierungsanfrage](#) (Certificate Sign Request - CSR) und sendet diese an die FMB Support VIDaL.
5. Nach der Prüfung der Zertifizierungsanfrage wird das Client-Zertifikat signiert und zusammen mit dem Server-Zertifikat über E-Mail an KT zurückgeschickt.
6. Der KT kann die VIDaL API unter Verwendung des mitgeteilten Client-Zertifikats nutzen (AuthN via mTLS).

Eine automatisierte Erneuerung von Zertifikaten wird derzeit noch erarbeitet. Eine entsprechende Aktualisierung des Dokuments und erforderliche Information an Entwickler externer Systeme ergeht in Kürze.

2.1.1.2 Systemmanagement Infrastruktur

Ein administrativer Zugriff auf die Infrastruktur ist für den Systembetrieb, das Monitoring und das Reporting der Servicequalität erforderlich. Der Zugriff wird den System-Administratoren und den Entwicklern der Applikation über eine Zugangs-Infrastruktur, bestehend aus Sprungservern und Dateiaustausch-Hosts zur Verfügung gestellt.

Dabei wird die passwortlose SSH-Authentifizierung verwendet, basierend auf einem SSH-Schlüsselpaar, das aus einem öffentlichen Schlüssel und einem privaten Schlüssel besteht.

2.1.2 Autorisierung

2.1.2.1 Anwendungsebene

Im Rahmen der Kommunikation zwischen den einzelnen Systemkomponenten gibt es keine Notwendigkeit für die Entwicklung eines Rollen-Rechte-Konzeptes, da einzelne Kommunikationspartner wie KT und LM grundsätzlich gleichberechtigt sind. Lediglich im Kontext der generellen Dienste der LPF für alle ViDaL-Anwendungen gleichermaßen gibt es spezifische Autorisierungsregeln. So darf eine Kommunikationsgruppe nur durch den KT "Gruppenbesitzer" verwaltet werden.

Beim operativen Systemmanagement gibt es verschiedene administrative Tätigkeiten wie

- Modellierung / Anpassung von Geschäftsprozessen
- ViDaL-Abläufe konfigurieren / steuern (Beispiel: Nachrichtenversand an bestimmte KT aussetzen)
- Administrative Mitteilungen senden
- Konfigurieren von Laufzeitaspekten der Anwendung wie Clustering, Verteilung, Skalierung, etc.
- Betriebsüberwachung

Das Systemmanagement bietet Möglichkeiten zur Steuerung der administrativen Zugriffsberechtigungen für die Systemkomponenten LPF, LM, LDD, LDB (Rechte- und Rollenkonzepte).

Es werden 3 Admin-Domänen umgesetzt, um das System Management von folgenden Systemkomponenten zu trennen:

- LPF und LM: Es gibt keine Unterschiede in Rechte- und Rollenkonzepten zwischen einzelnen LM. LPF ist der Identity & Access Management Provider (IAM-Provider) für die gemeinsame Admin-Domäne von LPF und LM
- Die LDB hat eine eigene Admin-Domäne
- Die LDD hat eine eigene Admin-Domäne

Es wird ein Standardmodell für rollenbasierte Zugriffskontrolle (Role Based Access Control Model, RBAC-Modell) implementiert: User, Admin. Bei Bedarf können weitere Rollen definiert werden.

Das Systemmanagement für die Systemkomponenten LPF und LM erfolgt zentral aus dem Systemmanagement für die LPF heraus.

Die Systemmanagements für die Systemkomponenten LDD und LDB bekommen jeweils einen separaten Zugang.

2.1.2.2 Systemmanagement Infrastruktur

Denkziele Personen - Systemadministratoren - werden auf der Zugangs-Infrastruktur administriert und können dadurch auf die administrativen Funktionen für den Systembetrieb, das Monitoring und das Reporting der Servicequalität zugreifen.

2.1.3 Einrichten eines Client-SSL-Zertifikats

In diesem Kapitel wird beschrieben, wie der Administrator eines KT ein Client-SSL-Zertifikat erstellen und durch die VIDaL PKI-Infrastruktur signieren kann. Dieses Zertifikat ist die Voraussetzung für die Kommunikation mit dem VIDaL Lagemodul.

Das Einrichten eines SSL-Zertifikats besteht aus den folgenden Schritten:

1. Schlüsselpaar generieren
2. Zertifizierungsanfrage erstellen
3. Signiertes Zertifikat in KeyStore speichern

Es gibt viele Tools, die das Erstellen von Zertifikaten ermöglichen und die damit verbundenen Aktivitäten (Erstellen von Schlüsselpaaren, Erstellen von Zertifizierungsanfragen, Signieren von Inhalten, etc.) unterstützen. Einige davon sind:

- [KeyStore Explorer](#)
- Windows-Tool `certreq` - unterstützt Templates, die von T-Systems geliefert werden können
- openssl - unterstützt Templates, die von T-Systems geliefert werden können

2.1.3.1 Schlüsselpaar generieren

Ein Schlüsselpaar beinhaltet zwei miteinander verbundene Schlüssel - den öffentlichen Schlüssel (Public Key) und den privaten Schlüssel (Private Key). Die Schlüssel und dazugehörige Zertifikate werden in einem Schlüsselspeicher (KeyStore) verwaltet.

2.1.3.1.1 Schlüsselspeicher anlegen

Die VIDaL-PKI arbeitet mit einem Schlüsselspeicher vom Typ Java KeyStore (JKS).

2.1.3.1.2 Schlüsselpaar generieren

Die VIDaL-PKI unterstützt RSA-Verschlüsselung. Als Schlüssellänge wird mindestens eine Länge von 4096 Bit vorausgesetzt.

2.1.3.2 Zertifizierungsanfrage erstellen

2.1.3.2.1 Zertifikat anlegen

Das Zertifikat wird als SSL-Zertifikat Version 3 (SSLv3) erwartet. Das Zertifikat muss mit dem Signaturalgorithmus SHA-256 mit RSA und einer Gültigkeitsdauer von 2 Jahren erzeugt werden.

Folgende weiteren Zertifikatsattribute sind zu befüllen:

Attribute	Bedeutung	Beispiel
CN	KT OID (siehe Anlage VIDaL OID-Nomenklatur unten)	1.2.3.1.276.5.1.1.58.28.1.1 Die OID wird einem KT durch den VIDaL-Plattformbetreiber im Onboarding-Prozess zugewiesen.
O	Organisation (OperatorName)	Krisenstab Ratingen
OU	Organisation/Abteilung (SystemName)	LMS Ratingen
E	E-Mail	support@lms-ratingen.de
L	Stadt / Ort	Ratingen
ST	Bundesland	Nordrhein-Westfalen
C	Land, ISO-Kürzel	DE

Tabelle 2.1: Zertifikatsanfrage Zertifikatsattribute

2.1.3.2.2 Anfrage für Zertifikatsignierung erstellen

Eine Zertifikatsignierungsanfrage (CSR) ist eine einfache Text-Datei. Sie beinhaltet den öffentlichen Schlüssel des KT und kann auf einem öffentlichen Weg (z.B. per E-Mail) versendet werden.

Die VIDaL-PKI arbeitet mit CSR in PKCS#10 Format.

Die Zertifikatsignierungsanfrage wird an die Funktionsmailbox (FMB) für das Onboarding der KT gesendet.

2.1.3.3 Signiertes Zertifikat in KeyStore speichern

Als Antwort auf eine Zertifikatsignierungsanfrage wird eine Antwort (CA-Antwort) auf einem öffentlichen Weg (z.B. per E-Mail) geliefert, die ein signiertes Zertifikat beinhaltet. Dieses Zertifikat wird automatisiert in den Schlüsselspeicher importiert.

2.2 Konzept E2E Verschlüsselung

Die Kommunikation zwischen den KT ist Ende-zu-Ende (E2E), d.h. von LM zu LM, mit symmetrischen Schlüsseln verschlüsselt.

2.2.1 Key Management

Der symmetrische Schlüssel wird zentral in der Lageplattform (LPF) generiert und an registrierte LM verteilt.

Von einem generierten Schlüssel wird ein Hash-Wert gebildet (Schlüssel-ID) und eine Liste von Secrets (mit dem PubKey des jeweiligen LM verschlüsselter Schlüssel) für jedes registrierte LM erzeugt:

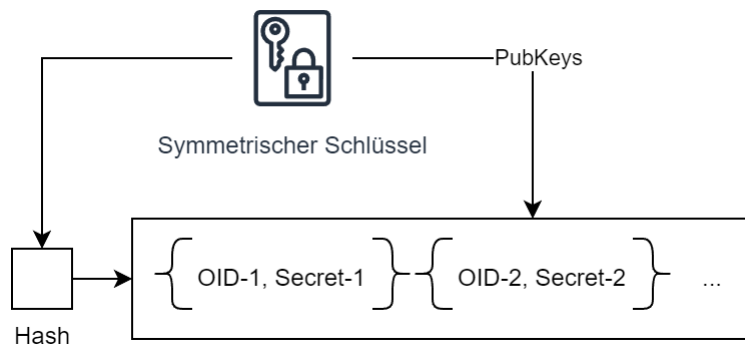


Abbildung 2.2: Key Management

Damit liegt der symmetrische Schlüssel zu keiner Zeit in der LPF unverschlüsselt oder de-chiffrierbar vor. Somit wird gewährleistet, dass der symmetrische Schlüssel auf der LPF nicht eingesehen werden kann, auch nicht durch Administratoren.

Ein neuer symmetrischer Schlüssel wird durch die LPF in folgenden Fällen generiert:

- regelmäßig nach einem durch einen LPF-Administrator konfigurierbaren Zeitintervall (z.B. jeden Tag / Stunde / etc.)
- manuell durch den LPF-Administrator
- jedes Mal, wenn ein neues LM registriert wird (damit für jedes LM ein Secret vorliegt)

2.2.2 Anwendungsfälle

2.2.2.1 Lagemodul registrieren

Registrierungs-Request (Admin API) führt zur Generierung eines neuen Schlüssels, siehe oben, und das LM-Secret wird zurückgegeben.

2.2.2.2 Nachricht senden

Das LM in der Rolle eines Senders verwendet zur Verschlüsselung einer Nachricht den ihm vorliegenden symmetrischen Schlüssel. Mit der Nachricht wird auch die Schlüssel-ID als Metadaten mit versendet.

Ist der Schlüssel nicht mehr aktuell (weil auf der LPF bereits ein neuer Schlüssel generiert wurde), so bekommt der Sender einen entsprechenden Fehler-Response mit dem neuen aktuellen Secret. Daraufhin kann die Nachricht mit dem neuen Schlüssel verschlüsselt und neu versendet werden.

2.2.2.3 Nachricht empfangen

Zusammen mit der Nachricht bekommt der Empfänger auch sein Secret mit dem die Nachricht entschlüsselt werden kann.

2.3 Konzept Adressierung

Für die Adressierung der einzelnen Kommunikationsteilnehmer (KT) in der Plattform ViDaL werden eindeutige Kennungen in Form von Object Identifier (OID Spezifikationen ISO/IEC 9834, DIN 66334) verwendet. Siehe Fachliches Feinkonzept - Adressierung, Anforderung 1.

2.3.1 OID-Hierarchie

Die Adressierung von einzelnen ViDaL Systemkomponenten ist hierarchisch organisiert und spiegelt die hierarchische Kommunikationsstruktur in ViDaL wider:

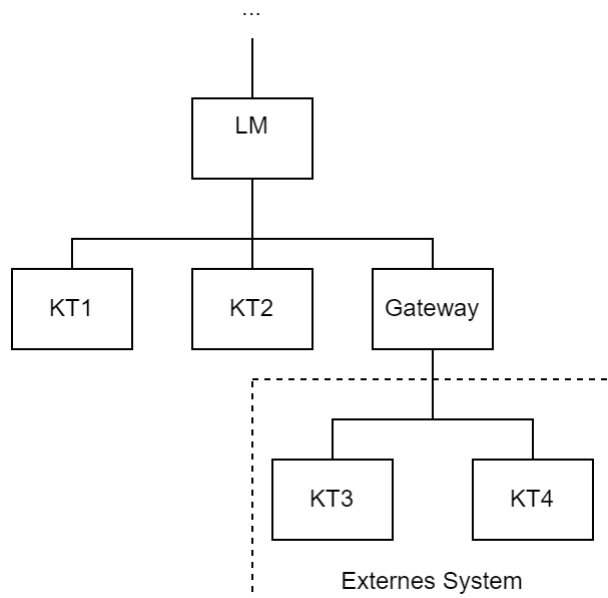


Abbildung 2.3: OID-Hierarchie

Diese Struktur ermöglicht die Implementierung einer einfachen und einheitlichen Routing-Funktion, die in jeder ViDaL-Systemkomponente (LPF, LM, Gateway) die Weiterleitung von übermittelten Nachrichten unterstützt.

Auch wenn die Bildung einer eigenen Adressierungsebene für Lagemodule (LM-Ebene im Bild) nicht zwingend notwendig ist, unterstützt diese Ebene das Implementieren einer uniformen Routing-Funktion. Zusätzlich bringt die direkte OID-Adressierung der Lagemodule folgende Vorteile:

- Ermöglichung der Umsetzung nützlicher technischer Funktionen, z.B. Kommunikationsdurchstich Modul zu Modul unter Verwendung von einheitlichen Adressierungs- und Routing-Funktionen.
- Unterstützung der Vereinheitlichung von Identity-Management für ViDaL-Systemkomponenten.

Die Systemkomponente Gateway stellt eine spezielle Form eines KT dar. Das Gateway wird am Übergang zu externen Systemen eingesetzt und stellt eine Gateway-Funktion bereit zum Mapping zwischen externe Quell- bzw. Zieladressen und den internen OID. Ein externes System bekommt dabei einen entsprechend reservierten OID-Bereich (Unterbaum) und das Gateway bekommt die Wurzel-OID-Adresse dieses Unterbaums. Siehe auch Fachliches Feinkonzept - Adressierung, Anforderung 4.

Die Wurzel-Adresse eines OID-Unterbaums kann grundsätzlich auch als Gruppenadresse für alle OID-Knoten in einem Unterbaum verwendet werden. Im Rahmen des ViDaL-Projektes wird diese Möglichkeit nicht genutzt.

2.3.2 Gruppenadressen

Für die Gruppenadressen (Adressierung von mehreren KT's, siehe Fachliches Feinkonzept - Adressierung, Anforderung 3) wird auch die OID-Nomenklatur verwendet.

Das ermöglicht ein einheitliches Adressenformat für Sender und Empfängerinformationen und vereinfacht die Adressierungslogik.

Hinter einer Gruppenadresse wird effektiv eine Liste von OIDs verwaltet.

Die Unterscheidung zwischen den im Fachlichen Feinkonzept - Adressierung, Anforderung 3 erwähnten statischen und dynamischen Gruppen ist rein organisatorischer Natur und liegt auf der Seite der Nutzer einer Gruppe.

2.3.3 OID-Typen

Es gibt folgende OID-Typen (siehe OID-Hierarchie Beispiel unten):

#	OID-Typ	Verwendung
1	OID-Endknoten	KT-Endadresse (nach AGS-Systematik), Adressierung von einzelnen KT wie ELS, LMS, LDD, LDB
2	OID-Unterbaum	Adresse eines OID-Gateways, der für alle OID-Knoten in einem OID-Unterbaum zuständig ist - wird für das Routing von Nachrichten an externe Systeme verwendet Gruppenadresse für alle OID-Knoten in einem OID-Unterbaum (vgl. AP 3 Kap. 3.8.4) wird nicht verwendet.
3	OID-Liste	Gruppenadresse für statische zusammengesetzte OID-Gruppen - Adressierung von mehreren KTs Gruppenadresse für dynamische, im Rahmen einer Lage zusammengesetzte OID-Gruppen - Adressierung von mehreren KTs

Tabelle 2.2: OID-Typen

2.3.4 ViDaL OID-Nomenklatur

ViDaL OID-Nomenklatur unterscheidet zwei Satzarten: Einzeladressen der Kommunikationsteilnehmer (Wurzel-Adressen von OID-Unterbäumen fallen auch in diese Kategorie) und Gruppenadressen:

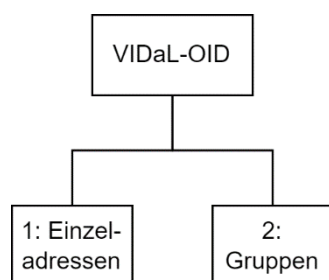


Abbildung 2.4: ViDaL OID-Nomenklatur

2.3.4.1 OID-Einzeladresse

Die ViDaL OID-Nomenklatur soll die Adressierung von KT über die staatlichen Grenzen hinaus ermöglichen. Dabei steht jedem Land frei, ein eigenes Adressierungsschema unterhalb des Landes-OID-Unterbaums zu definieren. Die Wurzel-Adresse eines Landes-OID-Unterbaums ist wie folgt definiert:

- <ViDaL-OID>.1.<Kennzahl des Landes> (Beispiel: 1.2.3.1.276 für Deutschland)

Auch jedes Bundesland in Deutschland ist frei, ein eigenes Adressierungsschema unterhalb des Bundesland-OID-Unterbaums zu definieren. Die Wurzel-Adresse eines Bundesland-OID-Unterbaums ist wie folgt definiert:

- <VIDaL-OID>.1.276.<Kennzahl des Bundeslandes> (Beispiel: 1.2.3.1.276.5 für NRW)

Unterschiedliche Organisationen können auch unterschiedliche Adressierungsschemata verwenden. Die Wurzel-Adresse eines Organisations-OID-Unterbaums für nichtpolizeiliche Gefahrenabwehr (eines nPOLGA-OID-Unterbaums) ist wie folgt definiert:

- <VIDaL-OID>.1.276.5.<Kennzahl von POL/nPOLGA, etc.> (Beispiel: 1.2.3.1.276.5.1 für nPOLGA in NRW)

Die OID-Adresse der einzelnen Kommunikationsteilnehmer (KT) in der Plattform VIDaL in NRW wird nach dem amtlichen Gemeindeschlüssel (AGS) gebildet (siehe Fachliches Feinkonzept - Adressierung, Anforderung 2):

#	Ebene	Beispiel
1	VIDaL-OID	1.2.3 - Festlegung in einem geschlossenen VIDaL-System
2	Satzart	1 - Einzeladresse
3	Kennzahl des Landes	276 - Deutschland nach ISO 3166
4	Kennzahl des Bundeslandes	5 - für NRW nach AGS
5	Kennzeichnung von POL/nPOL Gefahrenabwehr (KRITIS, etc.)	Polizeidienststellen, Gesundheitsämter, Veterinärämter, etc. Festlegungen: 0 - VIDaL Systemadressen, z.B. Lageplattform als KT für Systemmeldungen 1 - nPOLGA 99 - Test/Pilot Weitere Definitionen folgen.
6	Kennzahl des Regierungsbezirks	0 - Zentrale Adressen ohne Regierungsbezirksangabe 1 - Regierungsbezirk Düsseldorf nach AGS
7	Kennzahl des Landkreises oder der kreisfreien Stadt	0 - Zentrale Adressen ohne Landkreisangabe 58 - Landkreis Mettmann nach AGS
8	Gemeinde ("0" bei kreisfreien Städten)	0 - Zentrale Adressen ohne Gemeindeangabe oder kreisfreie Städte 28 - Stadt Ratingen, Stadtbezirk Ratingen nach AGS
9	Lagemodul	1 - erstes Lagemodul, fortlaufende Nummerierung von Lagemodulen
10	Kommunikationsteilnehmer	1 - z.B. ELS, fortlaufende Nummerierung von KT

Tabelle 2.3: OID-Einzeladresse

Beispiel: OID Feuerwehr ELS in Ratingen: 1.2.3.1.276.5.1.1.58.28.1.1

Info: Lagemodul OID ist eine logische Adresse bei der HA-Infrastruktur

2.3.4.2 OID-Gruppenadresse

Aufbau einer OID-Gruppenadresse:

#	Ebene	Beispiel	Kommentar
1	VIDaL-OID	1.2.3 - Festlegung in einem geschlossenen VIDaL-System	
2	Satzart	2 - Gruppenadresse	
3	Jahr	2023 - das Entstehungsjahr	Es ist eine 3 bis maximal 4-stellige Zahl an dynamischen Gruppen pro Jahr zu erwarten.
4	Gruppennummer	42 - fortlaufende Gruppennummerierung	

Tabelle 2.4: OID-Gruppenadresse

Beispiel einer Gruppe-OID: 1.2.3.2.2023.42

2.3.5 Identitätsmanagement

Zur Unterstützung des Identitätsmanagements wird die OID zu einem festen Bestandteil der Identität einer VIDaL-Komponente (LM, KT), z.B. als Teil des Client-Zertifikats:

- Die OID wird beim Erstellen von Client-Zertifikaten bzw. beim Zertifikatstausch berücksichtigt.
- Das ermöglicht ein automatisiertes Pflegen von Routing-Tabellen, die bei Meldungsübermittlung verwendet werden.
- Das ermöglicht die automatische Ermittlung der Sender-Adresse (KT-OID).

Pro territoriale Einheit aus dem AGS können beliebig viele Lagemodule aufgesetzt werden. Jedes Lagemodul bekommt eine OID-Adresse nach dem Schema (siehe OID-Einzeladresse oben):

<OID Ebenen 1 bis 8>.<fortlaufende LM-Nummer>

Beispiel OID des zweiten Lagemoduls in Ratingen: 1.2.3.1.276.5.1.1.58.28.2

Die OID eines KT ergibt sich aus der OID des entsprechenden Lagemoduls nach dem Schema

<LM OID>.<fortlaufende KT-Nummer>

Beispiel OID des ersten KT angeschlossen an dem zweiten Lagemodul in Ratingen:
1.2.3.1.276.5.1.1.58.28.2.1

Das VIDaL-System hat keine Information über den Typ des angeschlossenen KT.

2.3.5.1 Onboarding Kommunikationsteilnehmer

Zum Onboarding der KT wird ein Prozess eingerichtet, der zum einen das Aufsetzen eines Lagemoduls und zum anderen das Einrichten eines KT-Accounts ermöglicht.

2.3.6 Ansatz Integration externer Systeme

Vorwort:

Dieser Inhalt ist als Konzeptskizze für spätere Weiterentwicklungen gedacht.

Erfahrungen aus dem Pilotbetrieb sind auszuwerten und entsprechende Änderungen umzusetzen. Die möglicherweise erforderlichen Änderungen werden in der API-Spezifikation zu diesem Zeitpunkt aktualisiert werden.

Um zwei VIDaL Systeme (z.B. Systeme in zwei verschiedenen Bundesländern) zu integrieren, müssen folgende drei Probleme gelöst werden:

- Routing (Empfangen und Weiterleiten von Meldungen) zwischen beiden VIDaL-Systemen. Adressierbare Systemkomponenten eines Fremdsystems (KT, Lagemodule) können infrastruktur-technisch nicht direkt erreicht werden.
- Transformation zwischen landesspezifischen Informationsobjekten - mindestens OID-Adresstransformationen, eventuell aber auch Transformationen von anderen Informationsobjekten wie landesspezifische Codelisten, etc.
- Synchronisation zwischen landesspezifischen Konzepten, wie KT-Registrierungsinformationen, Standortadressen, etc. Dabei wird beispielsweise das KT-Register um KT-Informationen des externen Systems automatisch und dynamisch erweitert.

Sämtliche VIDaL-Schemata stellen dabei grundsätzlich landesübergreifende Konzepte dar und müssen bei der Integration von VIDaL-Systemen nicht behandelt werden.

Zur Lösung von den ersten zwei Problemen wird eine spezielle Systemkomponente Gateway eingesetzt. Das Gateway wird am Übergang zwischen beiden VIDaL-Systemen eingesetzt und wird in beiden Systemen registriert. In beiden Systemen bekommt das jeweilige Fremdsystem einen entsprechend reservierten OID-Bereich (Unterbaum) und das Gateway bekommt die Wurzel-OID-Adresse dieses Unterbaums. Somit ist das Gateway ein spezieller KT in beiden Systemen.

Dabei übernimmt das Gateway folgende Aufgaben:

- Routing (Empfangen und Weiterleiten von Meldungen) zwischen beiden VIDaL-Systemen
- OID-Adresstransformationen zwischen beiden Systemen. Im einfachsten Fall (das Adressierungskonzept des Fremdsystems basiert auch auf OID-Grundlagen) kann die Adresse eines Fremd-KT als Kombination aus der Gateway-Heim-OID und der eigentlichen KT-OID im Fremdsystem zusammengesetzt werden. In diesem Fall muss das Gateway ein entsprechendes OID-Prefix (Gateway-Heim-OID) bei der Adresstransformation einfach abschneiden. Verwendet dagegen das Fremdsystem andere Adressierungskonzepte, so muss das Gateway eine komplexere Adressen-Abbildungslogik umsetzen.
- Eventuell ein Mapping von anderen Informationsobjekten wie landesspezifische Codelisten, etc.

2.3.6.1 Synchronisation über Gateway

Das Gateway kann auch zur Lösung des dritten Problems eingesetzt werden: Synchronisation von landesspezifischen Konzepten zwischen VIDaL-Systemen. Dazu bekommt das Gateway gesonderte eingeschränkte Rechte, Informationen in einem genau spezifizierten Bereich für das externe System (Sandbox-Konzept) über die Service-API zu pflegen. Vorteil: Verwendung von existierenden Service-API, Nachteil: Sonderrechte (sprich Administratorrechte) für das Gateway notwendig, die über die Rechte eines Regel-KT hinausgehen. Dazu muss ein Sandbox-Konzept entwickelt werden.

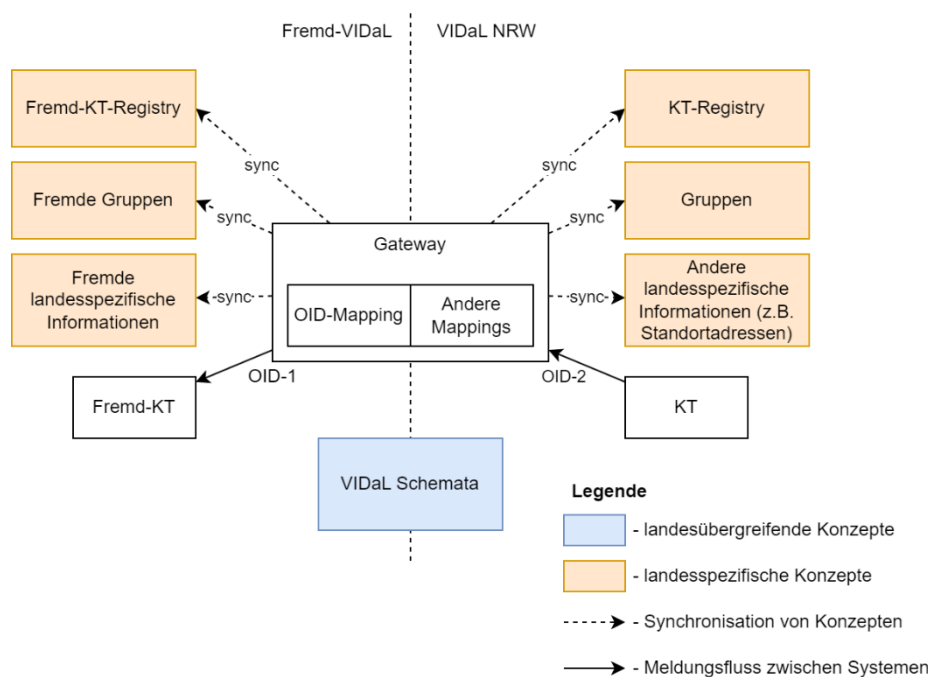


Abbildung 2.5: Synchronisation über Gateway

2.3.6.2 Synchronisation als VIDaL-Anwendung (empfohlen)

Die Synchronisation zwischen landesspezifischen Konzepten kann auch über eine spezielle VIDaL-Anwendung (spezifische Nachrichtenschemata zum Synchronisieren von Informationen) umgesetzt werden. Dabei tauschen die Lageplattformen beider Systeme Synchronisationsmeldungen aus und pflegen entsprechende landesspezifische Informationen aus dem externen System in eigene Informationsquellen ein. Diese VIDaL-Anwendung (Synchronisationsmeldungen) reiht sich prinzipiell in die Domäne Systemmeldungen ein. Der Vorteil liegt in einem besseren kontinuierlichen Sicherheitskonzept, da das Gateway in diesem Fall rechte-technisch ein regulärer KT ist und somit keine Sonderrechte benötigt; die LPF dagegen ist die Systemkernkomponente und von Natur her im Besitz von administrativen Rollen. Es ist allerdings die Definition einer spezifischen VIDaL-Anwendung "Synchronisation" mit dazugehörigen Nachrichtenschemata erforderlich.

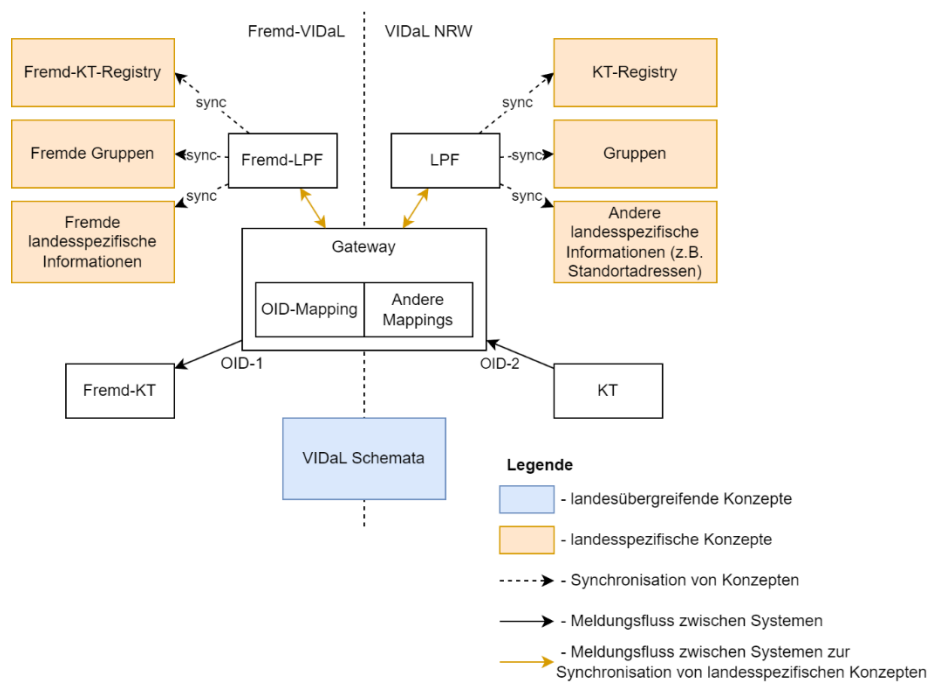


Abbildung 2.6: Synchronisation als ViDaL-Anwendung

2.3.6.3 Kommunikationsgruppen

Gruppenadressen aus dem externen System werden genauso behandelt wie Einzeladressen (z.B. mit Prefix des Gateways vorgesehen) und synchronisiert.

Der Gruppenbesitz kann nicht über die Systemgrenze hinaus gewechselt werden.

2.4 Konzept IP Adressen und Netze

2.4.1 IP-Adresskonzept

2.4.1.1 Modulares IP-Layout

10	Lokation	NTyp	Zähler	Netz
8 Bit	13 Bit	3 Bit	4 Bit	4 Bit
10
10.x.y.0/21 pro Lokation		8*16*16 IP-Adressen		

Abbildung 2.7: Modulares IP-Layout

Wir nutzen den 10er IP-Bereich.

Wir kodieren die Lokation und den Netztyp in den Prefix einer jeden IP-Adresse.

Für die Lokation stehen $2^{13} = 8196$ Möglichkeiten zur Verfügung.

Für den Netztyp stehen $2^3 = 8$ Möglichkeiten zur Verfügung.

Für die Lokationen und deren Kodierung(en) wird eine Liste gepflegt als json-File.

2.5 Konzept IP Routing und Firewall

2.5.1 Routing

Es wird nur statisches Routing eingesetzt. Alle Systembestandteile sind durch das IP-Konzept beschrieben.

2.5.2 Firewall

Es werden virtuelle pfSense Firewalls in der Community Edition Lizenz eingesetzt. Die Firewalls weisen folgende Eigenschaften auf:

- pfSense Version 2.70 basierend auf dem Betriebssystem FreeBSD 14
- Standardkonfiguration: Active-Passive Cluster (außer Lagemodul in der Lokation ohne HA)
- Virtuelle IPs durch CARP (Common Address Redundancy Protocol)
- Standard 4 virtuelle NICs in VMWare - WAN, LAN, MGMT und HA
- zusätzliche Interfaces auf VLAN-Trunks an den 4 NICs
- Benennung der logischen Interfaces nach VLANs und Modul

2.5.3 Lageplattform

Die Lageplattform bindet über IPSec-Tunnel für jedes Lagemodul jeweils einen Applikations- und einen Managementtunnel an.

Die Plattform selbst ist kein Lagemodul, sondern nur der Vermittler.

Die Plattform liefert das Managementnetz für alle Lagemodule.

Zwischen den Plattformknoten und damit zwischen den zentralen Standorten werden IP-Sec-Tunnel durch Transfernetze ersetzt.

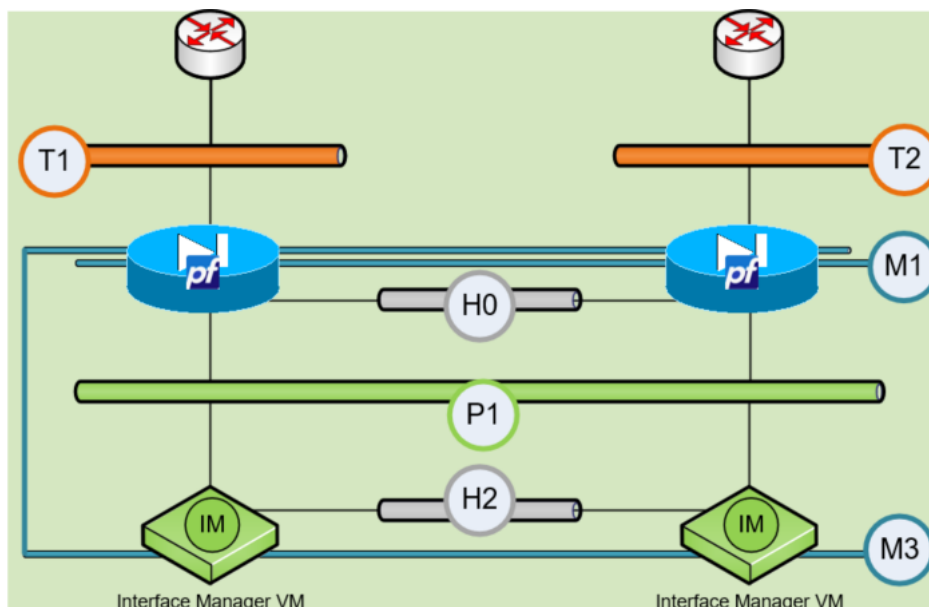


Abbildung 2.8: Lageplattform

2.5.4 Systemmanagement

Das Systemmanagement ist in VLANs nach den funktionalen Einheiten segmentiert.

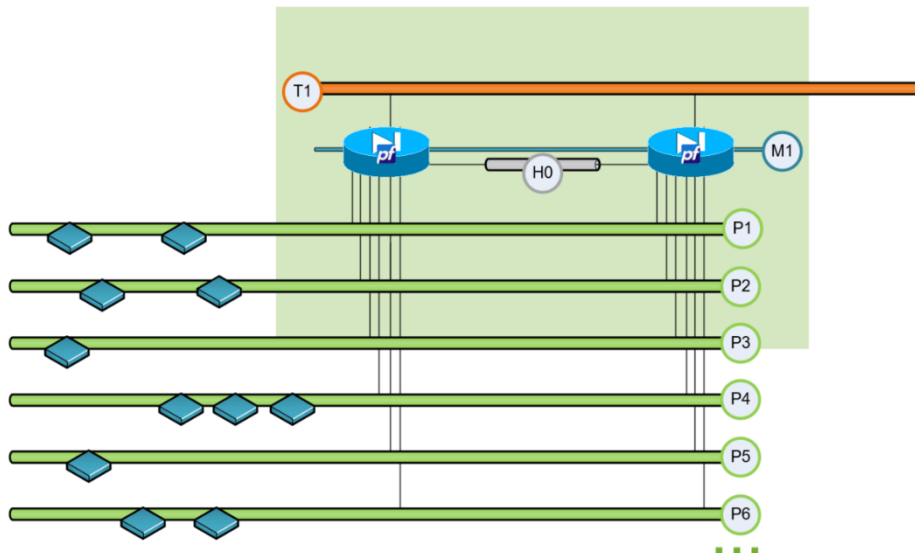


Abbildung 2.9: Systemmanagement

2.5.5 LDB Lagedatenbank

Die Lagedatenbank besteht aus einem Lagemodul-HA (high availability) und der Datenbank-Applikation mit SQL-Server in HA.

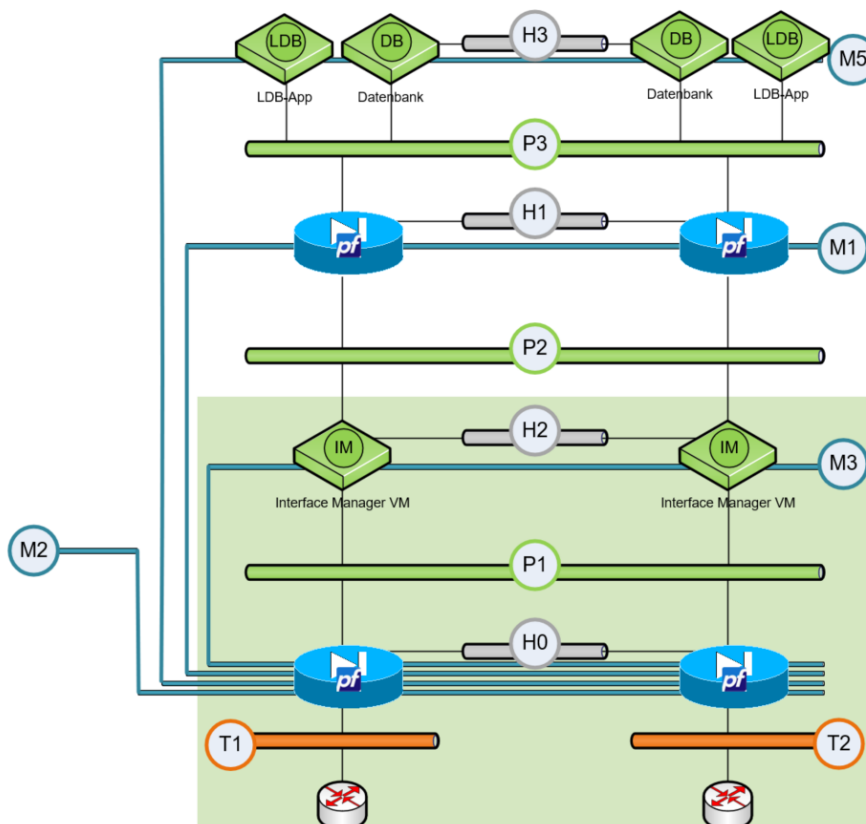


Abbildung 2.10: LDB Lagedatenbank

2.5.6 LDD Lagedokumentationsdienst

Der Lagedokumentationsdienst besteht aus einem Lagemodul-HA, der LDD-Applikation in HA und dem Netapp-Storage.

Der Netapp-Storage ist nicht redundant auf Plattformebene. Durch zwei LDD-Module wird die Verfügbarkeit hergestellt.

Die LDD-App speichert bei Ausfall des Netapp-Storage die Datenpakete zwischen.

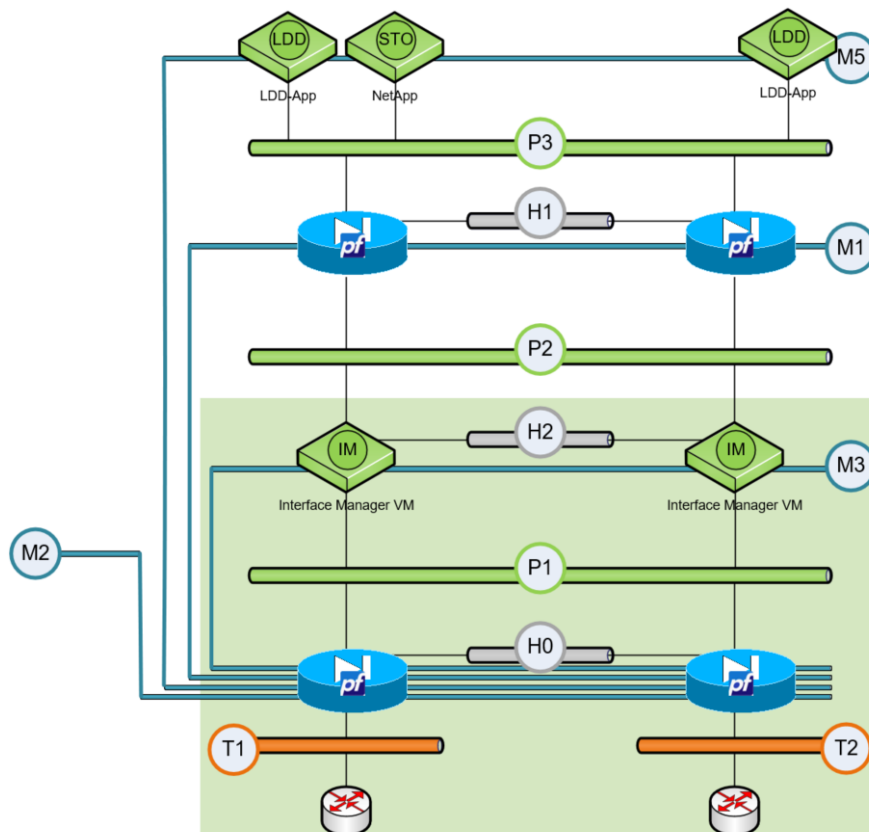


Abbildung 2.11: LDD Lagedokumentationsdienst

2.5.7 Lagemodul

Das Lagemodul bindet eine Lokation außerhalb des Rechenzentrums an die VIDAL-Plattform an. Dabei bildet das LM ein Application Gateway, das die Netze den KT und der LPF voneinander trennt. Die Verbindung zum WAN ist auf zwei getrennte Interfaces zur Verbindung an zwei Lageplattformknoten via IPsec aufgeteilt.

Innerhalb des Lagemoduls ist ein Test-Client implementiert, der über SSH erreicht werden kann. Dieser stellt eine Kommandozeilenschnittstelle mit speziellen Befehlen bereit, über die einfache Verbindungs- und Funktionstests durch den jeweils örtlichen Systemadministrator zur Überprüfung der Funktionsfähigkeit des Lagemoduls und der Anbindungen an LPF und KT-System durchgeführt werden können. Dieser Test-Client kann auch durch die zentralen LPF-Administratoren verwendet werden.

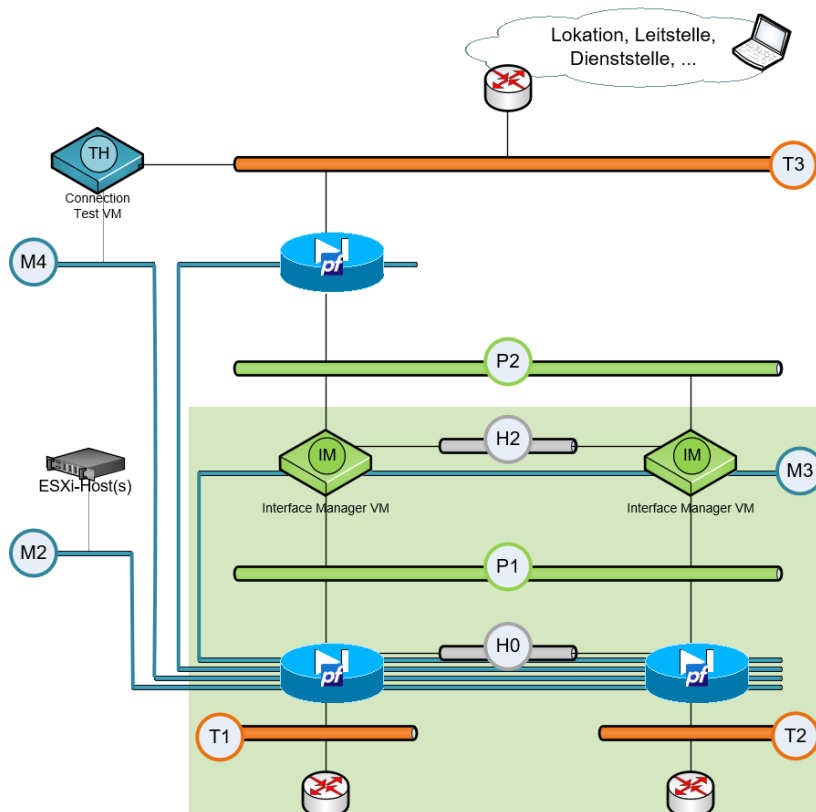


Abbildung 2.12: Lagemodul

2.5.8 Lagemodul HA (generisch für den Einsatz im Rechenzentrum)

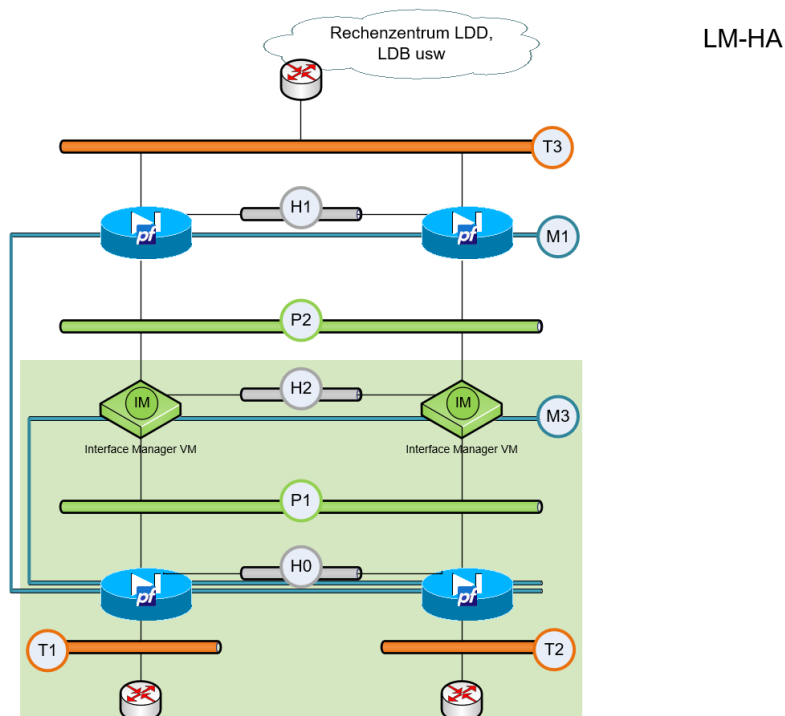


Abbildung 2.13: Lagemodul HA

3 Architekturkonzept

3.1 Softwarearchitektur

3.1.1 Systemübersicht

3.1.1.1 Broker-Muster

Bei der Strukturierung des VIDaL-Systems wird das Architekturmuster *Broker* verwendet. Bei diesem Architekturmuster kommunizieren verteilte unabhängige Systemkomponenten (Clients und Server - im Falle VIDaL sind das Kommunikationsteilnehmer - KT) miteinander mit Hilfe von Nachrichten. Eine zentrale Komponente (Broker - im Falle des VIDaL-Systems ist das die Lageplattform - LPF) spielt dabei die Rolle eines Vermittlers. Der Vermittler koordiniert die Kommunikation zwischen den Clients und dem Server, indem er die Client-Requests entgegennimmt und entlang einer verteilten System-Infrastruktur an den entsprechenden Server zustellt. Im Gegenzug finden die Server-Responses sowie eventuelle Fehlermeldungen durch den Broker ihren Weg zurück zum Client. Der Broker kann auch zusätzliche Aufgaben übernehmen, wie z. B. die Bereitstellung zusätzlicher Dienste für die Softwarekomponenten (im Falle des VIDaL-Systems werden die Plattformdienste KT-Register, Gruppenmanagement und andere mehr durch die LPF bereitgestellt).

Das Broker-Muster ermöglicht es den Komponenten, entkoppelt zu bleiben und sich auf ihre eigenen Aufgaben zu konzentrieren, während sie gleichzeitig in der Lage sind, mit anderen Komponenten im System zu kommunizieren und zusammenzuarbeiten. Es verringert die Anzahl der Abhängigkeiten zwischen den Komponenten, wodurch das System flexibler und leichter zu warten ist.

3.1.1.2 Adapter-Muster

Das zweite wichtige Architekturmuster, das bei der Strukturierung des VIDaL-Systems Verwendung findet, ist das *Adapter*-Muster. Bei diesem Muster erfolgt die Kommunikation zwischen zwei Komponenten (KT und LPF) mittels einer Adapter-Komponente (im Falle des VIDaL-Systems das Lagemodul - LM). Der Adapter ermöglicht bidirektionale Kommunikation zwischen Komponenten in einer standardisierten Form und ermöglicht die Komplexitätsreduzierung der angebundenen Schnittstellen.

Der Adapter kann auch zusätzliche Aufgaben übernehmen, wie z. B. Datentransformation oder Sicherheitsaufgaben. Im VIDaL-System übernehmen die Adapter, also die Lagemodule, auch die Funktion von Application Gateways, die das Netz des Brokers (LPF) von den Netzen, in denen die Clients (KT) arbeiten, voneinander trennen.

Im Falle des VIDaL-Systems übernimmt das Lagemodul die Ende-zu-Ende-Verschlüsselung von übermittelten Nachrichten.



Abbildung 3.1: Adapter-Muster

Der Broker und der Adapter kümmern sich somit um die technischen Aspekte der Kommunikation - also die *Vermittlungsebene*, während die fachlichen Systemkomponenten Clients und Server sich auf die eigentliche Anwendungslogik fokussieren - also die *Anwendungsebene*.

3.1.2 Vermittlungsebene

Die zentrale Aufgabe der Vermittlungsebene ist die Zustellung von Meldungen zwischen Sender und Empfänger. Außerdem müssen auf der Vermittlungsebene unterschiedliche querschnittliche Aufgaben durch LM und LPF übernommen werden. Hier sind einige Beispiele: KT-Authentifizierung und Autorisierung, Meldungsvalidierung, E2E-Verschlüsselung. Darüber hinaus stellt die LPF zentrale Dienste für KT und Systemadministratoren bereit, darunter das KT-Register, Gruppenmanagement, Nachrichten-Schema-Dienst und andere.

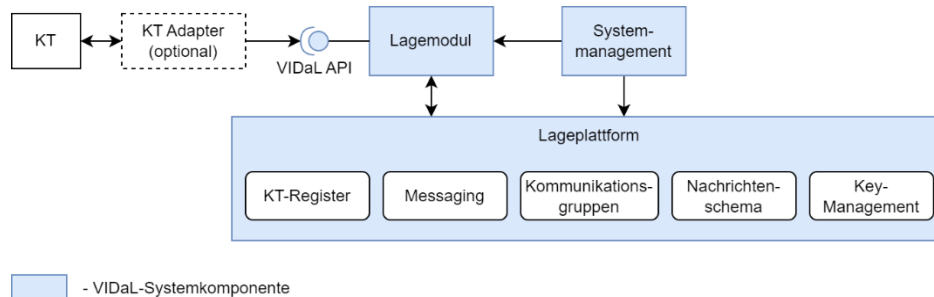


Abbildung 3.2: Überblick Vermittlungsebene

Die Lageplattform (LPF) vernetzt die Lagemodule, die jeweils Informationen bereitstellen und abrufen; interne Prozesse der Stellen des Krisenmanagements bleiben unberührt und sind nicht Gegenstand von VIDaL.

Das Lagemodul (LM) stellt die VIDaL API bereit - die einzige Kommunikationsschnittstelle für angebundene Kommunikationsteilnehmer wie zum Beispiel Lagemanagementsysteme, Leitstellensysteme, andere technische Knoten sowie weitere externe Systeme.

Einzelne Konzepte und Aufgaben der Vermittlungsebene werden an anderen Stellen der technischen Spezifikation beschrieben:

- [Konzept Authentisierung, Autorisierung, Accounting](#)
- [Konzept E2E Verschlüsselung](#)
- [Spezifikation Lageplattform](#)
- [Spezifikation Lagemodul](#)

3.1.3 Anwendungsebene

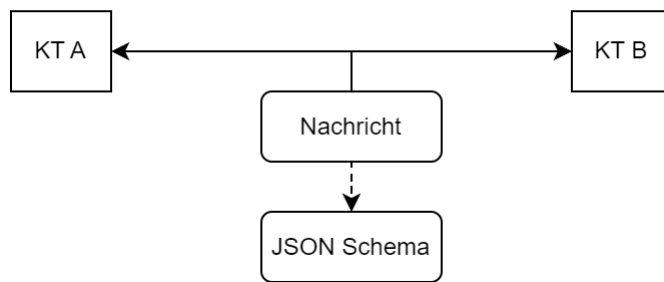
3.1.3.1 VIDaL-Anwendung

Eine VIDaL-Anwendung wird durch folgende Artefakte definiert:

- Ein Satz von standardisierten Nachrichten-Schemata (JSON, kanonisches Datenmodell)
- Ablaufmodell (definierte Abfolge von Nachrichten)
- Prozessdefinitionen (Festlegungen bezüglich Anwendungslogik, die bei der Implementierung in den KT-Systemen berücksichtigt werden müssen)

VIDaL-Anwendungen können zwei Kommunikationsmuster verwenden – eine direkte Kommunikation und eine Gruppenkommunikation:

Direkte Kommunikation



Kommunikationsgruppe

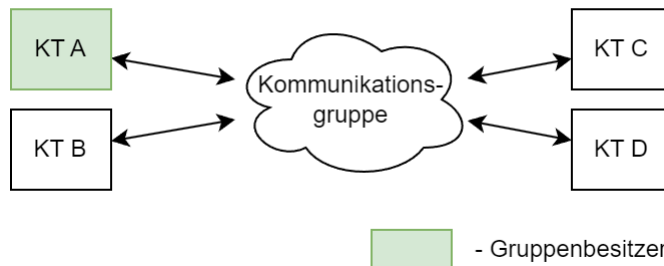


Abbildung 3.3: ViDaL Anwendung

3.1.3.2 Orthogonalität der ViDaL-Anwendungen

Sehr wichtig ist das Prinzip der Orthogonalität der ViDaL-Anwendungen und deren Unabhängigkeit von der Vermittlungsebene (LM, LPF). Anwendungsartefakte sind voneinander streng unabhängig. Das erlaubt eine freie Weiterentwicklung jeder einzelnen Anwendung.

Die LPF selbst ist frei von Fachlichkeit. Sie realisiert nur den Datentransport und sichert die Ende-zu-Ende-Zustellung der Daten. Lediglich im Lagemodul werden die über die ViDaL API entgegengenommenen Daten auf Basis von spezifizierten Datenschemata validiert.

Das System ist Abnehmersystem-agnostisch, lediglich an den Endpunkten jenseits der ViDaL-API kann gegebenenfalls eine Adaption mittels eines Adapterprinzips erforderlich sein. Solche KT-Adapter sind nicht Bestandteil des ViDaL Scopes.

3.1.3.3 Nachrichtenvalidierung

Da die Anwendungsmeldungen mit Hilfe von JSON-Schemata spezifiziert werden, können die versendeten Nachrichten gegen diese Schemata validiert werden. Die Nachrichtenvalidierung findet auf der Vermittlungsebene in den Lagemodulen statt.

Die Plattform stellt einen zentralen Dienst zum Verwalten der Nachrichtenschemata inklusive Versionierung bereit.

Weitere Aspekte der Anwendungsebene werden in der technischen Spezifikation beschrieben: [4 Spezifikation ViDaL-Anwendungen](#).

3.2 Operatives Systemmanagement

Das operative Systemmanagement beschäftigt sich mit den administrativen Aufgaben auf der Applikationsebene.

3.2.1 Administrative Domänen

Es werden 3 administrative Domänen auf der Applikationsebene umgesetzt, um das operative Systemmanagement von folgenden Systemkomponenten zu trennen:

- LPF und LM - es gibt keine Unterschiede in den Rechte- und Rollenkonzepten zwischen den einzelnen LMen. Die LPF ist IAM-Provider für eine gemeinsame Admin-Domäne beider Komponenten, LPF und LM
- LDB hat eine eigene Admin-Domäne
- LDD hat eine eigene Admin-Domäne

Es wird ein Standard-Admin-RBAC-Modell implementiert: User, Admin – auch andere Rollen können im Laufe des Betriebs eingeführt werden.

Das Systemmanagement für die Systemkomponenten LPF, LM erfolgt zentral aus dem Systemmanagement für die LPF heraus.

Das Systemmanagement für die Systemkomponenten LDD und LDB ist jeweils getrennt. Administratoren bekommen jeweils einen separaten Zugang.

3.2.2 Administrative Aufgaben

Auf der Anwendungsebene beschäftigt sich das operative Systemmanagement mit folgenden Aufgaben:

- Modellierung / Umgebungsspezifische Konfiguration von modellierten ViDaL-Systemprozessen (z.B. Keystore oder Truststore-Einstellungen, URLs, etc.)
- Konfigurieren von Laufzeitaspekten der Anwendung wie Clustering, Verteilung, Skalierung, etc.
- Applikationsmonitoring (mit Hilfe von plattform-internen Werkzeugen sowie Open Source-Tools wie Grafana, Prometheus, Loki, etc.)
- Andere administrative Aufgaben wie das Administrieren von angeschlossenen Kommunikationsteilnehmern, der Versand von administrativen Mitteilungen, etc.

3.2.3 Anwendungskonfiguration

Die Bestandteile der zentralen ViDaL-Systemkomponenten werden in Clustern betrieben. Dies inkludiert auch das Lagemodul vor den LDB und LDD. Hierfür werden IM-Bestandteile wie API-Gateway, Worker-Service Prozesse, Datenbank-Prozesse, etc. mehrfach ausgeführt und die gesamten Cluster-Knoten zusätzlich redundant an zwei unterschiedlichen Geostandorten betrieben. Die administrativen Funktionen zum Konfigurieren der beschriebenen Laufzeitaspekte der Anwendung gehören zum Leistungsumfang des IM.

3.3 Architekturentscheidungen

Nachfolgend werden wichtige, teure, kritische oder riskante Architekturentscheidungen, die zentrale Bedeutung für das Gesamtsystem haben, mit Begründungen für diese Entscheidungen beschrieben.

3.3.1 Schnittstelle für Kommunikationsteilnehmer

3.3.1.1 Kontext

Das Lagemodul stellt die ViDaL API bereit. Es ist die einzige Kommunikationsschnittstelle für verbundene Kommunikationsteilnehmer wie zum Beispiel Lagemanagementsysteme, Leitstellensysteme, andere technische Knoten sowie weitere externe Systeme:

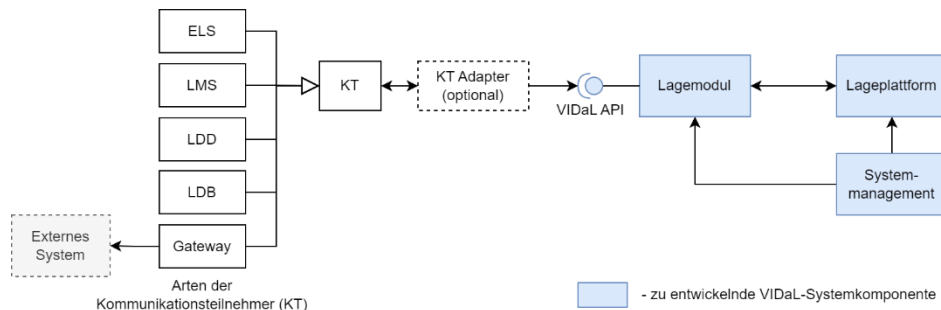


Abbildung 3.4: Schnittstelle für Kommunikationsteilnehmer

3.3.1.2 ViDaL API Technologie

Die Leistungsbeschreibung und die Konzepte aus dem Expertenforum ViDaL, AP3 Anforderungen an die LPF fordern die primäre Nutzung der REST/HTTP-Technologie für die Schnittstellen zwischen den Systemkomponenten. Daraus folgt eine klare Festlegung auf REST/HTTP für die synchrone Übermittlung von Nachrichten und anderen Befehlen/Anfragen an das Lagemodul.

Für das asynchrone Empfangen von Nachrichten kommen folgende Umsetzungsstrategien in Frage:

1. REST Server: KT muss REST API periodisch abfragen (Polling) um Nachrichten zu empfangen
2. REST + Web Socket: KT bekommt Nachrichten asynchron - App Protokoll über asynchrone Kommunikation wird vereinbart
3. REST Client / Server: KT bekommt Nachrichten asynchron über eine bereitgestellte REST API (KT ist auch ein REST Server)

3.3.1.3 Anwendungsprotokoll

Wegen der Anforderung zur sicheren Nachrichtenzustellung (siehe Fachliches Feinkonzept – Nachrichtenvermittlung) wird eine technische Nachrichtenempfangsbestätigung in der ViDaL API vereinbart.

3.3.1.4 Entscheidung

3.3.1.4.1 ViDaL API Technologie

Das Ziel des ViDaL-Systems ist eine Digitalisierung der menschlichen Kommunikation und impliziert (im Gegensatz zu etwa Steuerungsprozessen in M2M-Kommunikation) keine harten Echtzeit-Anforderungen. Aus diesem Grund wird für die Umsetzung der Kommunikationsschnittstelle ViDaL API die technologische Variante 1 (REST API mit Polling) festgelegt. REST API mit Polling von Nachrichten hat folgende Vorteile gegenüber anderen Varianten:

- Einfach zu implementieren
- Einfach zu konsumieren

Um die Auswirkung des Pollings auf die Systemreaktionszeit bei Meldungsaustausch (die maximale Zeit zwischen den Meldungssende- und Meldungsempfangszeitpunkten) zu minimieren, wird bei Meldungsaustausch ein [Long Polling](#) eingesetzt.

3.3.1.4.2 Anwendungsprotokoll

Das Prinzip der sicheren Nachrichtenzustellung ist ein E2E-Prinzip, das auch die Empfangslogik bis zum Persistieren der Nachrichtendaten auf der Seite des KT-Systems einschließt. Um die Ausfälle in dieser Empfangslogik zu kompensieren, wird ein zweistufiges Protokoll für Meldungsempfang vereinbart:

1. Meldungen abfragen - idempotent, kann mehrmals wiederholt werden mit dem gleichen Ergebnis
2. Meldungsempfang bestätigen - idempotent, kann mehrmals wiederholt werden mit dem gleichen Ergebnis. Bestätigte Meldungen werden im ViDaL verworfen und stehen bei erneuter Meldungsabfrage nicht mehr zur Verfügung

3.3.1.5 Auswirkung

Der Client muss die Schnittstelle periodisch abfragen, um Nachrichten zu empfangen. Das klassische Polling-Intervall ist dabei ein Maß zwischen Systemreaktionszeit (die maximale Zeit zwischen den Sende- und Empfangszeitpunkten) und Systemauslastung. Ein Polling-Intervall im Sekundenbereich (3 - 5 Sekunden) ist einzurichten. Um die Auswirkung des Pollings auf die Systemreaktionszeit bei Meldungsaustausch zu minimieren, wird die Verwendung eines [Long Polling](#) vorgesehen.

3.3.1.5.1 Polling bei Nachrichtenabfragen

Folgende Schleife ist bei Nachrichtenabfragen bei [Long Polling](#) zu implementieren:

1. Nachrichten abfragen mit Angabe maximaler Nachrichtenzahl N_{max}
2. Nachrichten verarbeiten
3. Nachrichtenempfang bestätigen
4. Ist die Anzahl von empfangenen Nachrichten gleich N_{max} - sofort zum Schritt 1 übergehen
5. Konfigurierte Pause zwischen 10 Sekunden und 1 Minute einlegen

Wichtig: Bei Programmausfällen zwischen den Schritten 2 und 3 kann es zum wiederholten Empfang von gleichen Nachrichten kommen. Die Client-Logik muss mit Nachrichtenduplikaten umgehen können, z.B. unter Berücksichtigung von eindeutigen Nachrichten-ID.

3.3.1.5.2 Long Polling bei Nachrichtenabfragen

Folgende Schleife ist bei Nachrichtenabfragen bei [Long Polling](#) zu implementieren:

1. Nachrichten abfragen mit Angabe maximaler Nachrichtenzahl N_{max} und maximaler Wartezeit T_{max}
2. Nachrichten verarbeiten
3. Nachrichtenempfang bestätigen

Wichtig: Bei Programmausfällen zwischen den Schritten 2 und 3 kann es zum wiederholten Empfang von gleichen Nachrichten kommen. Die Client-Logik muss mit Nachrichtenduplikaten umgehen können, z.B. unter Berücksichtigung von eindeutigen Nachrichten-ID.

4 Spezifikation ViDaL-Anwendungen

Mit der Lageplattform (LPF) steht eine zentral bereitgestellte hochverfügbare Kommunikationsinfrastruktur mit verteilter Architektur bereit. Mit dezentralen Lagemodulen (LM), die einen sicheren Zugang zur LPF ermöglichen, entsteht so ein ViDaL Kommunikationssystem, das allen Lagebeteiligten - ELS, LMS, Krisenstäbe - ermöglicht, Lageinformationen sicher auszutauschen. Die LPF selbst ist frei von Fachlichkeit. Sie realisiert nur den Datentransport und sichert die Ende-zu-Ende-Zustellung der Daten. Lediglich im Lagemodul werden die über ViDaL API entgegengenommenen Daten auf Basis von spezifizierten Datenschemata validiert.

Auf Basis der LPF werden verschiedene ViDaL-Anwendungen implementiert. Eine ViDaL-Anwendung beschreibt Austausch von definierten Nachrichten zwischen Kommunikationsteilnehmern (KT) bzw. Gruppen von KT.

ViDaL-Anwendungen sind voneinander unabhängig (orthogonal) definiert. D.h. Jede Anwendung, ihre Regeln und die von ihr verwendeten Formate und Schemata sind in sich vollständig und greifen nicht auf Elemente aus anderen Anwendungen zurück.

Damit kann jede Anwendung geändert oder auch gelöscht werden, ohne dass die Nutzung anderer Anwendungen davon direkt betroffen ist.

Eine ViDaL-Anwendung verwendet zum Austausch von Nachrichten die ViDaL-Plattformdienste:

- Messaging - Nachrichtenvermittlungsservice
- KT-Register - ein Service zum Verwalten von den an die Plattform angeschlossenen KT
- Gruppenmanagement - ein Service zum Verwalten von Kommunikationsgruppen
- Schema Service - ein Service zum Verwalten von Datenschemata für den Nachrichtenaustausch

Aktuell sind folgende ViDaL-Anwendungen definiert:

- **Meldungen** - Datenaustausch im Kontext von meldepflichtigen Ereignissen, Großeinsatzlagen und im Katastrophenfall (Lageberichte, Sofort-, Folge- und Schlussmeldungen). Inklusive Archivieren und Abfrage von dokumentationspflichtigen Ereignissen.
- **Ressourcen** - volle Information zur Verfügbarkeit der Ressourcen
- **Einheiten** - Information über Organisationseinheiten, die Prozesse zur Anforderung und Bereitstellung von unterstützenden Verbänden zwischen den Kreisen und Städten strukturieren
- **Kontinuierliche Einsatzstatistik** - landesweit aktuelle Einsatzstatistik

- [Adhoc-Information](#)
- [Organisationen](#) - Verwaltung von Organisationen mit Sicherheitsaufgaben des Landes NRW.

4.1 Schemata – allgemeine Festlegungen

4.1.1 Einleitung

Der Schemata-Entwurf dient der Umsetzung der Anforderungen an [3.3 ViDaL-Anwendungen](#). Dabei folgt die Spezifikation der einzelnen Schemata den entsprechenden fachlichen Vorgaben des ViDaL Expertenforums.

Wo technisch erforderlich, werden Ergänzungen oder Anpassungen vorgenommen, die dem Expertenforum zurückgemeldet werden müssen.

Bei der technischen Gestaltung von JSON-Inhalten (Namenskonventionen, Strukturierung von JSON-Objekten, etc.) berücksichtigt der Entwurf [REST API Design-Richtlinien](#) erarbeitet vom [TM Forum](#) und Standards erarbeitet durch die [OpenAPI Initiative](#).

4.1.2 Namenkonvention

JSON-Attributnamen werden in [camelCase-Notation](#) gehalten. Das ist die am meisten verbreitete Namenskonvention im Bereich JSON-Formatierung. Beispiele: authority, controlCenter.

4.1.3 JSON-Objekte

Fachlich zusammenhängende Attribute werden in JSON-Objekte zusammengefasst. Das erleichtert das Databinding (auch automatisches) in den OO-Programmiersprachen. Beispiel:

Lagebericht

```
{
  "report": {
    "id",
    "timestamp",
    "rootId",
    ...
  },
  "sender": {
    "authority",
    "controlCenter",
    ...
  },
  ...
}
```

Dabei soll ein [JSON-Path](#) für Einzelattribute eines solchen Objektes immer eine fachlich spezifische / eindeutige Beschreibung ergeben wie report.id, report.rootId, sender.controlCenter, etc. Das hilft bei der Vermeidung von Verwirrungen durch zu allgemeine Attributnamen.

4.1.4 Datentypeinschränkungen

4.1.4.1 string

Grundsätzlich werden string-Properties als Text mit dem Unicode-Zeichensatz interpretiert. Das ist eine [Standarddefinition für den JSON-Typ string](#). Die Textlänge kann dabei mit JSON-Schema-Schlüsselworten minLength und maxLength eingeschränkt werden. Beispiel:

string-Schema

```
{
  "type": "string",
  "minLength": 2,
  "maxLength": 3
}
```

Dies ist eine Abweichung zur Einschränkung auf "\w" (a-z, A-Z, 0-9, _) in "VIDaL - AP2 Medien und Strecken v10.pdf". Diese Einschränkung definiert den ASCII-Zeichensatz und schließt unter anderem auch Sonderzeichen der deutschen Sprache aus.

4.1.4.2 integer

Um den ganzzahligen Zahlenbereich eines unsigned int (32 Bit) im JSON-Format abzudecken, ist sicherzustellen, dass der Wert folgende Bedingungen erfüllt:

1. Es muss sich um eine positive ganze Zahl oder "0" handeln.
2. Die Zahl darf nicht größer sein als der maximale Wert, den ein 32-Bit-unsigned int darstellen kann; das sind 4.294.967.295 ($2^{32} - 1$).

Nach der [Standarddefinition für den JSON-Typ integer](#) kann der Wertebereich mit dem JSON-Typ integer mit Schema-Schlüsselworten minimum und maximum eingeschränkt werden. Beispiel:

integer-Schema

```
{
  "type": "integer",
  "minimum": 0,
  "maximum": 4294967295
}
```

4.1.5 Zusammenhängende Meldungen (Request/Response)

Um zusammenhängende Meldungen (typischerweise Anfragen und entsprechende Antworten) als solche zu erkennen und zu behandeln, wird auf der Client-Seite eine Request-ID generiert und mit der Anfrage-Nachricht übertragen. Eine Antwort-Nachricht bekommt auf der Server-Seite dieselbe Request-ID. Dadurch kann der Client jeden erhaltenen Response einem zuvor gesendeten Request zuordnen. Das bringt folgende Vorteile:

- Antwort-Nachrichten (Responses) müssen nicht alle Informationen beinhalten, damit der Client sie verstehen und zuordnen kann. Der vollständige Kontext wird durch Zuordnung eines Responses zu dem entsprechenden Request und darin enthaltene Anfrage-Informationen hergestellt. Dadurch wird die zu transferierende Datenmenge minimiert.
- Anfragen und Antworten können asynchron versendet und empfangen werden.

4.1.6 Fehlermeldungen

Um Fehlersituationen zu signalisieren, wird eine anwendungsspezifische Fehlermeldung verwendet. Alle anwendungsspezifischen Fehlermeldungen beinhalten allgemeine Attribute, die in [4.1.8 Schema - Fehlermeldung - v1.0](#) beschrieben sind. Dabei verwendet jede Anwendung anwendungsspezifische Fehler-Codes. Diese Fehler-Codes werden in den einzelnen Anwendungen in den abgeleiteten Schemata definiert.

4.1.7 Validierung von Meldungen

Einer ViDaL-Anwendung ist ein Satz von JSON-Schemata für die anwendungsspezifischen Nachrichten zugeordnet. Damit eine Nachricht einem Schema zugeordnet (und folglich automatisch validiert) werden kann, beinhaltet jede Nachricht den Nachrichtentyp (Schema-Namen) und die Schema-Version:

- `schemald` - Schemabezeichner - dient zur eindeutigen Identifizierung eines Schema-Typs im ViDaL-System
- `schemaVersion` - Version eines Schemas. Schemata werden weiterentwickelt. Eine Schema-ID zusammen mit der Schema-Version bezeichnen einen definierten Stand dieser Entwicklung.

Die JSON-Schema-Validierungslogik basiert auf der Prädikatenlogik und hat somit begrenzte Möglichkeiten im Bereich der bedingten Validierung. Es ist unter anderem nicht möglich, eine Validierungslogik und eine Bedingung zu ihrer Anwendung auf unterschiedlichen Strukturebenen in einem JSON-Schema zu platzieren.

Im Kontext der Validierung von ViDaL-Meldungen betrifft diese Einschränkung die Definition von Pflichtfeldern.

Für das folgende Beispiel-JSON-Dokument kann z.B. mit Hilfe eines JSON-Schemas definiert werden, ob das Attribut „vehicle“ ein Pflichtfeld ist oder nicht, und zwar abhängig vom Wert des Attributes „schemaUseCase“ (beide Attribute sind auf der gleichen Strukturebene). Die gleiche abhängige Definition ist dagegen in Bezug auf das Attribut „vehicle/chassisNumber“ nicht möglich (Attribute sind auf unterschiedlichen Strukturebenen):

```
{
  „schemaUseCase“: „create“,
  „vehicle“:
  {
    „chassisNumber“: „WVWZZZAUZ...“,
    „chassisSupplier“: „VW“
  }
}
```

Um diese Validierungsproblematik zu umgehen, werden bei der Schema-Definition folgende Regeln festgelegt:

1. Validierungsregel: In einem VIDaL-Schema enthaltene Objekte bilden Validierungskontexte. Innerhalb eines solchen Objektes wird eine eigene Validierungslogik definiert, die von Bedingungen aus den übergeordneten Ebenen nicht abhängen kann.
2. Update-Regel: Wird ein Objekt aus einer VIDaL-Meldung zum Aktualisieren von Dateninhalten verwendet, so wird das entsprechende Element in der Zieldatenstruktur auf der Empfängerseite mit dem Objekthinhalte aus der Meldung komplett überschrieben (das alte Objekt mit allen seinen Attributen gelöscht). In anderen Worten muss das gesendete Objekt alle Attribute beinhalten, die die Zieldatenstruktur beim Empfänger nach der Verarbeitung der Update-Meldung auch beinhalten soll. Diese Update-Regel gilt auch für Arrays - Arrays müssen vollständig übertragen werden, so wie sie in der Zieldatenstruktur beinhaltet werden sollen.

Die Anwendung der Schema-Definition-Regeln soll an folgenden Beispielen erläutert werden.

Angenommen, die Schema-Definition für das Beispiel von oben sieht wie folgt aus:

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflichtfeld		Beschreibung	Beispiel
			Anlage	Änderung		
/schemaUse-Case	enum[string]	[create, update]	Ja	Ja	Verwendung: create - Anlegen einer Ressource, update - Aktualisieren einer Ressource	create
/resource	object		Nein	Nein		
./id	string		Ja		Ressourcen-ID	123456
./type	enum[integer]	[1, 2, 3]	Ja		Typ der Ressource	1
/vehicle	object		Nein	Nein		
./chassisNumber	string		Ja		Fahrgestellnummer	WVWZZZAUZ...
./chassisSupplier	string		Nein		Hersteller des Fz-Fahrgestells	VW

Tabelle 4.1: Schema-Definition für das Beispiel

Die folgende Tabelle zeigt die Auswirkung der Schema-Definition-Regeln bei der angegebenen Reihenfolge der Meldungen:

Meldung in der Reihenfolge	Validierung	Zieldatenstruktur nach der Meldungsverarbeitung	Kommentar
{ „schemaUseCase“: „create“, „vehicle“: { „chassisNumber“: “WVWZZZAUZ...“, „chassisSupplier“: „VW“ }}	OK	{ „vehicle“: { „chassisNumber“: “WVWZZZAUZ...“, „chassisSupplier“: „VW“ } }	Neuanlage der Zieldatenstruktur
{ „schemaUseCase“: „update“, „resource“: { „id“: “1234“, „type“: 1 } }	OK	{ „resource“: { „id“: “1234“, „type“: 1 }, „vehicle“: { „chassisNumber“: “WVWZZZAUZ...“, „chassisSupplier“: „VW“ } }	Aktualisierung der Zieldatenstruktur: neues Objekt kommt dazu
{ „schemaUseCase“: „update“, „vehicle“: { „chassisSupplier“: „Volkswagen“ }}	Fehler!	{ „resource“: { „id“: “1234“, „type“: 1 }, „vehicle“: { „chassisNumber“: “WVWZZZAUZ...“, „chassisSupplier“: „VW“ }}	Fehlendes Pflichtfeld „vehicle.chassisNumber“. Die alte Zieldatenstruktur bleibt unverändert.
{ „schemaUseCase“: „update“, „vehicle“: { „chassisNumber“: „WVWZZZAUZ...2“ }}	OK	{ „resource“: { „id“: “1234“, „type“: 1 }, „vehicle“: { „chassisNumber“: “WVWZZZAUZ...2“, } }	Aktualisierung der Zieldatenstruktur: das Objekt „vehicle“ wurde ersetzt – dabei geht das optionale Feld „vehicle.chassisSupplier“ „verloren“.
{ „schemaUseCase“: „update“, „vehicle“: { „chassisNumber“: „WVWZZZAUZ...3“, „chassisSupplier“: „Volkswagen“ }}	OK	{ „resource“: { „id“: “1234“, „type“: 1 }, „vehicle“: { „chassisNumber“: “WVWZZZAUZ...3“, „chassisSupplier“: „Volkswagen“ }}	Aktualisierung der Zieldatenstruktur: das Objekt „vehicle“ wurde ersetzt – das neue Objekt bekommt auch das optionale Feld „vehicle.chassisSupplier“.
{ „schemaUseCase“: „update“, „vehicle“: null}	OK	{ „resource“: { „id“: “1234“, „type“: 1 } }	Aktualisierung der Zieldatenstruktur: das Objekt „vehicle“ wurde gelöscht.

Tabelle 4.2: Beispiel Auswirkung der Schema-Definition-Regeln

4.1.8 Schema -Fehlermeldung – v1.0

4.1.8.1 Schema-Bezeichner

Wird jeweils passend zur jeweiligen Anwendung festgelegt.

4.1.8.2 Beschreibung

Nachfolgend das folgende Schema dient zum Signalisieren von Fehlersituationen.

Die hier festgehaltenen Inhalte finden sich in allen Schemata für anwendungsbezogene Fehlermeldungen wieder.

4.1.8.3 Schema

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/errorCode	integer	minimum = 0	Ja	Anwendungsspezifischer Fehler-Code, sollte an HTTP-Statuscodes angelehnt sein - 4xx - Client-Fehler, 5xx - Server-Fehler
/extendedErrorCode	integer	minimum = 0	Nein	Erweiterter anwendungsspezifischer Fehler-Code
/errorReason	string	maxLength: 100	Ja	Lesbare Fehlerursache - kurze Zusammenfassung des Fehlers
/errorText	string	maxLength: 1000	Nein	Lesbarer Fehlerbeschreibung - detaillierte Beschreibung des Fehlers

Tabelle 4.3: Schema Fehlermeldung

4.1.8.4 Beispiel

```
{
  "errorCode": 401
  "errorReason": "Der User ist nicht autorisiert"
}
```

4.2 Spezifikation Meldungen

Hier werden die Payload-Schemata für [3.3.3 Meldungen](#) dokumentiert.

Der Initialstand wurde anhand von "VIDaL - AP2 Medien und Strecken v10.pdf" (für Bezeichner und Pflichtfelder), "VIDaL AP1 Anl1 Inhalte Melde- und Krisenstabserlasse v11 (in Arbeit).xlsx" und "VIDaL Inbetriebnahme und Pilotbetrieb v02.pptx" erstellt. Die vorliegenden Schemata beinhalten Ergänzungen, die den aktuellen fachlichen Stand in NRW abbilden.

In Anlehnung an [REST-Prinzipien](#) definieren die Schemata folgende Repräsentationen zur Veränderung des Anwendungszustandes durch einen Meldungsaustausch:

- Lageinformation - repräsentiert eine Meldung zu einer Lage
- Lageinformations-Abfrage - repräsentiert eine Abfrage zu einer konkreten Lage, z.B. bei Late-Entry. Ergebnis der Abfrage ist eine Reihe nacheinander folgenden Lageinformation-Meldungen

Darüber hinaus wird ein Schema zur Darstellung eines Fehlers bei der Veränderung des Anwendungszustandes definiert:

- Lageinformationsfehler

4.2.1 Schema – Lageinformation – v1.0

4.2.1.1 Schema-Bezeichner

situationReport

4.2.1.2 Schema

Block AP1	JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
	/situationRequestId	string[uuid]	UUID	Nein	ID des entsprechenden Requests, wenn Lageinformation nachträglich als Antwort auf eine Late-Entry-Abfrage gesendet wird.
	/report	object		Ja	

Block AP1	JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
M11	./id	string	pattern: ^\\d{8}\\d{10}\$	Ja	ID der vorliegenden Meldung, enthält den Sender als AGS (8 Stellen) und Datum (yyymmddhhmm)
M12	./rootId	string	pattern: ^\\d{8}\\d{10}\$	Ja	ID des Meldungsverlaufs = Ursprungsmeldung/ Lage-ID, enthält den Sender als AGS (8 Stellen) und Datum (yyymmddhhmm)
M13	./sequence	integer	minimum: 0 maximum: 4294967295	Ja	<p>Laufende Nummer der Meldung (beginnend mit 0 bei Sofortmeldung, alle darauf folgenden Meldungen beginnen mit 1 (z.B. auch von anderen Sendern)).</p> <p>Die laufende Nummer der Meldung wird durch jeden KT in der Sender-Rolle als eine Sequenz gepflegt. In einer gegebenen Lage (M12) ist die Kombination aus der KT-AGS (Teil der M11) und der laufenden Nummer der Meldung (M13) eindeutig, nicht die laufende Nummer der Meldung (M13) selbst.</p>
M14	./type	array[integer]	pattern: ^1 2 3 4 9\$ (nur für erstes Element)	Ja (mind. 1, max. 2 Elemente)	<p>Art der Meldung + Version definierte Auswahl, Bsp. NW:</p> <p>1: Sofortmeldung, Erstmeldung 2: Folgemeldung, Lagemeldung 3: Lagebericht 4: Nachricht 9: Schlussmeldung</p> <p>Liste, bis zu zwei Auswahlen gleichzeitig möglich</p>
M15	./refRootId	array[string]	pattern: ^\\d{8}\\d{10}\$	Ja	Liste von ID's referenzierter Meldungsverläufe (n * (Sender als AGS (8 Stellen) und Datum (yyymmddhhmm))), bei n=0: Wert "null" angeben
M16	./approvalOfficer	string	maxLength: 200	Nein	Freigabe der Meldung (Name der gesamtverantwortlichen Person)
M17	./approvalTime	string	format: date-time ¹	Nein	Freigabe der Meldung (Datum, Zeit), siehe "Regex DT"

Block AP1	JSON-Property	JSON-Daten- typ	Datentyp-Einschrän- kung	Pflicht	Beschreibung
	/sender	object		Ja	Angaben zum Sender
M211	./authority	string	maxLength: 200	Ja	Behörde (z.B. Übernahme automatisch aus Registry)
M212	./controlCenter	string	maxLength: 200	Nein	Leitstelle
M213	./operationsControl	string	maxLength: 200	Nein	Einsatzleitung (zuständige Orga-Einheit)
M214	./officer	string	maxLength: 200	Ja	Verantwortlicher Bearbeiter
M215	./mail	string	format: email ¹ maxLength: 200	Nein	E-Mail, siehe "Regex Email"
M216	./telephone	string	maxLength: 100	Nein	Telefon, siehe "Regex Telefon"
M22	./activateCrisisManage- ment	boolean	true=Ja	Ja	Aktivierung des Krisenstabes
	/receiver	object		Ja	Angaben zum Empfänger
M23	./main	string	maxLength: 200	Ja	Empfänger 1 (z.B. Landes-KM) (Behörde, wie in "senderAuthority")
M24	./other	array[string]	maxLength: 200 (0..n)	Nein	Weitere Empfänger (z.B. BezReg-KM) (Behörde, wie in "senderAuthority")
M25	./cc	array[string]	maxLength: 200 (0..n)	Nein	Empfänger CC-Liste (Behörde, wie in "senderAuthority")
	/event	object		Nein	Angaben zum Schadensereignis
M31	./type	string	maxLength: 100 Codeliste	Ja	Codeliste Einsatzstichwortkatalog: Einsatzstichwortliste auf Basis Hessen, siehe „VIDaL AP1 Anl72 Einsatzstichworte & Inhalte Kontinuierliche Einsatzstatistik“.
M32	./locationAGS	string	pattern: ^\d{8}\\$	Ja	Schadensort / Ereignisort (nur AGS)

Block AP1	JSON-Property	JSON-Daten- typ	Datentyp-Einschrän- kung	Pflicht	Beschreibung
M33	./address	string	maxLength: 200	Nein	Anschrift des Schadensorts / Ereignisorts z.B. Adressangaben aus ELS, ggf. Hinweis auf Flächenschaden
M34	./coordinate	string	pattern: \d{1,8}\.\d{1,7}\.\d{4,5}	Nein	Koordinaten des Schadensorts / Ereignisorts Georeferenz/Georeferenzen (Polygon), siehe "Regex GeoLoc"
M35	./object	string	maxLength: 200	Nein	Schadensobjekt / Ereignisobjekt Freitext (Lagerhalle, BAB, Scheune, Flugzeugabsturz)
M36	./timestamp	string	format: date-time ¹	Ja	Schadenszeitpunkt / Ereigniszeitpunkt Datum, Zeit
M37	./reportTimestamp	string	format: date-time ¹	Ja	Meldezeitpunkt des Schadens / Ereignisses Datum, Zeit
M371	./reportTimespan	string	format: period ¹	Ja	Berichtszeitraum (Start und Ende)
M38	./weather	string	maxLength: 2000	Nein	Wetterdaten (Wind-/Zugrichtung)
	/event/detail	object		Nein	Nähere Angaben zu Details des Schadensereignisses
M41	./incident	string	maxLength: 2000	Nein	Nähere Angaben zum Ereignis
M42	./damage	string	maxLength: 2000	Nein	Nähere Angaben zu Schäden
M49	./othersInvolved	string	maxLength: 2000	Nein	sonstige Informationen zu Betroffenen
	/event/hotspots/	array[object]	Anzahl: n	Nein	Liste der Schadensschwerpunkte
M43	./hotSpotAGS	string	pattern: ^\d{8}\$	Ja	AGS eines Schadensschwerpunktes (bei n=1 ist die AGS = M32)
M430	./hotSpotDescription	string	maxLength: 10000	Nein	Beschreibung des Schadensschwerpunkt mit Schäden, Besonderheiten, ...

Block AP1	JSON-Property	JSON-Daten- typ	Datentyp-Einschrän- kung	Pflicht	Beschreibung
M431	./hotSpotActive	boolean	true=aktiv	Nein	Lage aktiv/inaktiv am Schadensschwerpunkt (heiß/kalt)
M432	./missionsPerHotSpot	integer	minimum: 0 maximum: 99999999	Nein	Anzahl aktive Einsätze pro Schadensschwerpunkt (z.B. Hilfeleistungen, Auspumpen Keller, ...)
M441	./publicInjured1	integer	minimum: 0 maximum: 99999999	Nein	Bevölkerung Sichtungskategorie 1 / T1 (rot)
M442	./publicInjured2	integer	minimum: 0 maximum: 99999999	Nein	Bevölkerung Sichtungskategorie 2 / T2 (gelb)
M443	./publicInjured3	integer	minimum: 0 maximum: 99999999	Nein	Bevölkerung Sichtungskategorie 3 / T3 (grün)
M444	./publicInjured4	integer	minimum: 0 maximum: 99999999	Nein	Bevölkerung Sichtungskategorie 4 / T4 (schwarz = tot)
M445	./personsInvolved	integer	minimum: 0 maximum: 99999999	Nein	Bevölkerung betroffene / zu betreuende Personen
M446	./personsMissing	integer	minimum: 0 maximum: 99999999	Nein	Vermisste Personen
M451	./firebrigadeInjured1	integer	minimum: 0 maximum: 99999999	Nein	Feuerwehr Sichtungskategorie 1 / T1 (rot)
M452	./firebrigadeInjured2	integer	minimum: 0 maximum: 99999999	Nein	Feuerwehr Sichtungskategorie 2 / T2 (gelb)
M453	./firebrigadeInjured3	integer	minimum: 0	Nein	Feuerwehr Sichtungskategorie 3 / T3 (grün)

Block AP1	JSON-Property	JSON-Daten- typ	Datentyp-Einschrän- kung	Pflicht	Beschreibung
			maximum: 99999999		
M454	./firebrigadeInjured4	integer	minimum: 0 maximum: 99999999	Nein	Feuerwehr Sichtungskategorie 4 / T4 (schwarz = tot)
M461	./hiOrgInjured1	integer	minimum: 0 maximum: 99999999	Nein	HiOrg Sichtungskategorie 1 / T1 (rot)
M462	./hiOrgInjured2	integer	minimum: 0 maximum: 99999999	Nein	HiOrg Sichtungskategorie 2 / T2 (gelb)
M463	./hiOrgInjured3	integer	minimum: 0 maximum: 99999999	Nein	HiOrg Sichtungskategorie 3 / T3 (grün)
M464	./hiOrgInjured4	integer	minimum: 0 maximum: 99999999	Nein	HiOrg Sichtungskategorie 4 / T4 (schwarz = tot)
M471	./thwInjured1	integer	minimum: 0 maximum: 99999999	Nein	THW Sichtungskategorie 1 / T1 (rot)
M472	./thwInjured2	integer	minimum: 0 maximum: 99999999	Nein	THW Sichtungskategorie 2 / T2 (gelb)
M473	./thwInjured3	integer	minimum: 0 maximum: 99999999	Nein	THW Sichtungskategorie 3 / T3 (grün)
M474	./thwInjured4	integer	minimum: 0 maximum: 99999999	Nein	THW Sichtungskategorie 4 / T4 (schwarz = tot)
M481	./otherOrgInjured1	integer	minimum: 0	Nein	Sonstige Sichtungskategorie 1 / T1 (rot)

Block AP1	JSON-Property	JSON-Daten- typ	Datentyp-Einschrän- kung	Pflicht	Beschreibung
			maximum: 99999999		
M482	./otherOrgInjured2	integer	minimum: 0 maximum: 99999999	Nein	Sonstige Sichtungskategorie 2 / T2 (gelb)
M483	./otherOrgInjured3	integer	minimum: 0 maximum: 99999999	Nein	Sonstige Sichtungskategorie 3 / T3 (grün)
M484	./otherOrgInjured4	integer	minimum: 0 maximum: 99999999	Nein	Sonstige Sichtungskategorie 4 / T4 (schwarz = tot)
	/event/hotspots/resources	object		Nein	Eingesetzte Ressourcen am Schadensschwerpunkt
M721	./vehiclesAlerted	integer	minimum: 0 maximum: 99999999	Nein	Anzahl der Einsatzmittel (Fahrzeuge) - alarmiert
M723	./vehicles3	integer	minimum: 0 maximum: 99999999	Nein	Anzahl der Einsatzmittel (Fahrzeuge) - Status 3
M724	./vehicles4	integer	minimum: 0 maximum: 99999999	Nein	Anzahl der Einsatzmittel (Fahrzeuge) - Status 4
M727	./vehicles7	integer	minimum: 0 maximum: 99999999	Nein	Anzahl der Einsatzmittel (Fahrzeuge) - Status 7
M728	./vehicles8	integer	minimum: 0 maximum: 99999999	Nein	Anzahl der Einsatzmittel (Fahrzeuge) - Status 8
M731	./manpowerAlerted	integer	minimum: 0 maximum: 99999999	Nein	Anzahl der Einsatzkräfte - alarmiert

Block AP1	JSON-Property	JSON-Daten- typ	Datentyp-Einschrän- kung	Pflicht	Beschreibung
M732	./manpower3	integer	minimum: 0 maximum: 9999999	Nein	Anzahl der Einsatzkräfte - im Einsatz Feuerwehr
M733	./manpower4	integer	minimum: 0 maximum: 9999999	Nein	Anzahl der Einsatzkräfte - im Einsatz HiOrg
M734	./manpower7	integer	minimum: 0 maximum: 9999999	Nein	Anzahl der Einsatzkräfte - im Einsatz THW
M735	./manpower8	integer	minimum: 0 maximum: 9999999	Nein	Anzahl der Einsatzkräfte - im Einsatz Sonstige
	/resources	object		Nein	Eingesetzte Ressourcen (allgemeine Informationen)
M71	./operationalLead	string	maxLength: 200	Nein	Einsatzführung (Mit der Führung beauftragt Person und Funktion)
M74	./nrwEinheit	array[string]	pattern: ^\w{10}\$	Nein	eingesetzte Einheiten (= Fähigkeiten), nach NRW-Konzept Liste der LKE-IDs
M75	./other	string	maxLength: 2000	Nein	sonstige Informationen zu Ressourcen
	/additionalSituationInfo	object		Nein	Weitere Lageinformation
M51	./generalSituation	string	maxLength: 2000	Nein	allgemeine Lage
M52	./currentState	string	maxLength: 2000	Nein	Sachverhalt / Ermittlungsstand
M53	./publicSafety	string	maxLength: 2000	Nein	öffentliche Sicherheit und Ordnung
M54	./traffic	string	maxLength: 2000	Nein	Verkehr
M55	./publicHealth	string	maxLength: 2000	Nein	Gesundheitswesen
M56	./environment	string	maxLength: 2000	Nein	Umwelt

Block AP1	JSON-Property	JSON-Daten- typ	Datentyp-Einschrän- kung	Pflicht	Beschreibung
M57	./publicCare	string	maxLength: 2000	Nein	Versorgung der Bevölkerung
M58	./ICT	string	maxLength: 2000	Nein	Informations- und Kommunikationswesen
M59	./hazards	string	maxLength: 2000	Nein	Gefahren
M510	./prediction	string	maxLength: 2000	Nein	Lageentwicklung, Prognose
M511	./detailsHazard	string	maxLength: 2000	Nein	Schadenslage/Gefahrenlage
M512	./detailsAdministration	string	maxLength: 2000	Nein	Allgemeine Lage/Verwaltungslage
	/action	object		Nein	Maßnahmen
M61	./tacticalGoal	string	maxLength: 2000	Nein	(eigenes) Taktisches Ziel (ggf. mehrere)
M62	./informOthers	string	maxLength: 2000	Nein	Hinweis an betroffene/n Nachbarn
M631	./judicialMeasures	string	maxLength: 2000	Nein	straßprozessual
M632	./administrative	string	maxLength: 2000	Nein	administrativ-organisatorisch
M633	./operative	string	maxLength: 2000	Nein	operativ-taktisch
M634	./initiated	string	maxLength: 2000	Nein	eingeleitete Maßnahmen
M635	./planned	string	maxLength: 2000	Nein	beabsichtigte Maßnahmen
	/publicAnnouncement	object		Nein	Warnung der Bevölkerung
M81	./warning	boolean	true=Ja	Ja	Warnung erfolgt?

Block AP1	JSON-Property	JSON-Daten- typ	Datentyp-Einschrän- kung	Pflicht	Beschreibung
M82	./warningMedia	array[integer]	Embedded Codeliste: 1-10, 99	Nein	Warnmittel, definierte Mehrfachauswahl: 1: MoWaS regionale Medien 2: MoWaS überregionale Medien 3: MoWaS Warn-Apps 4: Hörfunksender ohne MoWaS-Ansteuerung 5: Durchsage im Lokalhörfunksender "Radio on air" 6: Sirenen 7: Warndurchsagen mittels Lautsprecher durch Warnfahrzeuge 8: Verbreitung in Sozialen Medien 9: Fahrgast- / Stadtinformationssysteme 10: Cell Broadcast 99: sonstige
M83	./additionalInfoWarning- Media	string	maxLength: 2000	Nein	Zusatzinformation zum Warnmitteln
M84	./warningText	string	maxLength: 2000	Nein	Warntext
M89	./clear	boolean	true=Ja	Ja	Entwarnung erfolgt?
	./additionalInfo	object		Nein	Zusatzinformationen
M91	./pressRelease	string	maxLength: 2000	Nein	Presse/Medienlage
M92	./specialOccurance	string	maxLength: 2000	Nein	besondere Vorkommnisse
M93	./attachments	string	maxLength: 2000	Nein	Sonstigen Informationen
M94	./nextReportScheduled	string	date-time ¹	Nein	Zeitpunkt des nächsten Lageberichts

Tabelle 4.4: Schema Lageinformation

4.2.1.3 Beispiel (gekürzt)

```
{
  "report": {
    "id": "12345678.2308171023",
    "rootId": "12345678.2308171023",
    "sequence": "123",
    "type": [
      "1",
      "0"
    ],
    "refRootId": [
      "12345678.2308171023",
      "12345678.2308171023",
      "12345678.2308171023"
    ],
    "approvalOfficer": "Peter Lehmann",
    "approvalTime": ""
  },
  "sender": {
    "authority": "Innenministerium NRW",
    "officer": "Manfred Müller",
    "activateCrisisManagement": true
  },
  "receiver": {
    "main": ""
  },
  "event": {
    "type": "Brandkatastrophe",
```

```

"detail": {
  "incident": "Schadensvorfall: ....."
},
"hotspots": [
  {
    "hotSpotAGS": "12345670",
    "resources": {
      "vehiclesAlerted": "23"
    }
  },
  {
    "hotSpotAGS": "12345671",
    "resources": {
      "vehiclesAlerted": "12",
      "manpowerAlerted": "45"
    }
  },
  {
    "hotSpotAGS": "12345672",
    "resources": {
      "vehicles3": "4",
      "vehicles4": "2"
    }
  }
]
},
"resources": {
  "other": "Sonstige Informationen: ..."
}

```

```

    },
    "additionalSituationInfo": {
        "generalSituation": "Allgemeine Situation: ....."
    },
    "action": {
        "tacticalGoal": "Einsatzziel: ....."
    },
    "publicAnnouncement": {
        "warning": true,
        "clear": false
    },
    "additionalInfo": {
        "pressRelease": "Presstext: ....."
    }
}

```

¹ <https://json-schema.org/understanding-json-schema/reference/string.html> → Built-in formats

4.2.2 Schema – Lageinformations-Abfrage -v1.0

4.2.2.1 Schema-Bezeichner

situationRequest

4.2.2.2 Schema

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/situationRequestId	string[uuid]	UUID	Ja	Request-ID
/reportRootId	string	pattern: ^\d{8}\.\d{10}\$	Ja	ID des Meldungsverlaufs = Ursprungsmeldung/ Lage-ID, enthält den Sender als AGS (8 Stellen) und Datum (yymmddhhmm)

Tabelle 4.5: Schema Lageinformations-Abfrage

4.2.2.3 Beispiel

Late-Entry-Abfrage
<pre>{ "situationRequestId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454", "reportRootId": "12345678.2308171023" }</pre>

4.2.3 Schema – Lageinformationsfehler – v1.0

4.2.3.1 Schema-Bezeichner

situationReportError

4.2.3.2 Schema

Das Schema ist identisch mit dem Schema - Fehlermeldung - v1.0. Dazu kommen folgende Attribute:

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/situationRequestId	string[uuid]	UUID	Ja	Request-ID

Tabelle 4.6: Schema Lageinformationsfehler

4.2.3.3 Fehler-Codes

VIDaL-Anwendung Meldungen definiert folgende Fehlermeldungen:

Fehler-Code (errorCode)	Fehlerursache (errorReason)	Anwendungsfall	Beschreibung
409	Der Meldeverlauf mit der Meldeverlauf-ID <id> bereits abgeschlossen	Meldungen abfragen (Late Entry)	Der abgefragte Meldeverlauf ist bereits abgeschlossen/archiviert
404	Zu dem Meldeverlauf mit der Meldeverlauf-ID <id> wurden keine Meldungen gefunden	Meldungen abfragen (Late Entry)	Zu der abgefragten Meldeverlauf-ID wurden keine Meldungen im LDD gespeichert
401	Der abfragende KT ist kein Teilnehmer des Meldeverlaufs mit der Meldeverlauf-ID <id>	Meldungen abfragen (Late Entry)	Der abfragende KT ist kein Teilnehmer des Meldeverlaufs
503	Storage nicht aktuell	Meldungen abfragen (Late Entry)	Durch Kommunikationsunterbrechung zu beiden Systemen ist das Storage System nicht aktuell. Das ist ein temporärer Fehler. Der KT soll die Late Entry Anfrage nach einer kurzen Wartezeit ca. 5 Minuten wiederholen.

Tabelle 4.7: Fehler-Codes Lageinformationsfehler

4.3 Spezifikation Einheiten

Hier werden die Payload-Schemata für 3.3.4 Einheiten dokumentiert.

Der Initialstand wurde anhand von "VIDaL - AP2 Medien und Strecken v10.pdf" und "VIDaL AP1 Anl6 Datenstruktur Konzepte+Einheiten v10.pdf" erstellt.

In Anlehnung an die REST-Prinzipien definieren die Schemata folgende Repräsentationen zur Veränderung des Anwendungszustandes durch einen Meldungsaustausch:

- Einheit - repräsentiert das Landeskonzept Einheit (LKE) mit seinen Eigenschaften
- Einheitenübersicht - repräsentiert eine Liste von mehreren Einheiten des LKE
- Einheit-Referenz - repräsentiert eine Referenz auf eine konkrete Einheit
- Einheitenfilter - repräsentiert einen Filter mit mehreren Filterkriterien. Ergebnis der Anwendung eines Filters ist die Einheitenübersicht

Darüber hinaus wird ein Schema zur Darstellung eines Fehlers bei der Veränderung des Anwendungszustandes definiert:

- Einheitenfehler

Im Gegensatz zum REST-Paradigma werden die gewünschten Operationen direkt als Teil einer Repräsentation übertragen - Attribut `schemaUseCase`.

4.3.1 Schema – Einheit – v1.0

4.3.1.1 Schema-Bezeichner

`unitLke`

4.3.1.2 Schema

JSON-Pro- property	JSON-Daten- typ	Datentyp- Einschränkung	Pflichtfeld für				Beschreibung	Beispiel
			Anlage	Ände- rung	Über- sicht	Detail- sicht		
/sche- maUse- Case	enum[st- ring]	[create, update, short- View, fullView]	Ja	Ja	Ja	Ja	Verwendung: create - Anlegen einer LKE, update - Aktu- alisieren einer LKE, shortView - LKE-Sicht mit wichtigs- ten Angaben (Übersicht), fullView - komplette LKE- Sicht (Detailsicht)	create
/unitRe- questId	string[uu- id]	UUID	Ja	Ja	Nein	Ja	ID des entsprechenden Requests. Wenn LKE-Übersicht als Element der LKE-Liste gelie- fert wird, dann wird entsprechende unitRequestId auf der Listenebene übertragen und nicht in jedem Lis- tenelement.	f8c3de3d-1fea- 4d7c-a8b0- 29f63c4c3454
/re- portTi- meLKE	string	format: date-time	Ja	Ja	Ja	Ja	Zeitpunkt der letzten Änderung, z.B. des VIDaL-Status	
/unit- LKE			Ja	Ja	Ja	Ja		
./id	string	pattern: ^E\d{8}\.\d{10}\$	Ja				LKEID (Owner.LfdNr), Format: E+AGS + lfd.Nr. (10 Stellen) ; lfd.Nr. wurde von 4 auf 10 Stellen erweitert / Festlegung am 11.06. im Anwender-Workshop	E05100000.000100 0000
./name	string	maxLength: 100	Ja				Einheiten-Bezeichnung	EE NRW VIE 2 (DRK, DRK)
./ope- rator	string	pattern: ^\d{8}\$	Ja				Durchführende Stelle - AGS	05124000

JSON-Pro- perty	JSON-Daten- typ	Datentyp- Einschränkung	Pflichtfeld für				Beschreibung	Beispiel
			Anlage	Ände- rung	Über- sicht	Detail- sicht		
./type	string	pattern: ^\d{1,2}\.\d{1,2}\.\d{1,3}\$ (Code-Liste)	Ja				Typ der Einheit (aus Schlüsselliste LKEArt)	05.02.001
./vidal- Status	string	pattern: ^2 2A 3 3A 4 6 7 8\$ (Code-Liste)	Ja				VIDaL-Status (verfügbar = 2)	2
./coordi- nate	string	pattern: ^\s*E:\s*\d{1,8}\.\d{2}\s*; \s*N:\s*\d{1,7}\.\d{2}\s*\$	Ja				GPS-Position des Führungsfahrzeuges des Verbands, Format: Geo-Position (ETRS89) Geo-Position-Angabe, falls vidalStatus = 3, 3A, 4. Sonst null-Wert.	"E: 32455098,11; N: 5428178,66", null
./poin- tInTime	string	format: date-time	Ja				Zeitpunkt Ankunft bei vidalStatuswert 2A oder 3: Zeit- punkt des Eintreffens von Status 3A. Zeitpunkt des geplanten Einsatzes bei Statuswert 7. Sonst null-Wert.	20230709:04:30, null
./time- Span	string	pattern: ^\d{1,2}\:\d{1,2}\$	Ja				Zeit-Angabe, falls vidalStatus = 2, 2A, 3, 7. Sonst null- Wert: vidalStatus = 2: Zeit bis 3 (geschätzter Zeitbedarf von Anforderung bis in Marsch gesetzt) vidalStatus = 2A, 3: Ankunft 3A (geschätzte Ankunft im Bereitstellungsraum)	04:30, null

JSON-Pro- perty	JSON-Daten- typ	Datentyp- Einschränkung	Pflichtfeld für				Beschreibung	Beispiel
			Anlage	Ände- rung	Über- sicht	Detail- sicht		
							vidalStatus = 7: Einsatzzeitpunkt (geplanter Zeitpunkt des Einsatzes der Einheit) Format: Zeit (hh:mm)	
/subU- nits	ar- ray[ob- ject]	Anzahl: n	Nein	Nein	Nein	Ja	Untereinheiten, eine feste Anzahl der Elemente wird bei der Neuanlage festgelegt. In allen Fällen muss das ganze Array vollständig übertragen werden.	
./ordi- nal	integer	minimum = 1 maximum = 99	Ja				Feste laufende Nummer der Untereinheit	1
./type	string	pattern: ^\\d{1,2}\\. \\d{1,2}\\. \\d{1,3}\$ (Code-Liste)	Ja				LKEArtID der Untereinheit (aus Schlüsselliste LKEArt)	05.02.002
./id	string	pattern: ^E\\d{8}\\. \\d{10}\$	Ja				Identifikator der Untereinheit, null-Wert, wenn keine Untereinheit zugewiesen	E05100000.000100 0001, null
/re- sources	ar- ray[ob- ject]	Anzahl: n	Nein	Nein	Nein	Ja	Bei Status 3, 3A und 4: Angabe der IDs der eingesetzten Ressourcen. Eine feste Anzahl der Elemente wird bei der Neuanlage festgelegt. In allen Fällen muss das ganze Array vollständig übertragen werden.	

JSON-Pro- perty	JSON-Daten- typ	Datentyp- Einschränkung	Pflichtfeld für				Beschreibung	Beispiel
			Anlage	Ände- rung	Über- sicht	Detail- sicht		
./ordinal	integer	minimum = 1 maximum = 99	Ja				Feste laufende Nummer der Ressource	1
./type	integer	minimum: 1 maximum: 999 (Code-Liste)	Ja				PKKATALOGID der Ressource (aus Ressourcenkatalog IG-NRW)	89
./id	string	pattern: ^[Rr]\d{8}\.\d{10}\$	Ja				Identifikator der Ressource	R11111111.222222 2222 auch hier null, wenn leer

Tabelle 4.8: Schema Einheit

4.3.1.3 Beispiel (gekürzt)

```
{
  "schemaUseCase": "update",
  "unitRequestId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454",
  "unit": {
    "id": "E05100000.0000000001",
    "operator": "05124000",
    "type": "05.02.001",
    "vidalStatus": "2"
  },
  "subUnits": [
```

```

{
    "ordinal": 1,
    "type": "05.02.002",
    "id": "E05100000.0000000011"
},
{
    "ordinal": 2,
    "type": "05.02.003",
    "id": null
}],
"resources": [
{
    "ordinal": 1,
    "type": "89",
    "id": "R11111111.222222222"
},
{
    "ordinal": 2,
    "type": "90",
    "id": "R11111111.333333333"
}]
}

```

4.3.2 Schema – Einheit-Referenz – v1.0

4.3.2.1 Schema-Bezeichner

unitLikeRef

4.3.2.2 Schema

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/schemaUse-Case	enum[string]	[get, delete]	Ja	Verwendung: get - zu Einheit-Abfrage, delete - zum Löschen einer Einheit
/unitRequestId	string[uuid]	UUID	Ja	Request-ID
/unitId	string	pattern: ^E\d{8}\.\d{10}\$	Ja	LKEID (Owner.LfdNr), Format: E+AGS + lfd.Nr. (10 Stellen) Wird vergeben durch Kreis/Stadt = Einsatzleitsystem
/unitShortView	boolean	n/a	Nein	Bei einer Einheit-Abfrage: ob nur Übersicht oder Detailsicht geliefert werden soll. Default: false (es soll Detailsicht geliefert werden)

Tabelle 4.9: Schema Einheit-Referenz

4.3.2.3 Beispiel

Abfrage zu einer Einheit

```
{
  "schemaUseCase": "get",
  "unitRequestId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454",
  "unitId": "E05100000.0000000001"
}
```

4.3.3 Schema – Einheitenfilter – v1.0

4.3.3.1 Schema-Bezeichner

unitFilter

4.3.3.2 Schema

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/unitRequestId	string[uuid]	UUID	Ja	Request-ID
/type	string	pattern: ^\d{1,2}\.\d{1,2}\.\d{1,3}\$ (Code-Liste)	Ja	Typ der Einheit (aus Schlüsselliste LKEArt)
/geoArea	array[string]	pattern: ^\d{8}\$	Nein	geographischer Bereich, AGS, 8-Stellig

Tabelle 4.10: Schema Einheitenfilter

4.3.3.3 Beispiel

Filter-Abfrage

```
{
  "unitRequestId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454",
  "type": {"codeListVersion": "0.1", "code": "05.02.002"},
  "geoArea": ["05011300", "05011400"]
}
```

4.3.4 Schema – Einheitenübersicht – v1.0

4.3.4.1 Schema-Bezeichner

unitList

4.3.4.2 Schema

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/unitRequestId	string[uuid]	UUID	Ja	ID des entsprechenden Requests.
/units	array[unit]		Ja	Liste von Einheiten-Kurzbeschreibungen unit.schemeUseCase = "shortView"

Tabelle 4.11: Schema Einheitenübersicht

4.3.4.3 Beispiel (gekürzt)

```
{
  "unitRequestId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454",
  "units": [
    {
      "schemaUseCase": "shortView",
      "id": "E05100000.0000000001",
      "type": 1,
      ...
    },
    {
      "schemaUseCase": "shortView",
      "id": "E05100000.0000000002",
      "type": 1,
      ...
    }
  ]
}
```

4.3.5 Schema – Einheitenfehler – v1.0

4.3.5.1 Schema-Bezeichner

`unitLikeError`

4.3.5.2 Schema

Das Schema ist identisch mit dem Schema – Fehlermeldung – v1.0. Dazu kommt folgendes Attribut:

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/unitRequestId	string[uuid]	UUID	Ja	Request-ID

Tabelle 4.12: Schema Einheitenfehler

4.3.5.3 Fehler-Codes

VIDaL-Anwendung Einheiten definiert folgende Fehlermeldungen:

Fehler-Code (errorCode)	Erweiterter Fehler- Code (extendedError- Code)	Fehlerursache (errorReason)	Anwendungsfall	Beschreibung
409	1	Eine Einheit mit der LKE-ID <id> bereits vorhanden	Neuanlage einer Einheit	Eine Einheit mit der im gelieferten Datensatz an- gegebenen LKE-ID liegt in der LDB bereits vor.
409	2	Operation für LKE-ID <id> im Status <status> unzulässig	Löschen einer Einheit	Die Einheit ist nicht in einem der Status 2, 6, 8.
404		Eine Einheit mit der LKE-ID <id> nicht gefunden	Änderung einer Einheit, Löschen einer Einheit, Datenabfrage zu einer be- stimmten Einheit	Eine Einheit mit der im gelieferten Datensatz an- gegebenen LKE-ID existiert in der LDB nicht.
401		Der Sender ist kein Verwalter der Einheit mit LKE-ID <id>	Änderung einer Einheit, Löschen einer Einheit	Der Sender ist kein Verwalter der Einheit - nur der Verwalter einer Einheit darf diese Ressource aktualisieren oder löschen.

Tabelle 4.13: Fehler-Codes Einheitenfehler

4.4 Spezifikation Ressourcen

Hier werden die Payload-Schemata für 3.3.5 Ressourcen dokumentiert.

Siehe auch:

- "VIDaL - AP2 Medien und Strecken v10" (für Bezeichner und Pflichtfelder)
- "VIDaL AP1 Anl4 Datensatz Ressourcen in VIDaL v10"
- "VIDaL Inbetriebnahme und Pilotbetrieb v02"

In Anlehnung an die REST-Prinzipien definieren die Schemata folgende Repräsentationen zur Veränderung des Anwendungszustandes durch einen Meldungs austausch:

- Ressource - repräsentiert eine Ressource mit ihren Eigenschaften
- Ressourcenübersicht - repräsentiert eine Liste von mehreren Ressourcen
- Ressource-Referenz - repräsentiert eine Referenz auf eine konkrete Ressource
- Ressourcenfilter - repräsentiert einen Filter mit mehreren Filterkriterien. Ergebnis der Anwendung eines Filters ist eine Ressourcenübersicht

Darüber hinaus wird ein Schema zur Darstellung eines Fehlers bei der Veränderung des Anwendungszustandes definiert:

- Ressourcenfehler

Im Gegensatz zu den REST-Paradigma werden die gewünschten Operationen direkt als Teil einer Repräsentation übertragen - Attribut schemaUse-Case.

4.4.1 Schema – Ressource – v1.0

An dieser Stelle wird das Basis-Schema für die Darstellung von Ressourcendaten für die Abfrage, für das Anlegen und für das Aktualisieren von Ressourcen definiert.

4.4.1.1 Schema-Bezeichner

resource

4.4.1.2 Schema

Anmerkung: Die Spalte „Nur für resourceType“ gibt an, für welchen Ressourcentyp entsprechendes Feld sinnvoll ausfüllbar ist. Ist das Feld wegen dieser Bedingung nicht sinnvoll ausfüllbar, sollte bei „Pflichtfeld=Ja“ hier null eingetragen werden.

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflichtfeld für				Nur für Resource-Type	Beschreibung
			Anlage	Änderung	Übersicht	Detail-sicht		
/schemaUseCase	enum[string]	[create, update, shortView, fullView]	Ja	Ja	Ja	Ja	1,2,3	Verwendung: create - Ressourceanlage, update - Aktualisieren einer Ressource, shortView - Ressource-Sicht mit den wichtigsten Angaben (Übersicht), fullView - komplette Ressource-Sicht (Detailsicht)
/resource-RequestId	string[uuid]	UUID	Ja	Ja	Nein	Ja	1,2,3	ID des entsprechenden Requests. Wenn eine Ressource-Übersicht als Element der LKE-Liste geliefert wird, dann wird entsprechende requestId auf der Listenebene übertragen und nicht in jedem Listenelement.
/resource-ManagingKt	string	OID	Nein	Nein	Ja	Ja	1,2,3	OID des Eigentümers der Ressource. Das Feld ist Read Only und wird bei den Verwendungen "create" und "update" ignoriert . Der KT - Anleger einer Ressource (siehe oben Verwendung "create") wird als Ressourceneigentümer festgelegt. Siehe auch das Konzept Der Eigentümer einer Ressource.
/resource	object		Ja	Ja	Ja	Ja		Allgemeine Angaben zu Ressourcen
./id	string	pattern: ^[Rr]\d{8}\.\d{10}\$	Ja				1,2,3	Eindeutiger Schlüssel, aus AGS.LfdNr in der Form Rxxxxxxx.nnnnnnnnn

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflichtfeld für				Nur für Resource-Type	Beschreibung
			Anlage	Änderung	Übersicht	Detailsicht		
								Wird vergeben durch Kreis/Stadt = Einsatzleitsystem
./type	integer	Embedded Codeliste: 1, 2, 3	Ja				1,2,3	Typ der Ressource, 1=Fahrzeug, 2=Massenresource/Material, 3=Sonstiges
./typeCatalogId	integer	Codeliste: 1 bis 999	Ja				1,2,3	Codeliste Ressourcenkatalog: siehe „VIDaL AP1 Anl3 Ressourcenkatalog IG-NRW“.
./locationAGS	string	^\d{8}\$	Ja				1,2,3	Geo-Ortung nach AGS (8-stellig)
./organisationRef	integer		Ja				1,2,3	Referenz Organisation, siehe Anl. 5, Blatt TORGANISATION. Organisationen werden durch die KT im Rahmen der VIDaL-Anwendung Organisationen verwaltet. Siehe Spezifikation Organisationen.
./modificationDate	string	format: date-time	Nein				1,2,3	Änderungsdatum (automatisch vergeben bei jeder Änderung von Daten, jedoch nicht bei Änderung des Status). Änderungsdatum wird sowohl in der Übersicht als auch in der Detailsicht mitübertragen.
./numberAvailable	integer	minimum: 0 maximum: 999999	Ja				1,2,3	Anzahl der anforderbaren / abgebbaren Ressourcen, bei Fahrzeugen = 1, sonst Anzahl der anforderbaren Ressourcen.

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflichtfeld für				Nur für Resource-Type	Beschreibung
			Anlage	Änderung	Übersicht	Detail-sicht		
./numberTotal	integer	minimum: 0 maximum: 999999	Ja				2,3	Anzahl der Ressourcen inkl. der nicht-anforderbaren Ressourcen.
./packingSize	number	minimum: 0 maximum: 99999.99 zwei Nachkommastellen	Ja				2	Verpackungsgröße, wenn für Ressourcentyp sinnvoll: Anzahl je Verpackungsgröße. null-Wert, wenn für Ressourcentyp nicht sinnvoll.
./packingUnit	enum	cbm,kg,l,pcs	Ja				2	Verpackungseinheit: Basis ist eine abschließende Liste möglicher Einheiten (z.B. cbm, kg, l (=Liter), pcs=(Stück)). null-Wert, wenn für Ressourcentyp nicht sinnvoll.
./current-Status	integer	Embedded Codeliste: 1, 2, 3, 4, 6, 7, 8	Ja				1	Kontinuierlicher Status der Ressource aus Leitstelle, entspricht dem FMS-Status mit Ausnahme von Stati 0, 5 und 9. 0 = Priorisierter Sprechwunsch; 1 = Einsatzbereit Funk; 2 = Einsatzbereit Wache; 3 = Einsatzübernahme; 4 = Einsatzort; 5 = Sprechwunsch; 6 = Nicht einsatzbereit; 7 = Einsatzgebunden; 8 = Bedingt verfügbar; 9 = Quittung/Fremdanmeldung

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflichtfeld für				Nur für Resource-Type	Beschreibung
			Anlage	Änderung	Übersicht	Detail-sicht		
./current-StatusTimestamp	string	format: date-time	Ja				1,2,3	Zeitstempel der letzten Statusänderung.
./operationalReadiness	boolean		Ja				1,2,3	Einsatzbereit: true=Ja, false=Die Ressource kann von anderen Organisationen nicht angefordert werden.
./readinessTimestamp	string	format: date-time (/oder leer)	Ja				1,2,3	Wenn Ressource nicht einsatzbereit, also bei RESSOURCE_STATUS=6: Datum, wann Ressource (voraussichtlich) wieder einsatzbereit sein wird. Sonst (RESSOURCE_STATUS != 6): null
./comment	string	maxLength: 2000	Nein				1,2,3	Freifeld für Bemerkungen
./stateConcept	boolean		Ja				1,3	Ist das Fz (oder Sonstige) Teil des Landeskonzpts: true=ja, false=nein
/vehicle	object		Nein	Nein	Nein	Nein	1	Angaben zu Fahrzeugen
./name	string	maxLength: 100	Nein				1	Offizielle Fz.-Bezeichnung lt. Zulassung (Herstellertyp)

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflichtfeld für				Nur für Resource-Type	Beschreibung
			Anlage	Änderung	Übersicht	Detail-sicht		
./licencePlateNumber	string	pattern: ^NRW 8-[1-9]{1}[0-9]{0,4}[EH]{0,1}\$ ^[A-ZÄÖÜ]{1,3} [A-ZÄÖÜ]{1,2}[1-9]{1}[0-9]{0,4}[EH]{0,1}\$ ^[A-ZÄÖÜ]{1,3} [1-9]{1}[0-9]{0,5}[EH]{0,1}\$ ^THW[89]{1}[0-9]{3,5}\$	Nein				1	KFZ-Kennzeichen
./chassisSupplier	string	maxLength: 100	Nein				1	Hersteller des Fz-Fahrgestells
./chassisNumber	string	maxLength: 17	Ja				1	Fahrgestellnummer
./registrationDate	string	format: date-time	Nein				1	Datum der Erstzulassung
./yearOfManufacture	integer	minimum: 0 maximum: 9999	Ja				1	Baujahr
./unloadedWeight	integer	minimum: 0 maximum: 99999	Ja				1	Leergewicht, Einheit=kg
./length	integer	minimum: 0 maximum: 99999	Nein				1	Länge, Einheit=Millimeter

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflichtfeld für				Nur für Resource-Type	Beschreibung
			Anlage	Änderung	Übersicht	Detail-sicht		
./width	integer	minimum: 0 maximum: 9999	Nein				1	Breite, Einheit=Millimeter
./height	integer	minimum: 0 maximum: 9999	Nein				1	Höhe, Einheit=Millimeter
./super-structure-Supplier	string	maxLength: 100	Nein				1	Hersteller des Fz-Aufbaus
./super-StructureNumber	string	maxLength: 30	Nein				1	Produktionsnr.des Fz.-Aufbaus
./licenceClass	string	maxLength: 3	Nein				1	Benötigte Führerscheinklasse
./licenceId	string	maxLength: 100	Nein				1	Nummer des Fz.-Briefs
./footprint	number	minimum: 0 maximum: 999,99 Zwei Nachkommastellen.	Nein				1	Stellfläche des Fz., Einheit=Quadratmeter
./pull-Weight	integer	minimum: 0 maximum: 99999	Nein				1	Zuggewicht des Fz., Einheit=kg

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflichtfeld für				Nur für Resource-Type	Beschreibung
			Anlage	Änderung	Übersicht	Detail-sicht		
./checkTimestamp	string	format: date-time	Nein				1	Termin f. Hauptuntersuchung
./internal-Name	string	maxLength: 100	Nein				1	Freifeld für interne Bezeichnung der Ressource
./ownerRef	integer	Embedded Codeliste: 1-7	Ja				1	Referenz Eigentümer, siehe Anlage 5 Tabellen für Stammdaten Ressourcen - Blatt TRES_EIGENTUEMER. Basis ist eine abschließende Liste der möglichen Eigentümer (u.a. auch "Bund"). 1=Bund; 2=Land; 3=Bezirksregierung; 4=Kreis; 5=eigene Organisation; 6=Gemeinde; 7=IdF
./crewMembers	string	pattern: ^\d{1,2}\/\d{1,2}\/\d{1,2}\$	Nein				1	Anzahl der Führer, Anzahl der Unterführer, und Anzahl der Einsatzkräfte - durch "/" getrennt, z.B.: "2/4/8"
./national-Description-Ref	integer	Embedded Codeliste: 0-18	Ja				1	Bundesbezeichnung des Fahrzeugs, siehe Anlage 5 Tabellen für Stammdaten Ressourcen - Blatt TRES_BUND Codeliste enthält zwei Spalten: "Bezeichnung" und "Kürzel". Sonderwert: 0 – kein Bundesfahrzeug. Beispiel (alle Werte wären zu viel für diese Tabelle):

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflichtfeld für				Nur für Resource-Type	Beschreibung
			Anlage	Änderung	Übersicht	Detail-sicht		
								1=Gerätewagen Dekontamination Personal, Kürzel=GW Dekon P;
./disaster-ControlRef	integer	Embedded Codeliste: 1-49	Nein				1	Referenz KFZ Kat-Schutz, siehe Anlage 5 Tabellen für Stammdaten Ressourcen - Blatt TRES_KATS_KFZ_ART Codeliste enthält eine Spalte. Beispiel (alle Werte wären zu viel für diese Tabelle): 1=ABC-ErkKW
./purchaser-Ref	integer	Embedded Codeliste: 1-7	Ja				1	Referenz Beschaffer, siehe Anlage 5 Tabellen für Stammdaten Ressourcen - Blatt TRES_EI-GENTUEMER. 1=Bund; 2=Land; 3=Bezirksregierung; 4=Kreis; 5=eigene Organisation; 6=Gemeinde; 7=IdF
./jobNum-ber	string	maxLength: 50	Nein				1	Nr. des Beschaffungsauftrags
./assignmentTimestamp	string	format: date-time	Nein				1	Datum der Zuweisung

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflichtfeld für				Nur für Resource-Type	Beschreibung
			Anlage	Änderung	Übersicht	Detail-sicht		
./provider-Ref	integer	Embedded Codeliste: 1-7	Ja				1	Referenz Träger, siehe Anlage 5 Tabellen für Stammdaten Ressourcen - Blatt TRES_EIGENTUEMER 1=Bund; 2=Land; 3=Bezirksregierung; 4=Kreis; 5=eigene Organisation; 6=Gemeinde; 7=IdF
./oldLicencePlateNumber1	string	maxLength: 13	Nein				1	1. altes Kennzeichen, ggf. ausfüllen, falls nicht vorhanden leer lassen
./oldLicencePlateNumber2	string	maxLength: 13	Nein				1	2. altes Kennzeichen, ggf. ausfüllen, falls nicht vorhanden leer lassen
./oldLicencePlateNumber3	string	maxLength: 13	Nein				1	3. altes Kennzeichen, ggf. ausfüllen, falls nicht vorhanden leer lassen
/radio	array[object]		Nein	Nein	Nein	Nein	1,3	Angaben zu Funkgeräten auf Fahrzeugen (Mehrfachangaben möglich)
./subscriberNumber	integer	minimum: 0 maximum: 9999999999999999	Ja				1,3	Ist Funk auf Fz. vorhanden? Ja=Rufnummer (ITSI), nein=0
./opta	string	pattern: ^NW[A-Z0-9-]{10,24}\$	Ja				1,3	zugeordnete OPTA OPTA-Schema mit 10 bis 24 Zeichen!

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflichtfeld für				Nur für Resource-Type	Beschreibung
			Anlage	Änderung	Übersicht	Detail-sicht		
								Basis ist Liste der registrierten OPERATIV TAKTISCHEN ADRESSEN.
								null falls Wert nicht vorhanden

Tabelle 4.14: Schema Ressource

4.4.1.3 Beispiel (gekürzt)

```
{
  "schemaUseCase": "update",
  "resourceRequestId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454",
  "resource": {
    "id": "R12345678.1234567890",
    "type": 1,
    "alterationDate": "2023-01-13T20:20:39+00:00",
    "numberAvailable": 1
  },
  "vehicle": {
    "name": "CAT 325FLCR",
    "unloadedWeight": 2500,
    "internalName": "CAT 325FLCR Kettenbagger"
  },
  "radio": [{
    "subscriberNumber": 0
  }]
}
```

}

4.4.2 Schema – Ressource-Referenz – v1.0

4.4.2.1 Schema-Bezeichner

resourceRef

4.4.2.2 Schema

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/schemaUseCase	enum[string]	[get, delete]	Ja	Verwendung: get - zu Ressource-Abfrage, delete - zum Löschen einer Ressource
/resourceRequestId	string[uuid]	UUID	Ja	Request-ID
/id	string	pattern: ^[Rr]\d{8}\.\d{10}\$	Ja	Eindeutiger Schlüssel, aus AGS.LfdNr in der Form Rxxxxxxx.nnnnnnnnn Wird vergeben durch Kreis/Stadt = Einsatzleitsystem
/resourceShortView	boolean	n/a	Nein	Bei einer Ressource-Abfrage: ob nur Übersicht oder Detailsicht geliefert werden soll. Default: false (es soll Detailsicht geliefert werden)

Tabelle 4.15: Schema Ressource-Referenz

4.4.2.3 Beispiel

Löschen einer Ressource

```
{
  "schemaUseCase": "delete",
  "resourceRequestId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454",
  "id": "R12345678.1234567890"
}
```

4.4.3 Schema – Ressourcenfilter – v1.0

4.4.3.1 Schema-Bezeichner

resourceFilter

4.4.3.2 Schema

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/schemaUseCase	enum[string]	[get]	Ja	Verwendung: z.Z. nur direkte Abfrage möglich. Später vielleicht auch Speichern als Idee für permanente Monitoring von Ressourcenänderungen?
/resourceRequestId	string[uuid]	UUID	Ja	Request-ID
/typeCatalogId	integer	Codeliste: 1 bis 9999	Ja	Codeliste Ressourcenkatalog: siehe „ViDaL AP1 Anl3 Ressourcenkatalog IG-NRW“.
/geoArea	array[string]	pattern: ^\d{8}\$	Nein	geographischer Bereich, AGS-Angabe 8-stellig

Tabelle 4.16: Schema Ressourcenfilter

4.4.3.3 Beispiel

Filter-Abfrage
<pre>{ "schemaUseCase": "get", "resourceRequestId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454", "typeCatalogId": 1, "geoArea": {"01130000", "01140000"} }</pre>

4.4.4 Schema – Ressourcenübersicht – v1.0

4.4.4.1 Schema-Bezeichner

resourceList

4.4.4.2 Schema

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/resourceRequestId	string[uuid]	UUID	Ja	ID des entsprechenden Requests.
/resources	array[resource]		Ja	Liste von Ressource-Kurzbeschreibungen resource.schemeUseCase = "shortView"

Tabelle 4.17: Schema Ressourcenübersicht

4.4.4.3 Beispiel

```
{
  "resourceRequestId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454",
  "resources": [
    {
      "schemaId": "resource",
      "schemaUseCase": "shortView",
      "schemaVersion": "0.2",
      "resource": {
        "id": "R12345678.1234567893",
        "type": 1,
        "locationAGS": "01130000",
        "numberTotal": 999999,
        "packingSize": 99999.99,
        "packingUnit": "kg",
        "stateConcept": false,
        "currentStatus": 4,
        "typeCatalogId": {
          "code": 388,
          "codeListVersion": "0.1"
        },
        "numberAvailable": 999999,
        "organisationRef": 123456,
        "modificationDate": "2024-07-05T10:22:00.137Z",
```

```

        "readinessTimestamp": "2023-12-19T20:10:00Z",
        "operationalReadiness": true,
        "currentStatusTimestamp": "2023-12-19T20:10:00Z"
    },
    ...
},
{
    "schemaId": "resource",
    "schemaUseCase": "shortView",
    "schemaVersion": "0.2",
    "resource": {
        "id": "R12345678.1234567893",
        "type": 1,
        "locationAGS": "01130000",
        "numberTotal": 999999,
        "packingSize": 99999.99,
        "packingUnit": "kg",
        "stateConcept": false,
        "currentStatus": 4,
        "typeCatalogId": {
            "code": 388,
            "codeListVersion": "0.1"
        },
        "numberAvailable": 999999,

```

```

        "organisationRef": 123456,
        "modificationDate": "2024-07-05T10:22:00.137Z",
        "readinessTimestamp": "2023-12-19T20:10:00Z",
        "operationalReadiness": true,
        "currentStatusTimestamp": "2023-12-19T20:10:00Z"
    },
    ...
}]
}

```

4.4.5 Schema – Ressourcenfehler – v1.0

4.4.5.1 Schema-Bezeichner

resourceError

4.4.5.2 Schema

Das Schema ist identisch mit dem Schema - Fehlermeldung - v1.0. Hinzu kommen folgende Attribute:

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/resourceRequestId	string[uuid]	UUID	Ja	Request-ID

Tabelle 4.18: Schema Ressourcenfehler

4.4.5.3 Fehler-Codes

VIDaL-Anwendung Ressourcen definiert folgende Fehlermeldungen:

Fehler-Code (errorCode)	Fehlerursache (errorReason)	Anwendungsfall	Beschreibung
409	Eine Ressource mit der Ressource-ID <id> bereits vorhanden	Neuanlage einer Ressource	Eine Ressource mit der im gelieferten Datensatz angegebenen Ressource-ID liegt in der LDB bereits vor.
404	Eine Ressource mit der Ressource-ID <id> nicht gefunden	Änderung einer Ressource, Löschen einer Ressource, Datenabfrage zu einer bestimmten Ressource	Eine Ressource mit der im gelieferten Datensatz angegebenen Ressource-ID existiert in der LDB nicht.
401	Der Sender ist kein Eigentümer der Ressource mit Ressource-ID <id>	Änderung einer Ressource, Löschen einer Ressource	Der Sender ist kein Eigentümer der Ressource - nur der Eigentümer einer Ressource darf diese Ressource aktualisieren oder löschen.

Tabelle 4.19: Fehler-Codes Ressourcenfehler

4.5 Spezifikation Kontinuierliche Einsatzstatistik

Hier werden die Payload-Schemata für die 3.3.6 Kontinuierliche Einsatzstatistik dokumentiert.

Der Initialstand wurde anhand von "VIDaL - AP2 Medien und Strecken v10.pdf" und "VIDaL AP1 Anl7 Inhalte Kontinuierliche Einsatzstatistik v10.pdf" erstellt.

In Anlehnung an die REST-Prinzipien definieren die Schemata folgende Repräsentationen zur Veränderung des Anwendungszustandes durch einen Meldungsaustausch:

- Einsatzstatistik - repräsentiert die aktuelle Einsatzstatistik zu einer durchführenden Leitstelle
- Einsatzstatistiken - repräsentiert eine Liste von Einsatzstatistiken zu mehreren durchführenden Leitstellen
- Einsatzstatistikfilter - repräsentiert einen Filter mit mehreren Filterkriterien. Ergebnis der Anwendung eines Filters ist die Einsatzstatistiken zu mehreren durchführenden Leitstellen

4.5.1 Schema – Einsatzstatistik – v1.0

4.5.1.1 Schema-Bezeichner

missionStatistics

4.5.1.2 Schema

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung	Beispiel
/statisticsDateTime	string	format: date-time	Ja	Datum und Zeit der Datenerfassung	2023-11-13T20:20:39+00:00
/sender	object		Ja		
./authority	string	maxLength: 200	Ja	Behörde (z.B. Übernahme automatisch aus Registry)	
./territorialEntity-Reported	string	^\d{8}\$	Ja	AGS-Angabe (8-stellig)	
/missionTypes	array[object]	Anzahl: n	Nein	Einsatzdefinition	
./type	string	Codeliste	Nein	eindeutige Statistiktyp	Entsprechend "VIDaL AP1 Anl72 Einsatzstichworte & Inhalte Kontinuierliche Einsatzstatistik"
/missionTypes/status	object		Nein	Statistik über Einsatzzustände	
./announced	integer	minimum: 0 maximum: 99999	Nein	Einsätze vorangemeldet	

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung	Beispiel
./active	integer	minimum: 0 maximum: 99999	Nein	Einsätze aktiv / laufend	
./time	integer	minimum: 0 maximum: 99999	Nein	Einsatzdauer Einsätze aktiv / laufend	
./finished	integer	minimum: 0 maximum: 99999	Nein	Einsätze zu beenden	
/missionTypes/persons	object		Nein	Statistik über Patienten/Betroffene	
./injured1	integer	minimum: 0 maximum: 99999	Nein	Patienten Sichtungskategorie 1 / T1	
./injured2	integer	minimum: 0 maximum: 99999	Nein	Patienten Sichtungskategorie 2 / T2	
./injured3	integer	minimum: 0 maximum: 99999	Nein	Patienten Sichtungskategorie 3 / T3	
./injured4	integer	minimum: 0 maximum: 99999	Nein	Patienten Sichtungskategorie 4 / T4	
./missing	integer	minimum: 0 maximum: 99999	Nein	Zahl der Vermissten	
./involved	integer	minimum: 0 maximum: 99999	Nein	Betroffene / zu Betreuende	

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung	Beispiel
/missionTypes/resources	object		Nein	Statistik über Einsatzmittel	
./alarmed	integer	minimum: 0 maximum: 99999	Nein	EM alarmiert	
./status3	integer	minimum: 0 maximum: 99999	Nein	EM Status 3 / Einsatzübernahme	
./status4	integer	minimum: 0 maximum: 99999	Nein	EM Status 4 / Einsatzort	
./status7	integer	minimum: 0 maximum: 99999	Nein	EM Status 7 / Einsatzgebunden	
./status8	integer	minimum: 0 maximum: 99999	Nein	EM Status 8 / bedingt verfügbar	

Tabelle 4.20: Schema Einsatzstatistik

4.5.1.3 Beispiel (gekürzt)


```
{
  "sender": {
    "authority": "Bezirksregierung Düsseldorf",
    "territorialEntityReported": "05124000"
  },
  "missionTypes": [
    {
      "type": 5,
      "status": {
        "announced": "3",
        "active": "5"
      },
      "persons": {
        "injured1": "7",
        "injured2": "12"
      },
      "resources": {
        "alarmed": "6",
        "s3": "4"
      }
    },
    {
      "type": 33,
      "status": {
        "announced": "7",
        "active": "14"
      }
    }
  ]
}
```

Einsatzstatistik

```
    "persons": {
      "injured1": "122",
      "injured2": "230"
    },
    "resources": {
      "alarmed": "25",
      "status3": "30"
    }
  }
]
```

4.5.2 Schema – Einsatzstatistikfilter – v1.0

4.5.2.1 Schema-Bezeichner

missionStatisticsFilter

4.5.2.2 Schema

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/missionRequestId	string[uuid]	UUID	Ja	Request-ID
/missionKind	string	maxlength: 20	Nein	Einsatzart aus der Codeliste (aus dem Katalog Hessen)
/missionKeyword	string	maxlength: 20	Nein	Einsatzstichwort aus der Codeliste (aus dem Katalog Hessen)

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/geoArea	array[string]	pattern: ^\d{8}\$	Nein	geographischer Bereich, AGS-Angabe 8-stellig

Tabelle 4.21: Schema Einsatzstatistikfilter

4.5.2.3 Beispiel

Einsatzstatistikabfrage
<pre>{ "missionRequestId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454", "missoinKind": "Brandeinsatz", "geoArea": { "05011300", "05011400"} }</pre>

4.5.3 Schema – Einsatzstatistiken – v1.0

4.5.3.1 Schema-Bezeichner

missionStatisticsList

4.5.3.2 Schema

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/missionRequestId	string[uuid]	UUID	Ja	ID des entsprechenden Requests.
/missionStatistics	array[<u>missionStatistics</u>]	N/A	Ja	Liste von Einsatzstatistiken

Tabelle 4.22: Schema Einsatzstatistiken

4.5.3.3 Beispiel (gekürzt)

```
{
  "missionRequestId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454",
  "missionStatistics": [
    {
      "sender": {
        ...
        "mail": "test@test.com"
      },
      "missions": [
        {...},
        {...}
      ],
      ...
    },
    {
      "sender": {
        ...
        "mail": "test@test.com"
      },
      "missions": [
        {...},
        {...}
      ],
      ...
    }
  ]
}
```

4.6 Spezifikation Adhoc-Information

Hier werden die Payload-Schemata für Adhoc-Information dokumentiert.

4.6.1 Schema – Adhoc-Information – v1.0

Schema für die Darstellung von Freitext-Nachrichten.

4.6.1.1 Schema-Bezeichner

adhocInformation

4.6.1.2 Schema

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/subject	string	maxLength: 2000	Ja	Die Betreffzeile.
/text	string	maxLength: 10000	Nein	Der Text der Meldung.

Tabelle 4.23: Schema Adhoc-Information

4.6.1.3 Beispiel

```
{
  "subject": "Testbetreff"
  "text": "Dies ist der Inhalt einer Adhoc-Meldung."
}
```


4.7 Spezifikation Organisationen

Das ViDaL System vernetzt die Organisationen mit Sicherheitsaufgaben des Landes NRW. Die ViDaL-Kommunikationsteilnehmer (KT) sind für die Pflege ihrer eigenen und untergeordneten Organisationen zuständig. Außerdem kann ein LDB-Administrator in der Rolle Organisationsadministrator Daten aller Organisationen bearbeiten, Organisationen anlegen und löschen.

Hier werden die Payload-Schemata für Organisationen dokumentiert. Der Initialstand wurde anhand von "ViDaL AP1 Anl5 Tabellen für Stammdaten Ressourcen" erstellt.

In Anlehnung an die REST-Prinzipien definieren die Schemata folgende Repräsentationen zur Veränderung des Anwendungszustandes durch einen Meldungs austausch:

- Organisation - repräsentiert eine Organisation mit ihren Eigenschaften
- Organisationsübersicht - repräsentiert eine Liste von Organisationen
- Organisations-Referenz - repräsentiert eine Referenz auf eine konkrete Organisation
- Organisationsfilter - repräsentiert einen Filter mit mehreren Filterkriterien. Ergebnis der Anwendung eines Filters ist die Organisationsübersicht

Darüber hinaus wird ein Schema zur Darstellung eines Fehlers bei der Veränderung des Anwendungszustandes definiert:

- Organisationsfehler

Im Gegensatz zum REST-Paradigma werden die gewünschten Operationen direkt als Teil einer Repräsentation übertragen - Attribut `schemaUseCase`.

4.7.1 Schema – Organisation – v1.0

4.7.1.1 Schema-Bezeichner

organisation

4.7.1.2 Schema

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflichtfeld für				Beschreibung	Beispiel
			An-lage	Ände-rung	Über-sicht	Detail-sicht		
/schemaUseCase	enum[string]	[create, update, shortView, fullView]	Ja	Ja	Ja	Ja	Verwendung: create - Anlegen einer Organisation, update - Aktualisieren einer Organisation, shortView - Organisations-sicht mit wichtigsten Angaben (Über-sicht), fullView - komplette Organisations-sicht (Detailsicht).	create
/orgRequestId	string[uuid]	UUID	Ja	Ja	Nein	Ja	ID des entsprechenden Requests. Wenn eine Organisationssicht als Element der Organisationsliste geliefert wird, dann wird entsprechende requestId auf der Listenebene übertragen und nicht in jedem Listenelement.	f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454
/lastUpdateDateTime	string	format: date-time	Nein	Nein	Nein	Ja	Zeitpunkt der letzten Änderung/Anpassung der Organisationseigenschaften. Das Feld wird in der Server-Implementierung in der LDB automatisch angepasst.	2023-04-23T18:25:43.511Z
/orgManagingKt	string	OID	Nein	Nein	Ja	Ja	OID des Verwalters der Organisation.	1.2.3.4.5.6

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflichtfeld für				Beschreibung	Beispiel
			An-lage	Ände-rung	Über-sicht	Detail-sicht		
							<p>Das Feld darf durch den Verwalter und durch den LDB-Administrator in der Rolle Organisationsadminintrator geändert werden (Übergabe der Rolle Verwalter).</p> <p>Falls beim Anlegen einer Organisation das Feld nicht gesetzt ist, wird der anlegende KT in das Feld automatisch eingetragen.</p>	
/orgld	string	pattern: ^O\d{8}\.\d{10}\$	Ja	Ja	Ja	Ja	<p>Eindeutige ID der Organisation. Format: O<AGS>.<Lfd.Nr.(10 Stellen)></p> <p>Die OrgID wird beim Anlegen der Organisation (create) vom verwaltenden ELS vergeben.</p> <p>Die ID wird in der Server-Implementierung in der LDB automatisch vergeben.</p>	005100000.0001000000
/organisation	object		Ja	Ja	Ja	Ja		
/name	String	maxLength	Ja				Bezeichnung der Organisation	WF thyssenkrupp Steel Europe AG, Duisburg
./phone	string	maxLength=50 pattern:	Ja				Telefon der Organisation	+491517953677

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflichtfeld für				Beschreibung	Beispiel
			Anlage	Änderung	Übersicht	Detail-sicht		
./emailAlarm	string	maxLength=100 format: email	Nein				Mailadresse zur Arlamierung der Organisation / Leitstelle	alarm@myorg.de
./locationAGS	string	pattern: ^\d{8}\$	Ja	Nein	Ja	Ja	Gemeindekennziffer der Organisation / des Ortes (AGS)	
./orgAddress	object		Nein	Nein	Nein	Nein		
./street	string	maxLength=100	Ja				Straße	„Mustermann-Straße“
./houseNumber	string	maxLength=10	Ja				Hausnummer	"1A"
./postCode	string	maxLength=10	Ja				Postleitzahl	"10115"
./place	string	maxLength=100	Ja					"Berlin"
./orgContactPerson			Nein	Nein	Nein	Nein		
./name	string	maxLength=100	Ja				Ansprechpartner der Organisation	Mustermann
./surname	string	maxLength=100	Ja				Ansprechpartner der Organisation	Max
./phone	string	maxLength=50 pattern: (\(?([d \-])\+ [+V\(\)]\)?([\-] V])?([d]+))	Ja				Telefon des Ansprechpartners der Organisation	+491517953677 Siehe auch https://regex101.com/r/CAVex8/143

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflichtfeld für				Beschreibung	Beispiel
			An-lage	Ände-rung	Über-sicht	Detail-sicht		
./email	string	maxLength=100 format: email	Ja				Mailadresse des Ansprechpartners der Organisation	<u>max.mustermann@myorg.de</u>
/orgToBeAd-dressed	string	pattern: ^O\d{8}\.\d{10}\$	Nein	Nein	Nein	Ja	Eindeutige ID der Organisation, über die Anforderungen an diese Organisation gestellt werden. Ist immer ein Kreis oder eine krfr. Stadt (TYPID = 5).	"000050100.0000000001"
/orgForDele-gation	string	pattern: ^O\d{8}\.\d{10}\$	Nein	Nein	Nein	Ja	Eindeutige ID der Organisation, die eine Stellvertretung für diese Organisation einrichten darf.	"000050100.0000000001"
/orgToBeRe-ported	string	pattern: ^O\d{8}\.\d{10}\$	Nein	Nein	Nein	Ja	Eindeutige ID der Organisation, an die die Jahresstatistik gemeldet wird.	"000050100.0000000001"
/orgStamm-Hio	string	pattern: ^O\d{8}\.\d{10}\$	Nein	Nein	Nein	Ja	Eindeutige ID des Landesverbandes die-ser HIO. Nur bei untergeordneten Hilfsorganisa-tionen gefüllt (TYPID = 13).	"000050100.0000000001"

Tabelle 4.24: Schema Organisation

4.7.1.3 Beispiel

Änderungen der Organisationseigenschaften

```
{
  "schemaUseCase": "update",
  "orgRequestId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454",
  "orgId": "000050100.0000000001",
  "organisation": {
    "emailAlarm": "alarm@myorg.de"
  },
  "orgContactPerson": {
    "email": "max.mustermann@myorg.de"
  }
}
```

4.7.2 Schema – Organisations-Referenz – v1.0

4.7.2.1 Schema-Bezeichner

organisationRef

4.7.2.2 Schema

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/schemaUseCase	enum[string]	[get, delete]	Ja	Verwendung: get - zur Abfrage der Organisationsdaten, delete - zum Löschen einer Organisation

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/orgRequestId	string[uuid]	UUID	Ja	Request-ID
/orgId	string	pattern: ^O\d{8}\.\d{10}\$	Ja	Organisations-ID
/orgShortView	boolean	n/a	Nein	Bei einer Abfrage: ob nur Übersicht oder Detailsicht geliefert werden soll. Default: false (es soll Detailsicht geliefert werden)

Tabelle 4.25: Schema Organisationsreferenz

4.7.2.3 Beispiel

Abfrage Übersichtsdaten zu einer Organisation
<pre>{ "schemaUseCase": "get", "orgRequestId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454", "orgId": "O00005001.00000000001", „orgShortView“: true }</pre>

4.7.3 Schema – Organisationsfilter – v1.0

4.7.3.1 Schema-Bezeichner

organisationFilte

4.7.3.2 Schema

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/orgRequestId	string[uuid]	UUID	Ja	Request-ID
/geoArea	array[string]	pattern: ^\d{8}\\$	Ja	Array von AGS, mindestens 1 Element

Tabelle 4.26: Schema Organisationsfilter

4.7.3.3 Beispiel

Filter-Abfrage: Organisationen nach AGS

```
{
  "orgRequestId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454",
  "geoArea": {"01130000", "01140000"}
}
```

4.7.4 Schema – Organisationsübersicht – v1.0

4.7.4.1 Schema-Bezeichner

organisationList

4.7.4.2 Schema

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/orgRequestId	string[uuid]	UUID	Ja	ID des entsprechenden Requests.

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/organisations	array[organisation]		Ja	Liste der Organisations-Kurzbeschreibungen organisation.schemeUseCase = "shortView"

Tabelle 4.27: Schema Organisationsübersicht

4.7.4.3 Beispiel (gekürzt)

```
{
  "orgRequestId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454",
  "organisations": [
    {
      "schemaUseCase": "shortView",
      "orgId": "005100000.0001000000",
      "organization": {
        "phone": "+49 171 1234567",
        "email": "info@myorg.de"
      },
      ...
    },
    {
      "orgAddress": {
        "street": "Mustermann Straße",
        "houseNumber": 1,
        "postCode": "10115",
        "place": "Berlin"
      },
      ...
    },
    {
      "schemaUseCase": "shortView",
      "orgId": "005100000.0001000001",
      ...
    }
  ]
}
```

```

        "organization": {
    ...
    },
    "orgAddress": {
        "street": "Mustermann Straße",
        ...
    }
    }]
}

```

4.7.5 Schema – Organisationsfehler – v1.0

4.7.5.1 Schema-Bezeichner

organisationError

4.7.5.2 Schema

Das Schema ist identisch mit dem Schema – Fehlermeldung – v1.0. Dazu kommt folgendes Attribut:

JSON-Property	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/orgRequestId	string[uuid]	UUID	Ja	Request-ID

Tabelle 4.28: Schema Organisationsfehler

4.7.5.3 Fehler-Codes

VIDaL-Anwendung Organisationen definiert folgende Fehlermeldungen:

Fehler-Code (errorCode)	Fehlerursache (errorReason)	Anwendungsfall	Beschreibung
404	Eine Organisation mit der ID <orgId> nicht gefunden	Änderung einer Organisation, Löschen einer Organisation, Datenabfrage zu einer bestimmten Organisation	Eine Organisation mit der im gelieferten Datensatz angegebenen Organisations-ID existiert in der LDB nicht.
401	Der Sender ist kein Verwalter der Organisation mit der ID <orgId>	Änderung einer Organisation, Löschen einer Organisation	Der Sender ist kein Verwalter der Organisation - nur der Verwalter einer Organisation oder ein Organisationsadministrator dürfen diese Organisation aktualisieren oder löschen.

Tabelle 4.29: Fehler-Codes Organisationsfehler

4.8 Codelisten

Der Begriff *Codeliste* in der vorliegenden VIDaL-Projektdokumentation hat die gleiche Bedeutung wie der Begriff *Katalog* in der Dokumentation des VIDaL-Expertenforums.

Eine Codeliste beschreibt einen Datentyp mit begrenzter Wertemenge (eine Enumeration) und kann somit grundsätzlich als ein Schema dargestellt werden.

Eine Codeliste wird als Schema dargestellt. So kann sie in dem Nachrichtenschema referenziert und zur Validierung von entsprechenden Werten genutzt werden (per Framework).

Werden Codelisten weiterentwickelt, dann benötigen sie eine Versionierung. Damit Nachrichtenschemata bei Codelistenänderungen nicht angepasst werden müssen, bekommt eine konkrete Meldung außer dem Codelistenwert auch den Verweis auf eine entsprechende Codelistenversion.

Codelisten müssen gepflegt und hinzugefügt werden. Mit dem LPF Schema Service gibt es dazu bereits ein benutzbares Werkzeug.

Durch die Darstellung der Codelisten als Schema erhält man eine durchgängige maschinenlesbare API-Spezifikation und Dokumentation der Anwendungs-Nachrichten mit expliziter Verbindung zwischen dem Wert in der Nachricht und der entsprechenden Codeliste.

4.8.1 Zusammenhang zwischen Meldung, Nachrichtenschema und Codeliste

Soll eine Meldung Werte einer Codeliste übertragen, so beinhaltet das entsprechende Nachrichtenschema einen Verweis auf das entsprechende Codelistenschema (im Bild: "CodeListA").

Die Meldung an sich beinhaltet dann den eigentlichen Code und die Version der Codeliste, die zum Erweitern des Codes benutzt werden soll.

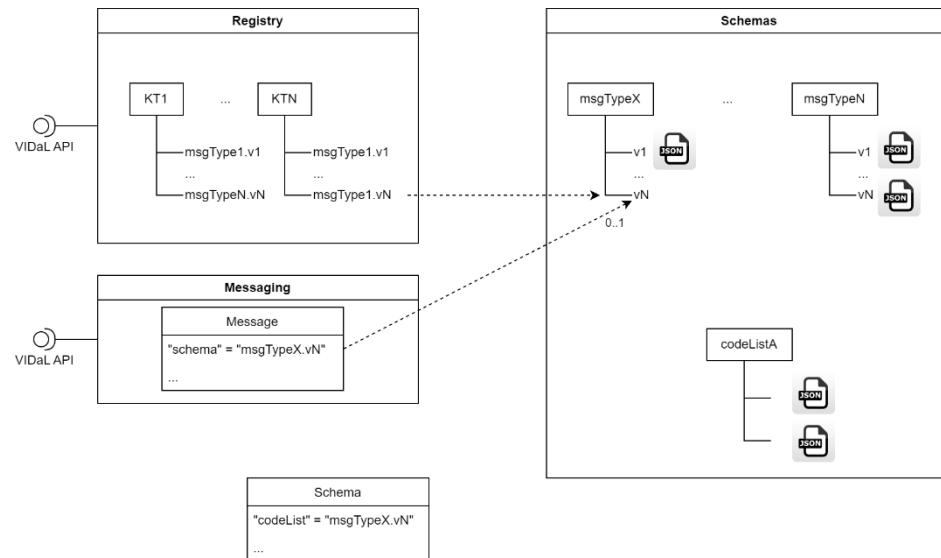


Abbildung 4.1: Zusammenhang zwischen Meldung, Nachrichtenschema und Codeliste

4.8.2 Beispiel

Codelistenschema

```
{
  "$codeListURI": "urn:numbers_codelist",
  "$codeListVersion": "1.0.0",
  "oneOf": [
    {
      "const": 1,
      "title": "Eins",
      "description": "Dies ist die Zahl Eins.",
      "appinfo": {"shortname": "1", "longname": "eins", "englishName": "one"}
    },
    {
      "const": 2,
      "title": "Zwei",
      "description": "Dies ist die Zahl Zwei.",
      "appinfo": {"shortname": "2", "longname": "zwei", "englishName": "two"}
    }
  ]
}
```

Nachrichtenschema

Codelistenschema

```
{
  "type": "object",
  "required": [
    "number"
  ],
  "properties": {
    "number": {
      "type": "object",
      "required": [
        "codeListVersion",
        "code"
      ],
      "properties": {
        "codeListURI": {
          "const": "urn:numbers_codelist"
        },
        "codeListVersion": {
          "type": "string"
        },
        "code": {
          "type": "number"
        }
      }
    }
  }
}
```

Codelistenschema
Meldung
<pre>{ "\$schema": "../schemas/message_schema_with_loose_codelist.json", "number": { "codeListVersion": "1.0.0", "code": 1 } }</pre>

4.8.3 Eingebettete Codelisten

Einfache und stabile Codelisten können in dem Nachrichtenschema eingebettet werden. Beispiel:

Eingebettete Codeliste

```
{
  "type": "object",
  "required": [
    "number"
  ],
  "properties": {
    "number": {
      "type": "object",
      "oneOf": [
        {
          "const": 1,
          "title": "Eins",
          "description": "Dies ist die Zahl Eins.",
          "appinfo": {"shortname": "1", "longname": "eins", "englishName": "one"}
        },
        {
          "const": 2,
          "title": "Zwei",
          "description": "Dies ist die Zahl Zwei.",
          "appinfo": {"shortname": "2", "longname": "zwei", "englishName": "two"}
        }
      ]
    }
  }
}
```

Vorteile eingebetteter Codelisten:

- kein zusätzliches Codelistenschema notwendig

Nachteile eingebetteter Codelisten:

- bei Änderungen der Codeliste muss das Nachrichtenschema angepasst werden
- bei Verwendung der Codeliste durch unterschiedliche Nachrichtenschemata wird die Codeliste kopiert und muss mehrmals gepflegt werden

4.8.4 Handhabung von komplexeren Codelisten

Die Elemente komplexerer Codelisten sind keine einfachen Strings, sondern Objekte. Ein Beispiel für eine komplexere Codeliste ist eine Liste von Organisationsstandorten. Ein Standort wird dabei mit Attributen beschrieben, wie Adresse (Stadt, PLZ, Straße, Hausnummer), Telefonnummer, etc. Dabei werden Änderungen an den Attributen eines Datenelementes separat von Änderungen der entsprechenden Codeliste (Hinzufügen oder Löschen eines Elementes) betrachtet.

Bei komplexeren Codelisten kommen folgende Umsetzungsvarianten infrage:

4.8.4.1 Beschreibung von Objektattributen direkt in der Codeliste

Bei dieser Variante wird das Objekt mit allen dazugehörigen Objektattributen in dem Codelistenschema beschrieben. Siehe Beispiel oben.

Vorteil direkter Beschreibung von Objektattributen direkt in der Codeliste:

- keine zusätzlichen Abstraktionen, außer dem Codelistenschema, notwendig

Nachteile direkter Beschreibung von Objektattributen direkt in der Codeliste:

- Bei Änderungen von Objektattributen entstehen neue Schemaversionen - wird besonders schlimm bei größeren Listen, wie z.B. Liste der Standortadressen

- Geeignet eher für Anwendungsfälle, bei denen eine Historisierung der Eigenschaften wichtig ist. Bei LDB geht es eher um aktuelle Informationen
- die Umsetzungsvariante ist also weniger geeignet.

4.8.4.2 Referenzieren von Objekten in der Codeliste

Bei dieser Variante wird das Objekt in der Codeliste nur referenziert. Das Verwalten des Objektes (CRUD) wird über einen Service abgewickelt.

Vorteil bei der Verwaltung von Objektattributen über einen Service:

- Objektattribute können angepasst werden, ohne dass neue Codelistenversionen entstehen

Nachteil bei der Verwaltung von Objektattributen über einen Service:

- zusätzliche Abstraktionen, außer Codelistenschema, notwendig

4.8.4.3 Direktes Referenzieren von Objekten in dem Nachrichtenschema

Diese Variante verzichtet komplett auf eine Codeliste. Das Verwalten des Objektes (CRUD) wird komplett über eine separate VIDaL-Anwendung oder einen externen Service abgewickelt. Eine Meldung beinhaltet in diesem Fall nur noch die eigentliche Objektreferenz (Datenbank-ID). Diese Datenbank-ID wird bei der Abfrage der entsprechenden VIDaL-Anwendung bzw. des entsprechenden externen Service verwendet.

Vorteile des direkten Referenzierens von Objekten in dem Nachrichtenschema:

- eine Abstraktionsebene (Codeliste) verschwindet
- Rechte für die CRUD-Operationen lassen sich unabhängig vom Schema-Service definieren
- Informationen in komplexeren Listen sind typischerweise Landesspezifisch, somit werden sie nicht über das VIDaL-Gremium festgelegt und können über länderspezifische Services verwaltet werden

Nachteil des direkten Referenzierens von Objekten in dem Nachrichtenschema:

- Daten müssen über eine weitere VIDaL-Anwendung bzw. einen weiteren Service verwaltet werden

4.8.5 Verwaltung von Codelisten durch den Schema-Service

- Einfache Codelisten werden durch den Schema-Service als eigenständiges Schema verwaltet:

- Einsatzstichwortkatalog: Einsatzstichwortliste auf Basis Hessen, siehe „VIDaL AP1 Anl2 Einsatzstichworteerlass Hessen“
 - Ressourcenkatalog: siehe „VIDaL AP1 Anl3 Ressourcenkatalog IG-NRW“
 - Liste der Einsatztypen: auf Basis Stichwortliste Hessen und Landeskonzpte NRW, siehe „VIDaL AP1 Anl7 Inhalte Kontinuierliche Einsatzstatistik“
- Im Falle von sehr stabilen und kleinen Listen werden diese direkt im jeweiligen Nachrichtenschema eingebettet.
 - Listen von komplexeren Datenstrukturen wie Organisations-Standortstammdaten werden über einen Service verwaltet und das Referenzieren von einzelnen Objekten wird durch eindeutige Codes direkt in der Meldung implementiert.

5 Spezifikation Lagemodul

5.1 Einleitung

Das Lagemodul (LM) stellt eine Schnittstellenkomponente für den standardisierten Datenaustausch zwischen den Kommunikationsteilnehmern (KT) dar.

Das Lagemodul stellt die LM VIDaL API bereit. Sie ist die einzige Kommunikationsschnittstelle für verbundene Kommunikationsteilnehmer im VIDaL Projekt.

Die VIDaL API erfüllt die an das Lagemodul gestellten Anforderungen 2, 3, 4.

Bausteinsicht

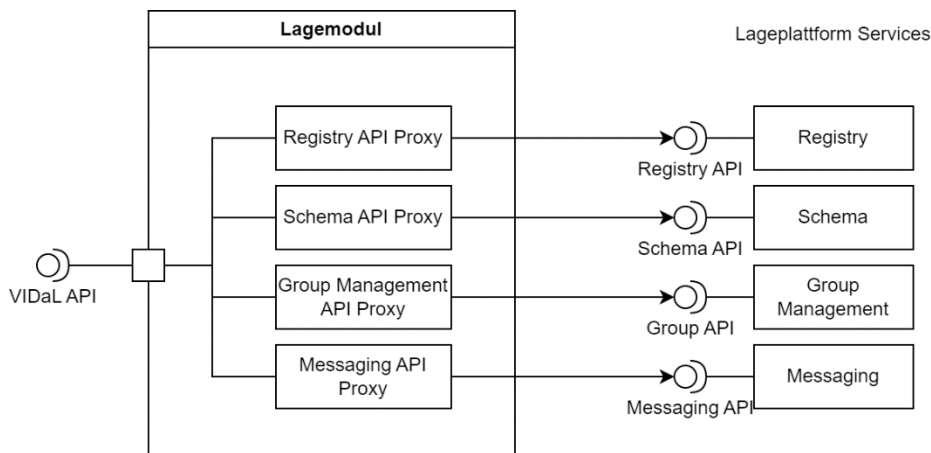


Abbildung 5.1: Bausteinsicht Lagemodul

5.2 LM VIDaL API v1.0

Das folgende Dokument enthält die Spezifikation der REST-API-Schnittstelle VIDaL zur Anbindung von Kommunikationsteilnehmern (KT) wie zum Beispiel Lagemanagementsystemen, Leitstellensystemen, anderen technischen Knoten, sowie weiteren externen Systemen an die Kommunikationsplattform VIDaL. Die Spezifikation umfasst die Datenmodelldefinition sowie alle verfügbaren Operationen. Siehe auch entsprechende Architekturentscheidungen.

Das VIDaL API Design verwendet die REST API Design-Richtlinien, erarbeitet durch das TM Forum sowie die Standards erarbeitet durch die OpenAPI Initiative.

5.2.1 Spezifikation

VIDaL API ist in folgende fachliche Bereiche aufgeteilt:

- Registry - Abfragen von Eigenschaften der registrierten Kommunikationsteilnehmer
- Nachrichtenschema - Abfragen von registrierten Nachrichtenschemata
- Group Management - Verwalten von Kommunikationsgruppen
- Messaging - Versenden und Empfangen von Nachrichten

5.2.2 Berechtigungskonzept

Grundsätzlich sind alle KT gleich berechtigt, verschiedene Funktionen der VIDaL API zu nutzen. Diese regulären Berechtigungen werden durch die Rollen/Rechte Konzepte von entsprechenden LPF-Services festgelegt.

Einem KT können auch Sonder-Rollen durch den LPF-Administrator zugewiesen werden:

- Gruppenadministrator – der KT bekommt dabei vollen Zugang zu allen Gruppen und kann jede Gruppe anlegen, modifizieren und löschen (es ist somit unter anderem möglich, ein Gruppenbesitz auch "ohne Einwilligung des Erstellers der Gruppe" an den KT einer höheren Behörde zu übergeben)

5.2.3 OpenAPI

Siehe `vidal_API.zip` im Lieferumfang.

5.2.4 VIDaL Registry API

Die VIDaL Registry API ist ein API-Proxy, der die LPF Registry Service API für Kommunikationsteilnehmer (KT) eingeschränkt veröffentlicht. Ein KT kann somit Eigenschaften (mit begrenzter Informationstiefe) von allen anderen KT abfragen.

Dabei werden nur folgende Endpoints für die Rolle KT (lesender Zugriff) veröffentlicht:

Endpoint	Operation	Beschreibung
registry	GET	<p>Liefert eine Liste aller KT mit einer für alle KT berechtigten Informationstiefe. Beispiel Response:</p> <pre>{ "clients" = [{ "oid" = "1.2.3.4.5.1", "systemName" = "LMS Essen", "operatorName" = "Krisenstab Essen", "operatorShortName" = "KS E", "supportedMessages" = [{ "msgType" = "bla", "versions" = ["1.0", "1.1"] }, { "msgType" = "foo", "versions" = ["1.0", "1.2"] }] }] },</pre>

Endpoint	Operation	Beschreibung
		<pre> { "oid" = "1.2.3.4.5.2", "systemName" = "ELS Essen", "operatorName" = "Leitstelle Essen", "operatorShortName" = "LSt E", "supportedMessages" = [{ "msgType" = "bla", "versions" = ["1.0", "1.1"] }, { "msgType" = "foo", "versions" = ["1.0", "1.2"] }] } </pre>
registry/{oid}	GET	Liefert KT-Beschreibung mit einer für alle KT berechtigten Informationstiefe.

Tabelle 5.1: VIDaL Registry API für die Rolle KT

5.2.5 VIDaL Schema API

Die VIDaL Schema API ist ein API Proxy, der die LPF Schema Service API für Kommunikationsteilnehmer (KT) eingeschränkt veröffentlicht. Über die VIDaL Schema API werden die im System registrierten Schemata für alle Nachrichtentypen in ihren Versionen zur Verfügung gestellt. Dabei werden nur folgende Endpoints für die Rolle KT (lesender Zugriff) veröffentlicht:

Endpoint	Operation	Beschreibung
schema	GET	Liefert eine Liste aller Meldungstypen. Beispiel Response: <pre>{ "msgTypes": ["bla", "foo"] }</pre>
schema/{msgType}	GET	Liefert eine Liste aller Versionen zu einem Meldungstyp. Beispiel Response: <pre>{ "msgType": "bla", "versions": ["1.0", "1.1"] }</pre>
schema/{msgType}/{version}	GET	Liefert ein Schema.

Tabelle 5.2: ViDaL-Schema API

5.2.6 ViDaL Group Management API

Die ViDaL Group Management API ist ein API-Proxy, der API des LPF Group Service für Kommunikationsteilnehmer (KT), siehe entsprechendes Berechtigungskonzept, mit der Gruppen-Rolle, die KT besitzt, veröffentlicht:

Endpoint	Operation	Beschreibung
group	GET	Liefert eine Liste aller Gruppen. Beispiel Response: <pre>{ "groups": [{ "id": "1.2.3.2.2023.42", "owner": "1.2.3.1.276.5.1.58.28.2.1", "description": "Krisengruppe X", "destinations": ["1.2.3.1.276.5.1.58.28.1.1", "1.2.3.1.276.5.1.58.28.1.2"] }] }</pre>
group	POST	Erzeugt / fügt hinzu eine neue Gruppe. Beispiel Request: <pre>{ "description": "Krisengruppe X",</pre>

Endpoint	Operation	Beschreibung
		<pre>"destinations": ["1.2.3.1.276.5.1.58.28.1.1", "1.2.3.1.276.5.1.58.28.1.2"] }</pre>
group/{id}	GET	Informationen zu einer Gruppe abfragen. Beispiel Response: <pre>{ "id": "1.2.3.2.2023.42", "owner": "1.2.3.1.276.5.1.58.28.2.1", "description": "Krisengruppe X", "destinations": ["1.2.3.1.276.5.1.58.28.1.1", "1.2.3.1.276.5.1.58.28.1.2"] }</pre>
group/{id}	PUT	Informationen zu einer Gruppe aktualisieren, z.B. neue KT in die "destinations" aufnehmen, KT aus der Liste nehmen oder Gruppenbesitzer wechseln.
group/{id}	DELETE	Gruppe löschen

Tabelle 5.3: VIDaL Group Management API

5.2.7 VIDaL Messaging API

Die VIDaL Messaging API ist ein API Proxy, der die LPF Messaging Service API für Kommunikationsteilnehmer (KT) veröffentlicht. Ein KT kann dadurch Nachrichten an andere KT senden und Nachrichten von anderen KT empfangen.

Bevor eine Nachricht an die LPF übermittelt wird, werden folgende Validierungen durchgeführt:

- Prüfung, ob der Sender berechtigt ist, die Nachricht zu senden (Vergleich Sender-OID mit Client-Identität)
- Payload-Validierung gegen vorgegebene Nachrichtenschemata

Falls konfiguriert, wird der Nachrichten-Payload vor der Übermittlung an die LPF verschlüsselt.

Beim Empfang einer verschlüsselten Nachricht, wird der Nachrichten-Payload vor der Auslieferung an den KT entschlüsselt.

Endpoint	Operation	Beschreibung
messaging/send	POST	Eine Meldung senden. Beispiel Request: <pre>{ "messageld": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454",</pre>

Endpoint	Operation	Beschreibung
		<pre> "description": "Lageinformation am 11.11.2022 um 11:11", "source": "1.2.3.4.5.6", "destination": ["1.2.3.3"], "payload": { "messageType": "situationReport", "messageVersion": "0.2", ... } } </pre>
messaging/receive	POST	<p>Meldungen empfangen. Beispiel Request:</p> <pre> { "destination": "1.2.3.4.5.6", "maxMessages": 5 } </pre> <p>Beispiel Response:</p> <pre> { "messages": [{ "messageId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454", "sequenceId": 123456, "description": "Lageinformation am 11.11.2022 um 11:11", "sentDate": "2023-03-25T21:56:47.483Z", "timeout": 300, "source": "1.2.3.4.5.6", "destination": ["1.2.3.4.5.7"], "payload": { "messageType": "situationReport", "messageVersion": "0.2", ... } }] } </pre> <p>"maxMessages": 5</p>

Endpoint	Operation	Beschreibung
		}
messaging/com- mit	POST	Empfang von Meldungen bis einschließlich angegebene Meldung bestätigen. Beispiel Request: { "destination": "1.2.3.4.5.6", "sequenceId": 123456 }

Tabelle 5.4: ViDaL Messaging API

Ein KT ist selbst dafür verantwortlich, periodisch in einem angemessenen Zyklus, die Informationen seiner Kommunikationspartner über unterstützte Nachrichtentypen (inkl. Versionen) über die ViDaL Registry API abzurufen und im Nachrichtenversand an die beteiligten Partner zu berücksichtigen. Dabei soll die Prämisse der höchsten von allen KT (Empfänger-OID) gemeinsam unterstützten Version gelten. Sollte es keine einzige Version des jeweiligen Nachrichtentyps geben, die von allen Empfänger-KT unterstützt wird, so muss der sendende KT die Nachricht ggf. mehrmals in der jeweils passenden Version für die einzelnen Empfänger versenden.

5.2.7.1 Quittungen

Aufgrund der Anforderungen im Fachlichen Feinkonzept in Bezug auf Zustellgarantien, wird eine Reihe technischer Quittungen für eine gesendete Nachricht implementiert (positive sowie negative Zustellbestätigung u.a.).

Ein Nachrichtensender bekommt folgende Quittungen:

- Zustellbestätigung: nachdem der Empfänger die Entgegennahme einer Nachricht bestätigt hat
- Benachrichtigung über fehlgeschlagene Zustellung: mit Hinweisen zu Fehlerursache:
 - KT vorübergehend abgemeldet: das KT-System des Empfängers ist vorübergehend abgemeldet, z.B. für eine Wartung
 - Timeout: nachdem ein für die Nachrichtenzustellung vordefiniertes Timeout verstrichen ist, ohne dass der Empfänger die Entgegennahme einer Nachricht bestätigt hat

Die technischen Quittungen sind in Form eines JSON-Schemas auf der untergeordneten Seite beschrieben.

5.2.7.2 Schema – Zustellquittung

5.2.7.2.1 Schema-Bezeichner

messageReceipt

5.2.7.2.2 Schema

JSON-Pro- perty	JSON-Datentyp	Datentyp-Einschränkung	Pflicht	Beschreibung
/ref-MessagId	string(uuid)	UUID	Ja	VIDaL-systemweit eindeutige ID der referenzierten Nachricht.
/receiverOID	string	OID	Ja	Empfänger-OID, OID des Empfängers der referenzierten Nachricht für den diese Quittung relevant ist
/statusCode	enum[integer]	[200, 503, 404, 504]	Ja	Status-Code: <ul style="list-style-type: none">200 - Zustellbestätigung, bei Gruppenadressierung gibt es für jeden Empfänger eine 200 Quittung404 - Empfänger der referenzierten Nachricht nicht gefunden. Die Benachrichtigung bezieht sich auf die Gruppen-OID, falls die angegebene Gruppe nicht gefunden wird, oder auf die KT-Einzeladresse, falls KT mit der angegebenen OID nicht gefunden wird.503 - KT - Empfänger der referenzierten Nachricht - vorübergehend abgemeldet504 - Timeout
/statusMessage	string	maxLength = 100	Nein	Text mit weiteren Einzelheiten und Abhilfemaßnahmen in Bezug auf den Fehler. Dies kann einem Client-Benutzer angezeigt werden.

Tabelle 5.5: Schema Zustellquittung

5.2.7.2.3 Beispiel

Negative Zustell-Quittung
<pre>{ "refMessageId": "f8c3de3d-1fea-4d7c-a8b0-29f63c4c3454", "receiverOID": "1.2.3.4.5.6.7", "statusCode": 503, "statusMessage": "Wartung bis voraussichtlich 27.09.2023 05:00" }</pre>

5.3 LM im Netzwerk des KT

Zum Betrieb des Lagemoduls an der Lokation des KT muss das vorkonfigurierte Lagemodul in das Netzwerk der Lokation integriert werden.

Das Lagemodul dient bei dem KT als Verbindungspunkt zur Lageplattform.

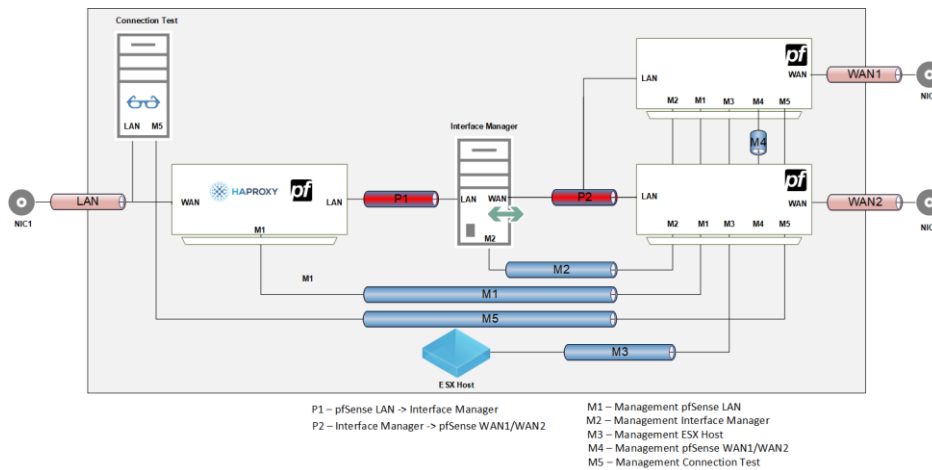


Abbildung 5.2: LM im Netzwerk des KT

Alle Komponenten des Lagemoduls werden virtualisiert auf einem physischen Server mit der Virtualisierungssoftware VMWare ESXi betrieben. Die internen Netzwerke zur Verbindung der Komponenten und zum Systemmanagement sind ebenfalls virtualisiert.

1. Firewall LAN: pfSense Firewall mit HAProxy als Reverse Proxy terminiert den VIDAL-HTTPS-Traffic von den Systemen am Standort (ELS/LMS), überprüft das Client-Zertifikat der Systeme und reicht die Anfrage an den Interfacemanager weiter.
2. Interface Manager: Nimmt den VIDAL-Traffic der ELS/LMS-Systeme entgegen und leitet diesen zur Lageplattform weiter. Falls die primäre Lageplattform nicht erreichbar ist, erfolgt ein Failover auf die zweite Lageplattform.
3. Firewalls WAN1: pfSense Firewall mit zwei IPSec-Tunneln zur primären Lageplattform. Ein Tunnel für VIDAL-Applikations-Traffic und ein zweiter zur Anbindung des Systemmanagements.
4. Firewalls WAN2: pfSense Firewall mit zwei IPSec-Tunneln zur sekundären Lageplattform. Ein Tunnel für VIDAL-Applikations-Traffic und ein zweiter zur Anbindung des Systemmanagements.
5. Connection Test: VM zur Erzeugung von VIDAL-Test-Traffic, um bei Verbindungsproblemen die Fehlersuche zu erleichtern.

Einsatzleit- und Lagemanagementsysteme verbinden zur LAN-Seite des Lademoduls über das ViDaL-Protokoll via HTTPS auf Port 8190 TCP.

Diese authentifizieren sich dort jeweils mit einem Clientzertifikat (mit ViDaL-OID), welches vorab von der ViDaL PKI ausgestellt wurde.

Das Lagemodul stellt redundant über zwei WAN-Interfaces die permanente Verbindung zu den Lageplattformen in den Rechenzentren über jeweils zwei IPSec-Tunnel her.

Das Lagemodul weist sich ebenfalls mit einem Client-Zertifikat gegenüber den Lageplattformen aus.

5.3.1 Einbau der Lagemodul Appliance

Das Lagemodul wird als 19" Server mit 1 Höheneinheit (HE) und redundanter Stromversorgung in einem vorhanden Technikraum installiert.

Die Stromversorgung sollte über zwei getrennte Stromphasen und idealerweise über mindestens eine USV bereitgestellt werden.

Der Technik- oder Serverraum muss klimatisiert sein.

Die drei Netzwerkinterfaces werden über drei 1 Gbit-Base-T Ports an den vorhandenen Switches am Standort angeschlossen. Direkte oder indirekte Verbindungen zwischen den Netzwerkinterfaces sind nicht gestattet, mit Ausnahme der hier dargestellten Verbindung über das Lagemodul.

5.3.2 Netzwerkintegration

Für das LAN-Interface werden zwei IP-Adressen benötigt. Eine IP-Adresse für das ViDaL-Lagemodul und eine IP-Adresse für den integrierten Connection Test. Beide IP-Adressen müssen sich im gleichen Subnetz befinden.

Die beiden WAN-Interfaces benötigen jeweils eine statische Public-IP im Internet und jeweils eine IP im Netzwerk des Standorts, falls das Internet per NAT/Routing angebunden wird.

Die Anbindung des Internets an die LM Appliance erfolgt entweder

- nativ auf dem WAN-Interface der LM Appliance
- geroutet an das WAN-Interface der LM Appliance in einem Kundentransfernetz
- 1:1 NAT auf das WAN-Interface der LM Appliance in einem Kundentransfernetz

Die beiden IPs können sich in getrennten Subnetzen befinden.

5.3.3 Informationen und Beistellungen des KT

Der Systembetreiber am Standort muss zur Vorkonfiguration des Lagemoduls folgende Daten zur Netzwerkintegration bereitstellen:

- alle IP-Adressen sind IPv4
- zwei LAN IPs mit IP-Adresse, Subnetz und Gateway-Adresse im gleichen Subnetz
- zwei Public-IPs für das WAN mit IP-Adresse, Subnetz und Gateway-Adresse
- optional für NAT/Routing zwei interne IPs mit IP-Adresse, Subnetz und Gateway-Adresse
- Zugang zu mindestens einem NTP-Server im Netz der Lokation mit IP-Adresse

Der Systembetreiber muss an seinem Standort folgende Infrastruktur zur Verfügung stellen

- 1 HE für einen 19" Server in einem klimatisierten Technik- oder Serverraum
- redundante Stromversorgung und USV-Anbindung
- drei Ports auf den vorhandenen Switchen mit 100/1GBase-T Port
- einen redundanten Internetzugang mit zwei Public-IPs oder zwei getrennte WAN-Router mit getrenntem Internetzugang und jeweils einer statischen Public-IP
- die benötigten IP-Adressen aus dem internen Netz
- zwei Public-IPs für die IPSec-Verbindung zur Lageplattform
- Freischaltungen auf der lokalen Firewall für HTTPS auf Port 8190 TCP für ViDaL, NTP auf der LAN-Seite und für IPSec auf der WAN-Seite

5.3.4 Integration in das vorhandene Standortnetzwerk

Es wird grundsätzlich empfohlen, das Lagemodul in einer (eigenen) DMZ zu betreiben.

Bei einer am Standort vorhandenen Internet-DMZ mit redundanter Internetanbindung wird das Lagemodul in dieser DMZ platziert:

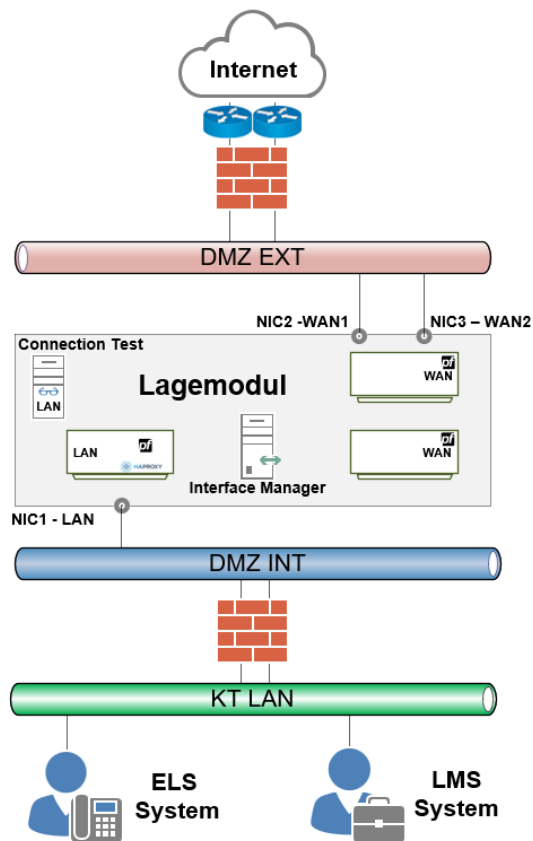


Abbildung 5.3: Integration in das vorhandene Standortnetzwerk

Das Lagemodul ist durch die interne DMZ-Firewall vom LAN des Standorts getrennt. Die beiden WAN-Anschlüsse werden an die externe DMZ-Firewall angebunden. Der Zugang zum Internet ist am Standort redundant.

Der Systemadministrator vor Ort muss die entsprechenden Firewall-Regeln in seinen Systemen aufnehmen.

Falls der Internetzugang nicht redundant ausgelegt ist, sollte der zweite WAN-Anschluss dediziert über einen zusätzlichen Router und einen zweiten Internetzugang angebunden werden.

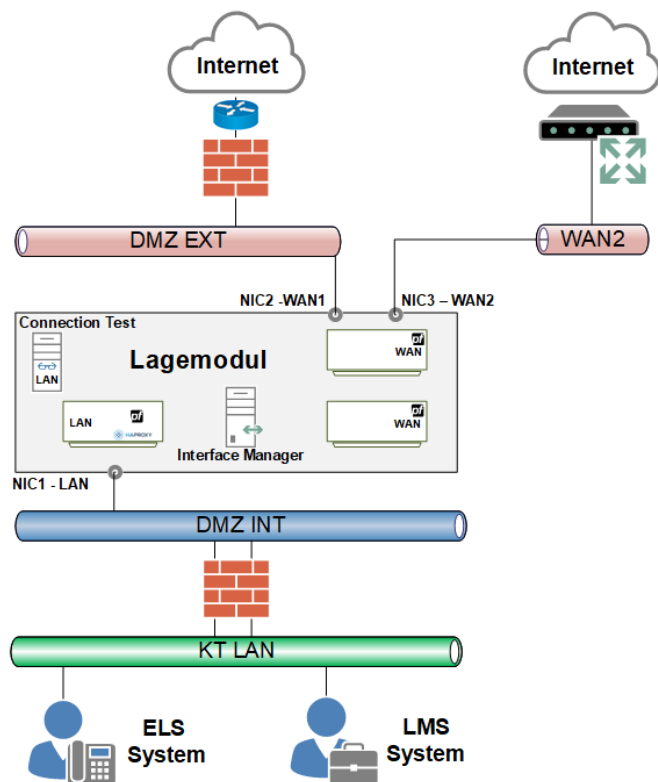


Abbildung 5.4: Verwendung eines dedizierten WAN-Anschlusses

Falls keine DMZ-Infrastruktur am Standort vorhanden ist, wird das Lagemodul direkt in das LAN des Standorts und die beiden WAN-Anschlüsse werden über getrennte Router und getrennte Zugänge mit dem Internet redundant angebunden.

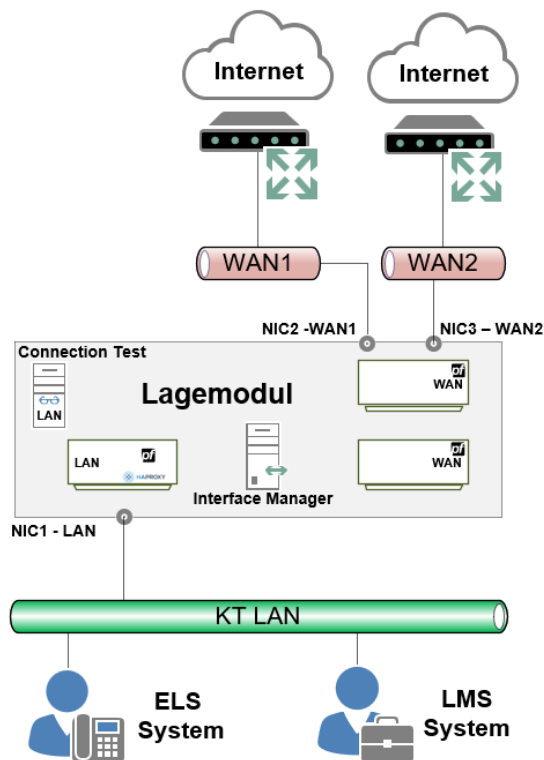


Abbildung 5.5: Anordnung ohne DMZ-Infrastruktur

Der redundante WAN-Anschluss erfolgt über getrennte Infrastruktur über zwei Internetzugänge.

5.4 LM Connection Test

Die Lagemodul Appliance beinhaltet eine virtualisierte Software Komponente Connection Test. Der Connection Test dient zur Erzeugung von VIDaL-Test-Traffic, um bei Verbindungsproblemen die Fehlersuche zu erleichtern.

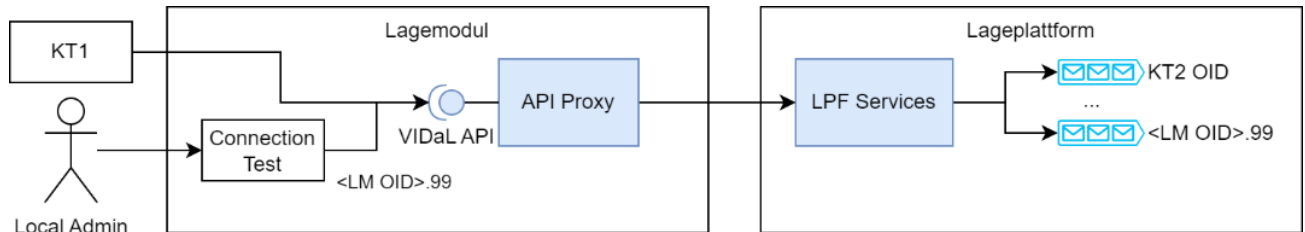


Abbildung 5.6: LM Connection Test

Der Connection Test ist aus Sicht der LPF ein regulärer KT (hat eine spezifische OID) und bietet folgende Funktionen:

- Status VIDaL Services
- E2E Kommunikationstest

5.4.1 Status der VIDaL Services

VIDaL Service	Status	Zusätzliche Informationen
Messaging	:tick:/:cross:	Mittlere Roundtrip-Zeit beim Senden einer Nachricht an sich selbst über die Lageplattform.
KT-Registry	:tick:/:cross:	
Schema Service	:tick:/:cross:	
Gruppen Service	:tick:/:cross:	

Tabelle 5.6: Status der VIDaL Services

5.4.2 E2E Kommunikationstest

Test	Beschreibung
KT-Anschlusstest	<p>Eine Testnachricht wird über die LPF an einen bestimmten KT gesendet. Der adressierte KT kann der Connection Test selbst oder ein KT am eigenen LM oder ein anderer KT sein.</p> <p>Die Testnachricht durchläuft den regulären Nachrichtenweg, wird aber vor Auslieferung verworfen.</p> <p>Die Lageplattform generiert entsprechende Quittungen, die dem Connection Test als dem sendenden KT zugehen. Der adressierte KT erhält die Testnachricht jedoch nicht.</p> <p>Außerdem prüft die LPF, ob der empfangende KT registriert und aktiv ist.</p> <p>Mögliche Ergebnisse:</p> <ul style="list-style-type: none">▪ Der Empfänger ist auf der Plattform registriert und aktiv und kann Nachrichten empfangen.▪ Der Empfänger ruft seine Meldungen nicht ab (vorübergehend abgemeldet).▪ Der Empfänger ist unbekannt.▪ Die Testnachricht hat die LPF nicht erreicht.

Tabelle 5.7: E2E Kommunikationstest

5.4.3 Kommunikation des lokalen Admins mit dem Connection Test

Der lokale Admin kommuniziert mit dem Connection Test über HTTPS (eine kleine WebApp) und muss sich im Rahmen eines OpenID Connect Verfahren mit einem Username / Passwort ausweisen.

6 Spezifikation Lageplattform

6.1 LPF Einleitung

Die Lageplattform (LPF) bietet redundante Services zur Vernetzung von Lagemodulen und Endpunkten der Kommunikationsteilnehmer (KT). Die LPF selbst ist frei von Fachlichkeit. Sie realisiert nur den Datentransport und sichert die Ende-zu-Ende-Zustellung der Daten.

6.1.1 Bausteinsicht

Die LPF beinhaltet folgende funktionalen Bausteine:

6.1.1.1 Services

- Registry Service: Ein Service zum Verwalten von den an die Plattform angeschlossenen KT
- Schema Service: Ein Service zum Verwalten von Datenschemata für den Nachrichtenaustausch
- Group Management Service: Ein Service zum Verwalten von Kommunikationsgruppen
- Messaging Service: Nachrichtenvermittlungsservice
- Key Management Service: Ein Service zum Generieren (periodisch und nach Bedarf) von verteilten Keys zum Verschlüsseln von Nachrichten-Payloads

6.1.1.2 Admin Webanwendung

Die LPF Admin ist eine Webanwendung mit folgenden Funktionen:

- Administrieren von den an die Plattform angeschlossenen KT
- Pflege der Datenschemata für den Nachrichtenaustausch
- Verwalten von Kommunikationsgruppen
- Generieren von symmetrischen Schlüsseln zum Verschlüsseln von Nachrichten-Payloads

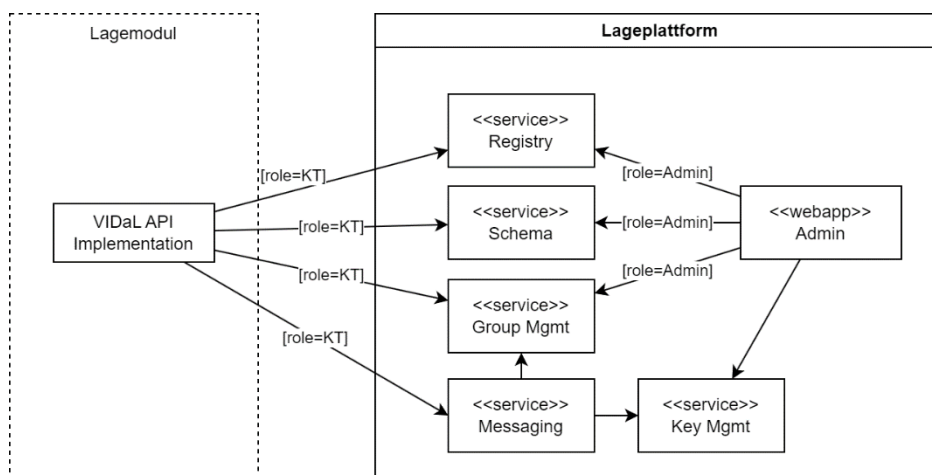


Abbildung 6.1: Admin Webanwendung

6.2 LPF KT Register

Zur Verwaltung der Kommunikationsteilnehmer (KT) wird ein Registry (Liste der Nutzer, Telefonbuch, Adressbuch, etc.) in der LPF angelegt.

Die KT sind Systeme (ELS, LMS, Stabsführungssystem, LDB, LDD, etc.) mit je einem Betreiber.
Zu dem Funktionsumfang der Registry gehören Funktionen wie KT (de)-registrieren, KT-Eigenschaften speichern, KT-Informationen abfragen oder ändern.

6.2.1 LPF Registry Service

Das KT-Register (Registry Service) beinhaltet Funktionen zum Verwalten von den an die Plattform angeschlossenen KT. Dazu gehören solche Funktionen wie KT (de)-registrieren, KT-Informationen abfragen oder ändern.

6.2.1.1 Registry Berechtigungskonzept

Zur Steuerung der Zugriffsberechtigung für Operationen im Zusammenhang mit der Pflege und Abfrage des KT-Registers werden zwei Rollen definiert:

- regulärer KT: Nur lesender Zugriff auf die Registry. Mit dieser Rolle können die KT-Eigenschaften (mit begrenzter Informationstiefe) von allen anderen KT abfragen.
- Registry-Administrator: Diese Rolle wird typischerweise einer natürlichen Person - einem User des Systemmanagements (Systemadministrator) gegeben. Somit hat der Registry-Administrator die Möglichkeit, KT zu administrieren (CRUD). Der Registry-Administrator hat dabei vollen Zugang zu allen KT und kann Kommunikationsteilnehmer (de)-registrieren und Eigenschaften einzelner KT modifizieren.

Beim Lesen von KT-Eigenschaften liefert die Registry verschiedene Informationsgehalte je nach Rolle des Aufrufers.

6.2.1.2 Registry API

Endpoint	Operation	Beschreibung	Rolle
registry	GET	Liefert eine Liste aller KT mit unterschiedlicher Informationstiefe je nach Berechtigung. Beispiel Response: <pre>{ "commParticipants": [{ "id": "1.2.3.4.5.1", "systemName": "LMS Essen", "operatorName": "Krisenstab Essen", "operatorShortName": "KS E", "supportedMessages": [{ "msgType": "bla", "msgVersion": "1.0" }, { "msgType": "bla", "msgVersion": "1.1" }, {</pre>	KT - geringer Informationsgehalt, Admin - höherer Informationsgehalt (inklusive technischen Ansprechpartner)

Endpoint	Operation	Beschreibung	Rolle
		<pre> "msgType": "foo", "msgVersion": "1.0" }], // Zusätzlich für Rolle Admin "techSupport": { "phone": "+49 69 123456", "email": "support@lst-essen.de", "address": "Musterstraße 1, 45133 Essen" } }, { "id": "1.2.3.4.5.2", "systemName": "ELS Essen", "operatorName": "Leitstelle Essen", "operatorShortName": "LSt E", "shortName": "LSt_2 E", "supportedMessages": [{ "msgType": "bla", "msgVersion": "1.0" }, { "msgType": "bla", "msgVersion": "1.1" }, { "msgType": "foo", "msgVersion": "1.0" }] // Zusätzlich für Rolle Admin "techSupport": { "phone": "+49 69 123456", "email": "support@lst-essen.de", "address": "Musterstraße 1, 45133 Essen" } }] </pre>	

Endpoint	Operation	Beschreibung	Rolle
		}	
registry	POST	<p>Erzeugt / fügt hinzu einen neuen KT (KT-Registrierung).</p> <pre>{ "id": "1.2.3.4.5.1", "systemName": "ELS Essen", "operatorName": "Leitstelle Essen", "operatorShortName": "LSt E", "shortName": "LSt_2 E", "supportedMessages": [...] ... // andere Infos je nach Berechtigung }</pre>	Admin
registry/{oid}	GET	Liefert KT-Beschreibung mit unterschiedlicher Informationstiefe je nach Berechtigung.	KT - geringerer Informationsgehalt, Admin - höherer Informationsgehalt (inklusive technischen Ansprechpartner)
registry/{oid}	PATCH	<p>Modifiziert einzelne Attribute eines KT. Somit können z.B. unterstützte Nachrichtenversionen angepasst werden. Das Element supportedMessages ist hierbei nicht als eigenständige Sub-Ressource zu sehen. Es soll eher einfach gehalten werden - nur das ganze Attribut darf geändert/überschrieben werden. Beispiel Request:</p> <pre>{ "supportedMessages": [{ "msgType": "bla", "msgVersion": "1.0" }, { "msgType": "bla", "msgVersion": "1.1" }, { "msgType": "foo", "msgVersion": "1.0" }] }</pre>	Admin

Endpoint	Operation	Beschreibung	Rolle
		<pre> }] } Zum Löschen aller unterstützten Versionen mit einer leeren Liste über- schreiben: { "supportedMessages": [] } </pre>	
registry/{oid}	DELETE	Löscht einen KT (Deregistrierung).	Admin

Tabelle 6.1: LPF Registry API

6.3 LPF Nachrichtenschema

In diesem Kapitel werden Bausteine beschrieben, die zur Pflege von Datenschemata für den Nachrichtenaustausch zwischen angeschlossenen KT dienen. Dazu gehören auch die sogenannten 4.8 Codelisten. Soll eine ViDaL-Meldung Werte einer Codeliste übertragen, so beinhaltet das entsprechende Nachrichtenschema einen Verweis auf das entsprechende Codelistenschematyp.

6.3.1 LPF Schema Service

Der Nachrichtenschema-Verwaltungsdienst (Schema Service) beinhaltet Funktionen zur Pflege von Datenschemata für den Nachrichtenaustausch zwischen angeschlossenen KT. Dazu gehören solche Funktionen wie ein Schema anlegen, abfragen oder als obsolet markieren. Diese Schemata werden zur Validierung der Nachrichten im LM genutzt.

Nachrichtenschemata sind Definitionen, die zwar "veralten" können, nicht aber verändert oder gelöscht werden können. Um ein Schema für neuere Entwicklungen auszuschließen, es wird es nach einer festzulegenden Frist als „inaktiv“ markiert. Damit kann es nicht mehr bei Sammelabfragen gelistet oder durch reguläre KT aufgelistet werden. Nachrichten, die mit diesem Schema verschickt werden sollen, können dann nicht mehr durch das LM validiert werden. Es MUSS eine Fehlermeldung an den KT gesendet werden. Lediglich der Schema-Service-Admin kann auf das inaktive Schema zugreifen.

6.3.1.1 Schema Service Berechtigungskonzept

Zur Steuerung der Zugriffsberechtigung für Operationen im Zusammenhang mit der Pflege und Abfragen der Schemata werden zwei Rollen definiert:

- regulärer KT: Lesender Zugriff auf Nachrichtenschemata, mit dieser Rolle kann ein KT im System registrierte Nachrichtenschemata abfragen.
- Schemata-Administrator: Diese Rolle wird typischerweise einer natürlichen Person - einem User des Systemmanagements (Systemadministrator) gegeben. Somit hat der Schemata-Administrator die Möglichkeit, Schemata zu administrieren (CRUD).

6.3.1.2 Schema API

Das Ressource Nachrichten-Datenschema ist in Sub-Ressourcen Nachrichtentyp (msgType) und Nachrichtentypversion (msgVersion) strukturiert, da es sonst zur Übermittlung von größeren Datenmengen kommen kann.

Endpoint	Operation	Beschreibung	Rolle
schema	GET	Liefert eine Liste aller aktiven Meldungstypen. Beispiel Response: <pre>{ "msgTypes" = ["bla", "foo"] }</pre>	KT, Admin
schema/{msgType}	GET	Liefert eine Liste aller aktiven Versionen zu einem Meldungstyp. Beispiel Response: <pre>{ "msgType" = "bla", "msgVersions" = ["1.0", "1.1"] }</pre>	KT, Admin
schema/{msgType}/{msg-Version}	GET	Download einer Schema-Datei. Schemata, die als inaktiv markiert wurden, können nur durch den Admin heruntergeladen werden.	KT, Admin
schema/{msgType}/{msg-Version}	PUT	Ein neues Schema zum Schema-Katalog hinzufügen. Beim Versuch ein existierendes Schema zu überschreiben, bekommt der Client den Fehler-Code 409 (Conflict) zurück.	Admin
schema/{msgType}/{msg-Version}	DELETE	Schema als inaktiv markieren. Ein als inaktiv markiertes Schema wird bei Sammelabfragen nicht mehr gelistet. Das inaktive Schema kann nur durch direkte Adressierung und nur durch Admin heruntergeladen werden.	Admin

Tabelle 6.2: LPF Schema API

6.3.1.3 Anwendungsfälle

6.3.1.3.1 Einführung eines neuen Schemas

Der Betreiber des ViDaL-Systems verabschiedet ein neues Nachrichtenschema. Daraufhin laufen die folgenden Schritte ab:

1. Alle betroffenen Systementwickler (Hersteller der Systemsoftware für ELS/LMS) werden (bspw. im Rahmen der Zusammenarbeit mit dem ViDaL Expertenforum) über das neue Schema informiert, gleichzeitig wird der Zeitplan zur Abkündigung der älteren Versionen bekanntgegeben. Dazu werden folgende Infos an die Systementwickler (bspw. über eine Funktionsmailbox) übermittelt:
 - a. ChangeLog und Name des neuen Schemas

- b. Datum bis wann das alte Schema noch benutzt werden kann. Dabei räumt der Systembetreiber ausreichend Vorbereitungszeit für anfallende Entwicklungs- und Installationsarbeiten ein.
2. Das neue Schema wird durch den Schemata-Administrator im VIDaL-System via Schema-Service-API der LPF registriert (PUT). Ab diesem Moment ist das neue Schema gleichzeitig mit älteren Schemata zugänglich und kann beim Nachrichtenaustausch verwendet werden.
3. Alle angeschlossenen KT werden auf der administrativen Ebene (Systemmeldung) über das neue Schema informiert.
4. Alle angeschlossenen KT werden auf der administrativen Ebene (Systemmeldung) über die Frist zum Löschen des alten Schemas informiert.
5. Nach Ablauf der gesetzten Fristen wird das alte Schema durch den Schemata-Administrator im VIDaL-System via Schema-Service-API der LPF als obsolet / inaktiv gesetzt. Somit wird das alte Schema für einen regulären KT nicht mehr sichtbar/erreichbar.
6. Alle angeschlossenen KT werden auf der administrativen Ebene (Systemmeldung) über das Löschen des alten Schemas informiert.

6.3.1.3.2 Versand einer Nachricht mit einem inaktiven Schema

Ein inaktives Schema wird genauso wie ein unbekanntes Schema bei Versand einer Nachricht mit einer Fehlermeldung beantwortet.

6.3.1.3.3 Systemmeldungen im Zusammenhang mit dem Schemamanagement

Im Zusammenhang mit dem Schemamanagement werden in der Lageplattform folgende Systemmeldungen automatisch generiert:

Systemmeldung	Empfänger	Beschreibung
Neues Schema	Alle angeschlossenen KT	Beim Registrieren eines neuen Nachrichtenschemas im VIDaL-System werden alle angeschlossenen KT benachrichtigt.
Schema EOL	Alle angeschlossenen KT	Beim Setzen einer EOL-Frist für ein Nachrichtenschema im VIDaL-System werden alle angeschlossenen KT benachrichtigt.
Schema gelöscht	Alle angeschlossenen KT	Beim Löschen eines Nachrichtenschemas im VIDaL-System werden alle angeschlossenen KT benachrichtigt.

Tabelle 6.3: Systemmeldungen im Zusammenhang mit dem Schemamanagement

6.4 LPF Gruppenmanagement

Für eine identische Informationslage auf allen Ebenen und in allen Stäben muss eine gemeinsame Kommunikationsgruppe gebildet werden können. Mit Hilfe einer Kommunikationsgruppe können mehreren KT gleichzeitig adressiert werden.

Hinter einer Gruppenadresse wird eine Liste von OIDs verwaltet. Darüber hinaus besitzt eine Gruppe folgende Attribute:

- Gruppen-OID: Siehe [OID-Gruppenadresse in Adressierungskonzept](#)
- Gruppenbesitzer: Die OID eines KT, der erweiterte Berechtigungen in Bezug auf diese Gruppe besitzt, wie bspw. Gruppe modifizieren oder löschen. Der Gruppenersteller wird automatisch zum Besitzer dieser Gruppe. Ein Gruppenbesitz kann grundsätzlich von einem KT auf einen

anderen übertragen werden. Damit wird auch das Recht, die Gruppe zu modifizieren oder zu löschen, übertragen (dies ist u.a. erforderlich, damit sich ein Ersteller einer Lage aus einer von ihm erstellten dynamischen Gruppe zurückziehen kann, ohne die Konsequenz diese Gruppe zu löschen, wenn die Lage grundsätzlich noch nicht beendet ist, jedoch nur noch andere KT betroffen sind. Ein weiteres Beispiel: Der Gruppenbesitz geht an eine höhere Hierarchieebene über – hierarchische Eskalation.). Der Gruppenbesitzer muss nicht zwangsläufig als ein Nachrichtenempfänger in dieser Gruppe aufgelistet sein!

Das Gruppenmanagement ist durch den [6.4.1 LPF Group Service](#) implementiert. Die Kommunikationsgruppen können über folgende Wege verwaltet werden:

- [LM ViDaL API v1.0](#)
- [6.7 LPF operatives Systemmanagement](#)

6.4.1 LPF Group Service

Der Group Service beinhaltet Funktionen zum Verwalten von Kommunikationsgruppen (CRUD).

6.4.1.1 Gruppenmanagement Berechtigungskonzept

Zur Steuerung der Zugriffsberechtigung für Operationen im Zusammenhang mit Kommunikationsgruppen werden zwei Rollen definiert:

- regulärer KT und
- Gruppenadministrator: Diese Rolle wird typischerweise einer natürlichen Person (einem User des Systemmanagements = Systemadministrator) gegeben. Somit hat der Gruppenadministrator die Möglichkeit, Gruppen zu administrieren (CRUD). Der Gruppenadministrator hat dabei vollen Zugang zu allen Gruppen und kann jede Gruppe anlegen, modifizieren und löschen (es ist somit unter anderem möglich, ein Gruppenbesitz auch "ohne Einwilligung des Erstellers der Gruppe" an den KT einer höheren Behörde zu übergeben). Die Rolle Gruppenadministrator kann auch einem technischen Client (KT) gegeben werden.

Operation	Kommunikationsteilnehmer	Gruppenad- ministrator
Gruppe anlegen	Ja	Ja
Gruppe auflisten, Gruppen- eigenschaften anschauen	Nur die Gruppen, in der der KT-Gruppenbesitzer ist oder in denen der KT drin ist. Gruppenbesitzer muss NICHT notwendigerweise selber Mitglied dieser Gruppe sein.	alle Gruppen
Gruppe modifizieren inklusive Gruppenbesitz übertragen	Nur die Gruppe, in der der KT-Gruppenbesitzer ist.	alle Gruppen
Gruppe löschen	Nur die Gruppe, in der der KT-Gruppenbesitzer ist.	alle Gruppen

Tabelle 6.4: Gruppenmanagement Berechtigungskonzept

6.4.1.2 Group API

Die Verwaltung von Kommunikationsgruppen lässt sich auf ein Entity-basiertes CRUD-Modell abbilden:

Endpoint	Operation	Beschreibung	Rolle
group	GET	<p>Liefert eine Liste aller Gruppen. Beispiel Response:</p> <pre>{ "groups": [{ "id": "1.2.3.2.2023.42", "owner": "1.2.3.1.276.5.1.58.28.2.1", "description": "Krisengruppe X", "destinations": ["1.2.3.1.276.5.1.58.28.1.1", "1.2.3.1.276.5.1.58.28.1.2"] }] }</pre>	KT, Admin, siehe auch Berechtigungskonzept oben
group	POST	<p>Erzeugt / fügt eine neue Gruppe hinzu. Der Gruppenersteller wird automatisch zum Besitzer dieser Gruppe. Der Gruppenbesitzer muss nicht zwangsläufig als ein Nachrichtenempfänger in dieser Gruppe aufgelistet sein. Die ID der Gruppe wird von der LPF erzeugt und den Beteiligten mit der Systemmeldung zum Anlegen der Gruppe (s. unten) an alle Beteiligten verteilt.</p> <p>Beispiel Request:</p> <pre>{ "description": "Krisengruppe X", "destinations": ["1.2.3.1.276.5.1.58.28.1.1", "1.2.3.1.276.5.1.58.28.1.2"] }</pre>	KT, Admin
group/{id}	GET	<p>Informationen zu einer Gruppe abfragen. Beispiel Response:</p> <pre>{ "id": "1.2.3.2.2023.42", "owner": "1.2.3.1.276.5.1.58.28.2.1", "description": "Krisengruppe X", "destinations": ["1.2.3.1.276.5.1.58.28.1.1", "1.2.3.1.276.5.1.58.28.1.2"] }</pre>	KT, Admin, siehe auch Berechtigungskonzept oben
group/{id}	PUT	<p>Informationen zu einer Gruppe aktualisieren, z.B. neue KT in die "destinations" aufnehmen, KT aus der Liste nehmen oder Gruppenbesitzer wechseln. Der Gruppenbesitzer muss nicht zwangsläufig als ein Nachrichtenempfänger in dieser Gruppe aufgelistet sein.</p> <p>Beispiel Request: Empfänger hinzufügen:</p> <pre>{ "owner": "1.2.3.1.276.5.1.58.28.2.1",</pre>	KT, Admin, siehe auch Berechtigungskonzept oben

Endpoint	Operation	Beschreibung	Rolle
		<pre> "description": "Krisengruppe X", "destinations": ["1.2.3.1.276.5.1.58.28.1.1", "1.2.3.1.276.5.1.58.28.1.2", "1.2.3.1.276.5.1.58.28.1.3"] } </pre>	
group/{id}	DELETE	Gruppe löschen	KT, Admin, siehe auch Berechtigungskonzept oben

Tabelle 6.5: Group API

6.4.1.3 Systemmeldungen im Zusammenhang mit dem Gruppenmanagement

Im Zusammenhang mit dem Gruppenmanagement werden in der Lageplattform folgende Systemmeldungen automatisch generiert:

Systemmeldung	Empfänger	Beschreibung
Neue Gruppe	Alle KT in der neuen Gruppe und der Besitzer der Gruppe	Beim Anlegen einer Gruppe werden alle KT in der neuen Gruppe und der Besitzer der Gruppe informiert.
Neue Mitglied aufgenommen	Alle KT in der Gruppe und der Besitzer der Gruppe	Bei Änderungen in einer existierenden Gruppe, wenn neue Mitglieder aufgenommen wurden, werden alle KT in der Gruppe und der Besitzer der Gruppe über die Gruppe und alle ihre KT informiert.
Mitglied verlässt die Gruppe	Alle KT in der Gruppe und der Besitzer der Gruppe sowie gerade ausgeschiedene Mitglieder	Bei Änderungen in einer existierenden Gruppe, wenn Mitglieder die Gruppe verlassen, werden alle KT in der Gruppe und der Besitzer der Gruppe über die Gruppe und alle ihre KT informiert. Auch gerade ausgeschiedene Mitglieder bekommen diese Meldung.
Eigentümer der Gruppe wechselt	Alle KT in der Gruppe sowie der abgebende und der neue Gruppenbesitzer	Bei Besitzwechsel in der Gruppe werden alle KT in der Gruppe sowie der abgebende und der neue Gruppenbesitzer informiert.
Gruppe gelöscht	Alle KT in der gelöschten Gruppe und der Besitzer der Gruppe	Beim Löschen einer Gruppe werden alle KT in der Gruppe und der Besitzer der Gruppe informiert.

Tabelle 6.6: Systemmeldungen Gruppenmanagement

6.5 LPF-Messaging

Dieses Kapitel beschreibt die Bausteine der LPF, die der Nachrichtenvermittlung dienen.

6.5.1 Spezifikation Kontinuierliche Einsatzstatistik

Der Messaging Service beinhaltet Funktionen zum Versenden und Empfangen von Nachrichten, die im Rahmen des ViDaL Projekts spezifiziert sind.

6.5.1.1 Messaging Berechtigungskonzept

Alle Clients (Kommunikationsteilnehmer, Administratoren) dürfen grundsätzlich Nachrichten senden und an sie adressierte Nachrichten empfangen, siehe auch Fachliches Feinkonzept - AAA, Anforderung 3.

6.5.1.2 Messaging API

Der Messaging Service stellt eine REST API bereit. Dabei kann das Versenden und Empfangen von Nachrichten nicht ohne weiteres auf das Entity-basiertes CRUD-Modell abgebildet werden. Diese Funktionalität verlangt eher Operation-basierte Interfaces. Es wird daher auf das Pattern "Modellieren von Operationen in REST" zurückgegriffen, siehe [REST API Design-Richtlinien](#). Es werden folgende Resources für entsprechende Operationen definiert:

- messaging/send: Nachricht senden
- messaging/receive: Nachrichten empfangen
- messaging/commit: empfangene Nachrichten bestätigen

Siehe auch Messaging in ViDaL-API.zip im Lieferumfang.

Empfangsbestätigungen bzw. Fehlermeldungen werden durch das System versendet, über den gleichen Mechanismus (LPF Messaging Service) wie alle anderen Meldungen vermittelt und durch den Sender der ursprünglichen Meldung über die ViDaL Messaging API empfangen.

Da die Empfangsbestätigungen und Fehlermeldungen aus der KT Sicht mit der ViDaL Messaging API zusammenhängen, werden sie unter [ViDaL Messaging API](#) eingeführt (Quittungen) und auf der Unterseite [Schema - Zustellquittung](#) beschrieben.

6.6 LPF Systemmeldungen

In bestimmten Anwendungsfällen muss die Lageplattform (oder ihr Administrator) einem oder mehreren KT Informationen zukommen lassen. Das erfolgt in Form von Systemmeldungen, die (im Regelfall) automatisch generiert und über den Messaging Service versendet werden. Die Lageplattform ist dabei in der Rolle des KT-Absenders der Systemmeldung, sie bekommt eine spezielle eigene OID zugewiesen.

Systemmeldungen können auch von KT-Systemen an das ViDaL System gerichtet werden. Siehe "Vorübergehende KT-Abmeldung" in der Tabelle unten.

Das Schema einer Systemmeldung ist unter [6.6.1 Schema - Systemmeldung](#) beschrieben.

Folgende Systemmeldungen sind definiert:

System-meldung (subject)	Empfänger	Mel-dungs-Code (code)	Meldungsattribute (attributes). Alle At-tribute sind Pflicht	Text (text)	Beschreibung
Schemamanagement					
Neues Schema	Alle angeschlossenen KT	Schema Created	<ul style="list-style-type: none"> ■ schemald (string): Schema-Bezeichner ■ schemaVersion (string): Schema-Version 	Erklären-des Text men-schenles-bar	Beim Registrieren eines neuen Nachrichtenschemas im ViDaL-System werden alle angeschlossenen KT benachrichtigt. Formatierung
Schema EOL	Alle angeschlossenen KT	Schema EOL	<ul style="list-style-type: none"> ■ schemald (string): Schema-Bezeichner ■ schemaVersion (string): Schema-Version ■ eol (date): Schema End Of Life 	Erklären-des Text men-schenles-bar	Beim Setzen einer EOL-Frist für ein Nachrichtenschema im ViDaL-System werden alle angeschlossenen KT benachrichtigt.
Schema gelöscht	Alle angeschlossenen KT	Schema Deleted	<ul style="list-style-type: none"> ■ schemald (string): Schema-Bezeichner ■ schemaVersion (string): Schema-Version 	Erklären-des Text men-schenles-bar	Beim Löschen eines Nachrichtenschemas im ViDaL-System werden alle angeschlossenen KT benachrichtigt.
Gruppenmanagement					
Neue Gruppe	Alle KT in der neuen Gruppe und der Besitzer der Gruppe	Group Created	<ul style="list-style-type: none"> ■ group (OID): Gruppen-ID ■ owner (OID): Gruppenbesitzer ■ destinations (array[OID]): OIDs einzelner Gruppenteilnehmer 	Erklären-des Text men-schenles-bar	Beim Anlegen einer Gruppe werden alle KT in der neuen Gruppe und der Besitzer der Gruppe informiert.

System-meldung (subject)	Empfänger	Mel-dungs-Code (code)	Meldungsattribute (attributes). Alle At-tribute sind Pflicht	Text (text)	Beschreibung
Neue Mit-glied auf-genom-men	Alle KT in der Gruppe und der Besitzer der Gruppe	Group Member Entered	<ul style="list-style-type: none"> group (OID): Gruppen-ID owner (OID): Gruppenbesitzer destinations (array[OID]): OIDs einzelner Gruppenteilnehmer entered (array[OID]): OIDs neuer Gruppenteilnehmer 	Erklären-des Text men-schenles-bar	Bei Änderungen in einer existierenden Gruppe, wenn neue Mitglieder aufgenommen wurden, werden alle KT in der Gruppe und der Besitzer der Gruppe über die Gruppe und alle ihre KT informiert.
Mitglied verlässt die Gruppe	Alle KT in der Gruppe plus gerade ausgeschiedene Mitglieder und der Besitzer der Gruppe	Group Member Left	<ul style="list-style-type: none"> group (OID): Gruppen-ID owner (OID): Gruppenbesitzer destinations (array[OID]): OIDs einzelner Gruppenteilnehmer left (array[OID]): OIDs ausgeschiedener Gruppenteilnehmer 	Erklären-des Text men-schenles-bar	Bei Änderungen in einer existierenden Gruppe, wenn Mitglieder die Gruppe verlassen, werden alle KT in der Gruppe und der Besitzer der Gruppe über die Gruppe und alle ihre KT informiert. Auch gerade ausgeschiedene Mitglieder bekommen diese Meldung.
Eigentümer der Gruppe wechselt	Alle KT in der Gruppe sowie der alte und der neue Besitzer der Gruppe	Group Owner Changed	<ul style="list-style-type: none"> group (OID): Gruppen-ID owner (OID): Gruppenbesitzer destinations (array[OID]): OIDs einzelner Gruppenteilnehmer 	Erklären-des Text men-schenles-bar	Bei Besitzwechsel in der Gruppe werden alle KT in der Gruppe sowie der alte und der neue Gruppenbesitzer informiert.

System-meldung (subject)	Empfänger	Mel-dungs-Code (code)	Meldungsattribute (attributes). Alle At-tribute sind Pflicht	Text (text)	Beschreibung
			<ul style="list-style-type: none"> previousOwner (OID): der ab-gebende Grup-penbesitzer 		
Gruppe gelöscht	Alle KT in der gelöschten Gruppe und der Besitzer der Gruppe	Group Deleted	<ul style="list-style-type: none"> group (OID): Gruppen-ID 	Erklären-des Text men-schenles-bar	Beim Löschen einer Gruppe wer-den alle KT in der Gruppe und der Besitzer der Gruppe informiert.
KT-Meldungen (von KT-Systemen an das ViDaL System)					
KT Status	ViDaL System	KT Status	<ul style="list-style-type: none"> kt (OID): KT OID status (enum: "on", "off") 	Erklären-des Text men-schenles-bar	<p>Eine geplante vorübergehende Ab-meldung eines KT, bzw. Zurückmel-dung.</p> <p>Die Meldung wird in der Lageplatt-form automatisch verarbeitet. Der Nachrichtenabsender bekommt eine negative Zustellquittung mit dem Code `503` (KT vorübergehend abgemeldet). Siehe 5.2.7.2 Schema - Zustellquittung.</p> <p>Die KT OID soll der Sender OID ent-sprechen. Sonst wird die Status-meldung schweigend ignoriert.</p>
Sonstige Meldungen					
Je nach Bedarf	Ausgewählte KT, je nach Auf-gabe	Free Text Message	Keine	Je nach Bedarf	Eine Meldung mit frei-definierba-rem Textinhalt. Die Freitext-Mel-dung wird typischerweise von Vi-DaL-Systemadministratoren ver-wendet, um die lokalen Administra-toren von KT-Systemen zu benach-richtigen.

Tabelle 6.7: LPF Systemmeldungen

6.6.1 Schema – Systemmeldung

Schema für die Darstellung von Systemmeldungen.

6.6.1.1 Schema-Bezeichner

systemMessage

6.6.1.2 Schema

JSON-Pro- perty	JSON-Datentyp	Datentyp- Einschrän- kung	Pflicht	Beschreibung
/sub- ject	string	maxLength: 2000	Ja	Die menschenlesbare Betreffzeile.
/code	enum[string], embedded Codeliste		Ja	Der maschinenlesbare eindeutige Code der Meldung. Im ViDaL Lageplattform-System werden Meldungs-Codes für alle Systemmeldungen festgelegt. Die Liste der Meldungs-Codes ist eine embedded Codeliste, die direkt im Systemmeldung-Schema enthalten ist.
/attri- butes	map[string, string]		Nein	Ein für jeden Meldungs-Code spezifisches Set von Attributen
/text	string	maxLength: 10000	Nein	Der menschenlesbare Text der Meldung.

Tabelle 6.8: Schema Systemmeldung

6.6.1.3 Beispiel

```
{
  "subject": "New Schema missionStatistics 0.2",
  "code": "Schema Created",
  "attributes": {
    "schemaId": "missionStatistics",
    "schemaVersion": "0.2"
  },
  "text": "Neues Schema missionStatistics_0.2 im Schema-Service bereit  
gestellt"
}
```

6.7 Operatives Systemmanagement

Die administrativen Aufgaben werden in zwei Ebenen unterteilt:

- Operatives Systemmanagement: Administrieren von fachlichen Aspekten des laufenden Systems auf der Anwendungsebene und Applikationsmonitoring sowie die
- Systemmanagement Infrastruktur: Administrieren von technischer Infrastruktur, technisches Systemmonitoring

In diesem Kapitel werden die Aufgaben des operativen Systemmanagements für die LPF beschrieben. Das Konzept des technischen Systemmanagements ist für alle ViDaL-Systemkomponenten einheitlich im Betriebskonzept dokumentiert.

6.7.1 Administrative ViDaL-Funktionen

Auf der LPF werden folgende ViDaL-spezifischen administrativen Funktionen umgesetzt:

- Administrieren von den an die Plattform angeschlossenen KT
 - Administrieren von Kontaktdaten
 - Festlegen von OIDs, siehe auch 2.1.3 Einrichten eines Client-SSL-Zertifikats
 - Administrieren von durch den KT unterstützten MessageTypes und deren Versionen
 - Administrieren von KT-Rollen (z.B. Rolle Gruppen-Admin zur Übertragung von Gruppenbesitz)
- Pflege der Datenschemata und Codelisten für den Nachrichtenaustausch
 - Ein neues Schema zum Schema-Katalog hinzufügen
 - Ein Schema als obsolet markieren
- Verwalten von Kommunikationsgruppen
 - Eine KT-Gruppe anlegen / löschen
 - Eine KT-Gruppe verändern (KT hinzufügen / entfernen, Besitz der Gruppe verändern)
- Generieren von symmetrischen Schlüsseln zum Verschlüsseln von Nachrichten-Payloads
 - manuelles Erzeugen von neuen symmetrischen Schlüsseln

Die ViDaL-spezifischen administrativen Funktionen auf der LPF werden über eine einheitliche Webanwendung bereitgestellt.

6.7.2 Authentifizierung / Autorisierung

Die LPF-Administratoren sind natürliche Personen und melden sich per 2FA an der administrativen Web APP GUI mit OpenID Connect an. (Zwei-Faktor-Authentisierung)

6.7.3 Systemmeldungen

Alle Systemmeldungen, die im Rahmen des operativen Systemmanagement generiert werden (dazu zählen sowohl automatische Meldungen, als auch manuelle Ad-hoc-Mitteilungen eines Systemadministrators) werden auf der Seite 6.6 LPF Systemmeldungen strukturell spezifiziert.

7 Spezifikation Lagedokumentationsdienst

7.1 Einleitung

Der Lagedokumentationsdienst (LDD) beinhaltet die Geschäftsprozesse zur revisionssicheren Archivierung aller Datensätze von Meldungen und Lageberichten sowie deren sequentieller Abfrage (Event Sourcing, Use Case Late Entry) im Kontext eines Großereignisses (Lage).

Siehe auch Anforderungen an den Lagedokumentationsdienst im Fachlichen Feinkonzept.

7.1.1 Bausteinsicht

Der LDD beinhaltet folgende funktionale Bausteine:

- **KT-Adapter:** Ein Service zum Empfangen von fachlichen Anfragen der ViDaL-Anwendung Meldungen und zum Senden von korrespondierenden Antworten mittels Polling der ViDaL-API des zugehörigen Lagemoduls
- **SituationReport:** Ein Service, der serverseitige Anteile der ViDaL-Anwendung Meldungen (siehe Spezifikation Meldungen) bereitstellt, inkl. der revisionssicheren Aufbewahrung eingegangener und verwalteter Meldungen auf einem WriteOnce Storage für die eingestellte Aufbewahrungsdauer.

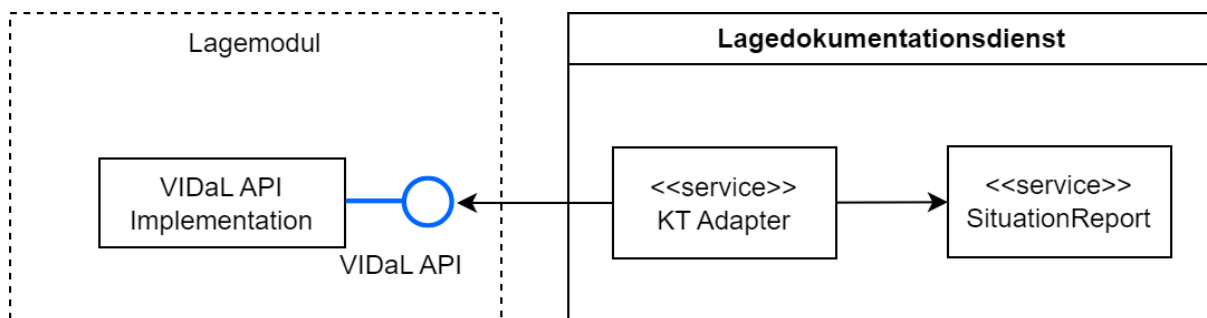


Abbildung 7.1: Bausteinsicht LDD

Die zentrale Komponente LDD in ihrer Gesamtheit beinhaltet somit ein eigenes, ihr zugeordnetes Lagemodul sowie die eigentliche Fachapplikation Lagedokumentationsdienst. Genau wie bei allen anderen KT, wird auch hier das Lagemodul von der Lageplattform aus verwaltet und zentral über das Systemmanagement selbiger aktualisiert. Die Fachapplikation LDD hingegen ist autark, basiert dabei ebenfalls auf dem Produkt Interface Manager, hat jedoch ihr eigenes Systemmanagement und auch ein eigenständiges Identity und Access Management (Rollen- und Rechteverwaltung / Benutzermanagement). Die Fachapplikation LDD ist somit betriebstechnisch/technologisch und administrativ unabhängig von der LPF.

7.2 LDD Berechtigungskonzept

Siehe Anforderung 2 an den Lagedokumentationsdienst im Fachlichen Feinkonzept.

Siehe auch Gruppenmanagement Berechtigungskonzept.

Zur Steuerung der Zugriffsberechtigung für Abfragen der archivierten Nachrichten werden zwei Rollen definiert:

- regulärer KT und
- LDD-Administrator: Eine spezifische Rolle in IAM des LDD. Diese Rolle wird typischerweise einer natürlichen Person, einem User des Systemmanagements (Systemadministrator) gegeben. Somit hat ein Systemadministrator mit der Rolle LDD-Administrator die Möglichkeit, archivierte Nachrichten abzufragen. Der LDD-Administrator hat dabei Zugang zu allen Nachrichten.

7.2.1 Zugriffsberechtigung für regulären KT

Der reguläre KT kann die archivierten Nachrichten über spezielle Abfragen nachrichten anfordern. Dabei gelten folgende Regeln.

Falls eine Nachricht an eine Kommunikationsgruppe gesendet wurde, dann gelten folgende Zugriffsberechtigungen für diese Nachricht in Bezug auf KT-Mitglieder dieser Gruppe und den Absender der Nachricht:

- angeforderte Nachrichten kommen immer asynchron über den Standard-Nachrichtenaustausch der LPF
- solange die Gruppe aktiv ist, dürfen alle KT, die Mitglieder dieser Gruppe sind, Nachrichten lesend abrufen
- ist ein KT nicht mehr Mitglied der Gruppe, darf er KEINE Nachrichten mehr abrufen (auch seine eigenen ehemals versandten Nachrichten nicht mehr)
- wird eine Gruppe gelöscht, darf KEIN KT mehr Nachrichten dieser Gruppe lesend abrufen. Nach dem Ende des Lebenszyklus einer Gruppe dürfen nur noch natürliche Personen aufgrund ihrer spezifischen Rolle, welche im IAM des LDD hinterlegt ist, Nachrichten bspw. zwecks Prüfung und Revision (z.B. juristische Aufarbeitung), lesend abrufen.

7.2.2 Zugriffsberechtigung für LDD-Administrator

Der LDD-Administrator kann die archivierten Nachrichten über das LDD-Systemmanagement abfragen.

Der LDD-Administrator hat lesenden Zugang zu allen archivierten Nachrichten des LDD.

8 Spezifikation Lagedatenbank

8.1 LDB Einleitung

Die Lagedatenbank (LDB) beinhaltet die Geschäftsprozesse zur Bereitstellung und Abfrage der aktuellen Daten zu den verfügbaren Ressourcen, landesweiten Einheiten (LKE, taktischen Fähigkeiten) sowie zur kontinuierlichen Einsatzstatistik.

Siehe auch Anforderungen an die 3.7 Lagedatenbank im Fachlichen Feinkonzept.

8.1.1 Bausteinsicht

Die LDB beinhaltet folgende funktionalen Bausteine (jeweils umgesetzt mittels Low Code Modellierung und einer fachlichen Persistenzschicht im Interface Manager):

- **KT-Adapter:** Ein Service zum Empfangen von fachlichen Anfragen der VIDaL-Anwendungen und zum Senden von korrespondierenden Antworten mittels VIDaL-API
- **UnitLKE:** Ein Service, der serverseitige Anteile der VIDaL-Anwendung Einheiten (siehe 4.3 Spezifikation Einheiten) bereitstellt
- **Resource:** Ein Service, der serverseitige Anteile der VIDaL-Anwendung Ressourcen (siehe 4.4 Spezifikation Ressourcen) bereitstellt
- **Statistics:** Ein Service, der serverseitige Anteile der VIDaL-Anwendung Kontinuierliche Statistik (siehe 4.5 Spezifikation Kontinuierliche Einsatzstatistik) bereitstellt
- **Organisation:** Ein Service, der serverseitige Anteile der VIDaL-Anwendung Organisationen (siehe 4.7 Spezifikation Organisationen) bereitstellt

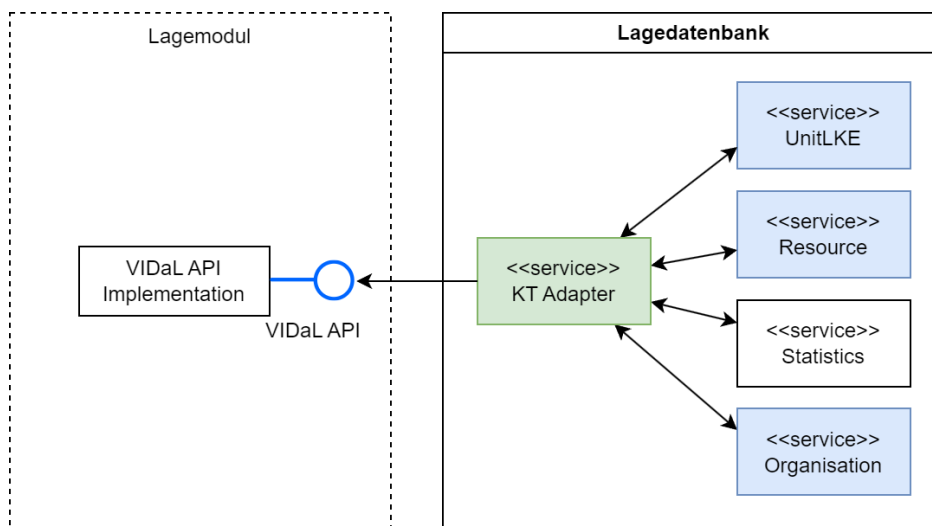


Abbildung 8.1: Bausteinsicht LDB

Die zentrale Komponente LDB in ihrer Gesamtheit beinhaltet somit ein eigenes, ihr zugeordnetes Lagemodul sowie die eigentliche Fachapplikation Lagedatenbank. Genau wie bei allen anderen KT wird auch hier das Lagemodul von der Lageplattform aus verwaltet und zentral über das Systemmanagement selbiger aktualisiert. Die Fachapplikation LDB hingegen ist autark, basiert dabei ebenfalls auf dem Produkt Interface Manager, hat jedoch ihr eigenes Systemmanagement und auch ein eigenständiges Identity- und Access-Management (Rollen- und Rechteverwaltung / Benutzermanagement).

Die Fachapplikation LDB ist somit betriebstechnisch/technologisch und administrativ unabhängig von der LPF.

8.2 LDB Datenmodell - Einheit

8.2.1 ER-Modell

Der Modellentwurf folgt im Allgemeinen der technischen 4.3 Spezifikation Einheiten und berücksichtigt insbesondere das 4.3.1 Schema - Einheit - v1.0.

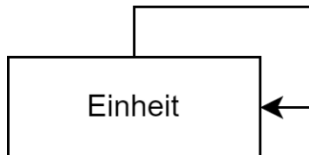


Abbildung 8.2: ER-Modell Einheit

Die Entität UnitLKE beschreibt die fachliche Entität Einheit. Die Entität beinhaltet folgende Attribute:

- unitId: ViDaL-weite ID einer Einheit
- unitOwner: KT OID - Verwalter der Einheit, wird beim Anlegen einer Einheit festgelegt und kann nicht geändert werden.
- unitDoc: JSON-Objekt - beschreibt alle Attribute der Einheit.

Für die Entität UnitLKE sind folgende Operationen definiert:

- Neuanlage
- Modifizieren, auch partielles
- Löschen
- Suchabfragen nach Werten einzelner Einheitsattribute, allerdings nicht nach den Attributen untergeordneter Einheiten

Das Modifizieren von Eigenschaften einer Einheit sowie das Löschen einer Einheit ist nur durch den Verwalter der Einheit erlaubt.

Das Löschen einer Einheit ist nur dann möglich, wenn diese Einheit einen der ViDaL-Statuswerte (2, 6, 8) hat.

Der Datensatz unitDoc beinhaltet Referenzen / ID von untergeordneten Einheiten / Untereinheiten sowie zugeordneten Ressourcen. Diese Referenzen stellen sogenannte schwache Referenzen dar. Das bedeutet, dass auf der Datenmodellebene beider Anwendungen (Einheiten und Ressourcen) keine Mechanismen implementiert sind zum Schutz vor Löschung einer Einheit / Ressource aufgrund einer existierenden Referenz auf diese Einheit / Ressource in einer Einheit-Instanz.

Der Datensatz unitDoc stellt einen strukturierten Datensatz dar (JSON-Dokument). Somit ist es möglich, Suchabfragen nach Werten einzelner JSON-Elemente (JSON-Path) auszuführen.

8.3 LDB Datenmodell – Resource

8.3.1 ER-Modell

Der Modellentwurf folgt im Allgemeinen der technischen 4.4 Spezifikation Ressourcen und berücksichtigt insbesondere das 4.4.1 Schema - Ressource - v1.0.

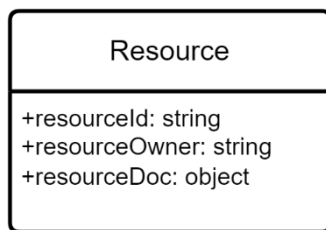


Abbildung 8.3: ER-Modell Resource

Die Entität Resource beschreibt die fachliche Entität Ressource. Die Entität beinhaltet folgende Attribute:

- resourceId: VIDaL-weite ID einer Ressource
- resourceOwner: KT OID - Eigentümer der Ressource, wird beim Anlegen einer Ressource festgelegt und kann nicht geändert werden.
- resourceDoc: JSON-Objekt - beschreibt alle Attribute der Ressource.

Für die Entität Resource sind folgende Operationen definiert:

- Neuanlage
- Modifizieren, auch partielles
- Löschen
- Suchabfragen nach Werten einzelner Ressourcenattribute

Das Modifizieren von Eigenschaften einer Ressource sowie das Löschen einer Ressource ist nur durch den Eigentümer der Ressource erlaubt.

Der Datensatz resourceDoc beinhaltet eine Referenz auf die verwaltende Organisation. Diese Referenz stellt eine sogenannte schwache Referenz dar. Das bedeutet, dass auf der Datenmodellebene beider Anwendungen (Ressourcen und Organisationen) keine Mechanismen implementiert sind zum Schutz vor Löschung einer Organisation aufgrund einer existierenden Referenz auf diese Organisation in einer Ressource-Instanz.

Der Datensatz resourceDoc stellt einen strukturierten Datensatz dar (JSON-Dokument). Somit ist es möglich, Suchabfragen nach Werten einzelner JSON-Elemente (JSON-Path) auszuführen.

8.4 LDB Datenmodell - Einsatzstatistik

8.4.1 ER-Modell

Der Modellentwurf folgt im Allgemeinen der technischen 4.5 Spezifikation Kontinuierliche Einsatzstatistik und berücksichtigt insbesondere das 4.5.1 Schema - Einsatzstatistik - v1.0.

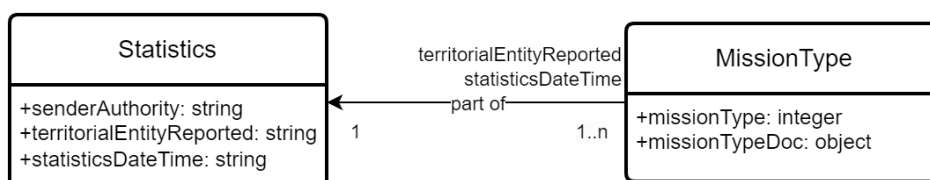


Abbildung 8.4: ER-Modell Einsatzstatistik

Die Entität Statistics beschreibt übergreifende Informationen zur fachlichen Entität Kontinuierliche Einsatzstatistik. Die Entität beinhaltet folgende Attribute:

- senderAuthority: Behörde (z.B. Übernahme automatisch aus Registry)

- territorialEntityReported: Geografischer Bereich auf den sich die gemeldeten Daten beziehen. Es kann sein, dass ein ELS für mehr als einen derartigen geographischen Bereich zuständig ist
- statisticsDateTime: Statistikdatenerfassungszeitpunkt

Die einzelnen Statistikdaten aggregiert nach Einsatztypen sind durch die Entität MissionType beschrieben. Diese Entität beinhaltet folgende Attribute:

- missionType: Eindeutige Nummer des Statistiktyps entsprechend "VIDaL AP1 Anl7 Inhalte Kontinuierliche Einsatzstatistik v10"
- missionTypeDoc: JSON-Objekt - beschreibt alle relevanten statistischen Werte zu Status, Personen, Ressourcen aggregiert nach dem Statistiktyp entsprechend "VIDaL AP1 Anl7 Inhalte Kontinuierliche Einsatzstatistik v10"

Die Entität MissionType beschreibt Einzelteile der gesamten statistischen Auswertung durch ein ELS zu einem bestimmten Zeitpunkt - siehe Beziehung "part of".

Für das Datenmodell Einsatzstatistik sind folgende Operationen definiert:

- Aktualisierung der Statistik für den geografischen Bereich der ELS zu einem bestimmten Zeitpunkt. Die Aktualisierung überschreibt den alten Stand vollständig.
- Suchabfragen nach Werten einzelner Attribute der Einsatzstatistik. Die Implementierung kann ein Parameter "Datenaktualität" vorsehen. Das ist eine Zeitspanne nach der ein gemeldeter Statistikdatensatz nicht mehr als aktuell betrachtet wird. Folglich wird ein solcher Datensatz bei Abfragen nicht mehr ausgeliefert.

Der Datensatz missionTypeDoc stellt einen strukturierten Datensatz dar (JSON-Dokument). Somit ist es möglich, Suchabfragen nach Werten einzelner JSON-Elemente (JSON-Path) auszuführen.

8.5 LDB Datenmodell - Organisation

8.5.1 ER-Modell

Der Modellentwurf folgt im Allgemeinen der technischen Spezifikation Organisationen und berücksichtigt insbesondere das Schema – Organisation – v1.0.

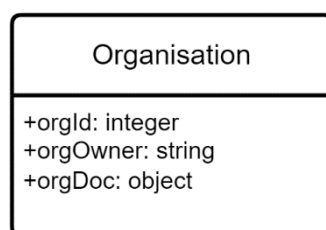


Abbildung 8.5: ER-Modell Organisation

Die Entität Organisation beschreibt die fachliche Entität Organisation. Die Entität beinhaltet folgende Attribute:

- orgId - VIDaL-weite ID einer Organisation
- orgOwner - KT OID - Eigentümer der Organisation, wird beim Anlegen einer Organisation auf die KT ID des Request-Senders festgelegt und kann nicht geändert werden.
- orgDoc - JSON-Objekt - beschreibt alle Attribute der Organisation.

Für die Entität Organisation sind folgende Operationen definiert:

- Neuanlage

- Modifizieren, auch partielles
- Löschen
- Suchabfragen nach Werten einzelner Organisationsattribute

Das Modifizieren von Eigenschaften einer Organisation sowie das Löschen einer Organisation ist nur durch den Eigentümer der Organisation oder durch den LDB-Administrator in der Rolle Organisationsadministrator erlaubt.

Der Datensatz orgDoc beinhaltet Referenzen / ID auf anderen zusammenhängenden Organisationen. Diese Referenzen stellen sogenannte schwache Referenzen dar. Das bedeutet, dass auf der Datenmodellebene keine Mechanismen implementiert sind, zum Schutz vor Löschung einer Organisation aufgrund einer existierenden Referenz auf diese Organisation in einer anderen Organisation-Instanz.

Der Datensatz orgDoc stellt einen strukturierten Datensatz dar (JSON-Dokument). Somit ist es möglich, Suchabfragen nach Werten einzelner JSON-Elemente (JSON-Path) auszuführen.

Abkürzungsverzeichnis

Abkürzung	Beschreibung
AAA	Authentication, authorization, accounting
AG	Auftraggeber
AGS	Allgemeiner Gemeindeschlüssel
API	application programming interface (Programmierschnittstelle)
E2E	Ende zu Ende
ELS	Einsatzleitsystem
ER	Entity Relationship (Modell zur Darstellung von Beziehungen)
HA	High availability
IAM	Identity & Access Management
IM	Interface Manager
KM	Krisenmanagement
KT	Kommunikationsteilnehmer
LAN	Local area network
LDB	Lagedatenbank
LDD	Lagedokumentationsdienst
LM	Lagemodul
LPF	Lageplattform
LKE	Einheiten gem. Landeskonzepten
MGMT	Management
mTLS	Mutual transports layer security, Authentifizierungsmethode
OID	Object Identifier
PKI	Private Key Infrastructure
RSA	Rivest–Shamir–Adleman, asymmetrisches kryptografisches Verfahren
SSH	Secure shell
WAN	Wide area network

Änderungshistorie / Release Notes

Version	Stand	Autor/Bearbeiter	Änderungen/Kommentar
0.9	20.10.2023	Frank Rottländer	Auslieferung
0.9.2	16.11.2023	Alexander Kryukov	Überarbeitung nach Review
0.9.3	26.11.2023	Alexander Kryukov	Aufnahme Organisationen
1.0	26.11.2023	Alexander Kryukov, Frank Rottländer	Finale Überarbeitung nach Freigabe
1.1	01.02.2024	Alexander Kryukov	Klärung von Validierungsregeln Pflichtfeld-Systematik überarbeitet aufgrund technischer Einschränkungen auf JSON- Ebene.
1.2	09.08.2024	Alexander Kryukov	Auslieferung Inhalt komplett überarbeitet.
1.3	06.09.2024	Frank Rottländer	Auslieferung Einarbeitung von Reviewkommentaren
1.4	16.10.2024	Frank Rottländer	Anpassung von Formatierung und erneute Auslieferung