Spezifikation Universal Control Room Interface (UCRI2)
Version 2.0.0

Inhaltsverzeichnis

1. Einleitung

UCRI steht für Universal Control Room Interface - ein Protokoll für die Kommunikation zwischen zwei oder mehreren Einsatzleitsystemen.

Nach einer erfolgreichen Einführung und eingehender Approbation im Feld wurden viele Erfahrungen gesammelt und Anforderungen identifiziert, die Weiterentwicklung des UCRI Protokolls auf einer neuen architektonischen Basis erforderten. UCRI2 ist eine komplett überarbeitete Version des Protokolls, die alle Anwendungsfälle der Vorgängerversionen (die letzte UCRI Version 1.1) unterstützt und Grundlage für flexible Weiterentwicklung des Protokolls darstellt.

1.1. Ziele für UCRI2

Ziele für die Entwicklung von UCRI2 und die Unterschiede zu UCRI 1.x sind:

1. Einfache, schnell zu implementierende API, schnelles PoC möglich
    1.1. standardisierte maschinenlesbare Spezifikation
    1.2. sichere Zustellung und Verarbeitung von Meldungen
2. Trennung technischer und fachlicher Aspekte
    2.1. Einfache Erweiterbarkeit
    2.2. Technische Komponenten müssen bei fachlichen Erweiterungen nicht angepasst werden
    2.3. Fachliche Erweiterungen können ohne Anpassungen der Infrastruktur erfolgen
3. Routing im Kern des Konzeptes
    3.1. Einfaches Zusammenspiel mehrerer Hersteller
    3.2. Unterstützung zentraler als dezentraler Anbindungen
4. Die RPC/REST basierte Architektur in eine Messaging basierte Architektur überführen
5. Authentifizierung OAuth2 konform

2. UCRI2 Systemarchitektur

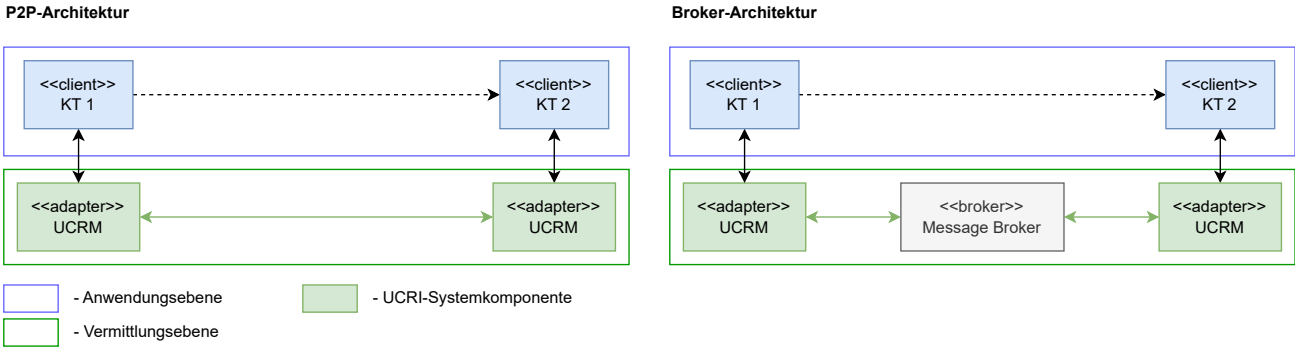
Im Gegensatz zu UCRI Version 1 ist UCRI 2 grundlegend für die n:m-Kommunikation verschiedener Teilnehmer entwickelt worden.

2.1. Messaging

Bei der Strukturierung der UCRI2-Schnittstelle wird der Architekturstil Messaging verwendet. Bei diesem Architekturstil kommunizieren verteilte unabhängige Systemkomponenten (im allgemeinen Kommunikationsteilnehmer genannt - KT) miteinander mit Hilfe von Nachrichten.

Messaging-Systeme trennen die fachliche Anwendung (gewöhnlich strukturiert nach Client-Server-Prinzip) von der Vermittlungsebene - also von technischen Aspekten der Nachrichtenübermittlung zwischen technischen Systemen der KT. Nachrichten können während der Übertragung umgewandelt werden, ohne dass Sender oder Empfänger von der Umwandlung wissen. Die Entkopplung ermöglicht es Integratoren, je nach Anforderung unterschiedliche Kommunikationstopologien zu unterstützen, von dezentralen P2P-Protokollen bis zu zentralisierten Broker-Architekturen.

Messaging-Systeme ermöglichen 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.



2.2. Vermittlungsebene

Das andere wichtige Architekturmuster, das bei der Strukturierung der UCRI2-Schnittstelle Verwendung findet, ist das Adapter-Muster. Bei diesem Muster erfolgt die Kommunikation zwischen dem technischen System der KT (Anwendungsebene) und der Vermittlungsebene mittels einer Adapter-Komponente Leitstellenmodul (UCRI Control Room Module - UCRM). Der Adapter ermöglicht bidirektionale Kommunikation zwischen den Ebenen in einer standardisierten Form und ermöglicht die Komplexitätsreduzierung der angebundenen Schnittstellen.

Das UCRM stellt die UCRM Client API bereit - die einzige Kommunikationsschnittstelle für direkt verbundene Kommunikationsteilnehmer wie Leitstellensysteme oder andere technische Knoten, sowie weitere externe Systeme (vgl. UCRI Gateway).

Der Adapter kann auch zusätzliche Aufgaben übernehmen, wie z. B. Datentransformation oder Sicherheitsaufgaben wie Ende-zu-Ende-Verschlüsselung von übermittelten Nachrichten.

Die untereinander kommunizierenden UCRM - Adapter an der Seite von technischen Systemen der KT und optional der Broker (bei einer zentralen Broker-Architektur) bilden die Vermittlungsebene und 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.

Zentrale Aufgabe der Vermittlungsebene ist die Zustellung von Meldungen zwischen Sender und Empfänger. Außerdem werden auf der Vermittlungsebene unterschiedliche querschnittliche Aufgaben übernommen. Hier sind nur einige Beispiele: KT-Authentifizierung und Autorisierung, Meldungsvalidierung, eventuell E2E-Verschlüsselung. Bei der P2P-Topologie kommunizieren die UCRM der zwei KT dabei direkt miteinander, bei einer zentralen Architektur - über einen Messagebroker.

Einzelne Aufgaben der Vermittlungsebene, die bei der P2P-Topologie durch die Kommunikation zwischen einzelnen UCRMs umgesetzt werden, sind:

1. Verwaltung der Kommunikationstopologie inkl. Adressierungskonzept, KT-Status-Monitoring und KT-Register
2. Konzept Authentisierung, Autorisierung, Accounting
3. Optional E2E-Verschlüsselung
4. Übermittlung von Nachrichten unter der Verwendung des UCRI-Adressierungskonzepts inkl. Routing-Funktion
5. Validierung von Anwendungsmeldungen

Die Vermittlungsebene ist frei von Fachlichkeit. Sie realisiert nur den Datentransport und sichert optional die Ende-zu-Ende-Verschlüsselung der Daten.

2.3. Anwendungsebene

Um die fachlichen Aspekte von den technischen Aspekten zu trennen, erfolgt in UCRI 2 eine Trennung zwischen Übertragungs- und Anwendungsebene. Hierbei erfolgt sowohl eine Trennung der UCRI2-Anwendungen untereinander und auch deren Unabhängigkeit

von der Vermittlungsebene. Das erlaubt eine freie Weiterentwicklung jeder einzelnen Anwendung.

Eine UCRI2-Anwendung (im Folgenden UCRI2-App) wird durch folgende Artefakte definiert:

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

2.4. Rollen der beteiligten Komponenten

In UCRI2 existieren demnach zwei unterschiedlichen Rollen:

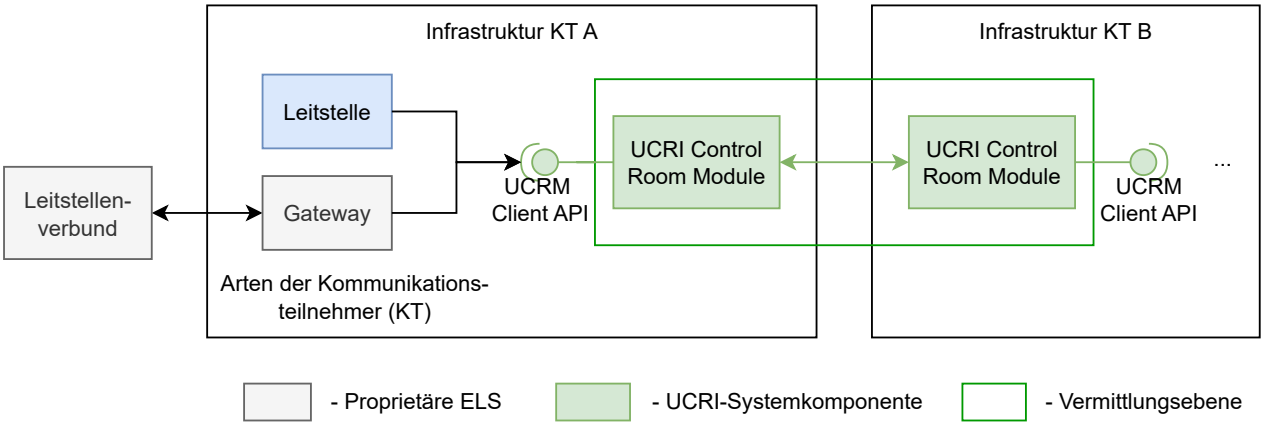
1. Ein teilnehmendes System nimmt hierbei die **Client-Rolle** ein und konsumiert die UCRI2 Client API, die vom UCRM angeboten wird.
2. Ein UCRM nimmt dagegen die **UCRM-Rolle** ein und bietet für Clients die UCRI2 Client API an. Zur Verbindung mit anderen UCRMs wird je nach Kommunikationstopologie eine andere API genutzt. Falls eine Verbindung zwischen UCRMs verschiedener Hersteller erfolgen soll, muss dies die UCRI2 Peer-to-Peer-API (P2P-API) sein.

Folglich ist für eine UCRI2-konforme Umsetzung relevant, welche Rolle umgesetzt werden soll:

1. Ein KT, der sich per UCRI2 mit anderen KT verbinden will, muss ausschliesslich die UCRI2 Client API aus der Consumer-Perspektive umsetzen.
2. Ein UCRM muss die UCRI2 Client API aus der Provider-Perspektive umsetzen und kann die UCRI2 P2P-API umsetzen, falls eine herstellerübergreifende Koppelung mit anderen UCRMs gewünscht wird.

2.5. UCRI Gateway

Die Systemkomponente Gateway stellt einen spezialisierten KT dar. Das Gateway wird am Übergang zu Gruppen von KT eingesetzt, die auf der Vermittlungsebene nicht direkt erreichbar sind und über eine proprietäres Leitstellenprotokoll angebunden werden können (Leitstellenverbunde). Das Gateway stellt eine Gateway-Funktion bereit zum Mapping zwischen externe Quell- bzw. Zieladressen und UCRI-internen KT-Adressen.



2.6. Adressierungskonzept

Für die Adressierung der einzelnen Kommunikationsteilnehmer (KT) auf der Vermittlungsebene werden eindeutige Kennungen in Form von Object Identifier (OID Spezifikationen ISO/IEC 9834, DIN 66334) verwendet. Für die Festlegung von OIDs gelten folgende Vorgaben:

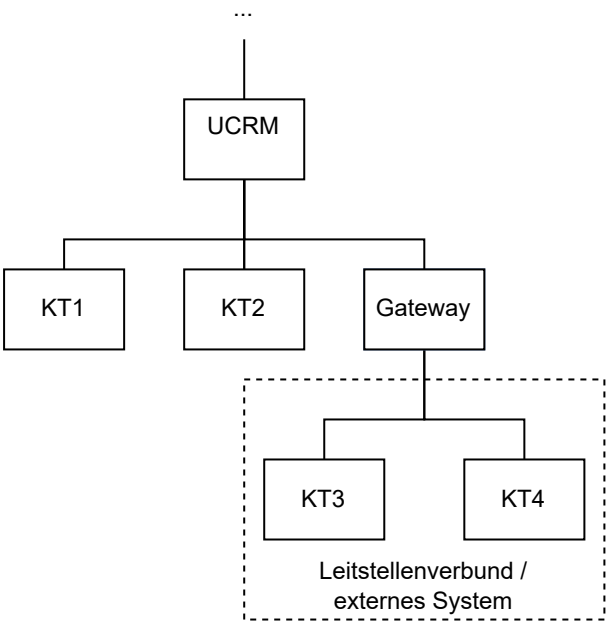
1. Falls möglich, sollten offiziell zugeteilte OIDs zum Einsatz kommen. Diese können z.B. für das deutsche Gesundheitswesen beim Bundesinstitut für Arzneimittel und Medizinprodukte beantragt werden.
2. Falls keine offiziell vergebenen OIDs zum Einsatz kommen, dürfen selbst vergebene OIDs NIEMALS in einem Adressraum liegen, in dem auch offiziell zugeteilte OIDs vergeben werden, um Überschneidungen mit offiziell zugeteilten OIDs zu vermeiden.

Zusätzlich wird empfohlen, die genutzten OIDs dem Expertenforum UCRI zu melden, damit Adresskonflikte direkt bei der Vergabe vermieden werden können.

Die in den folgenden Unterkapiteln vorgestellten beispielhaften OID-Hierarchien und OID-Nomenklatur stellen nur eine Empfehlung dar. Falls von offiziellen Stellen vergebene OIDs zum Einsatz kommen, entsteht nicht zwangsläufig auch eine Hierarchie, wie sie in den Folgenden Unterkapiteln dargestellt wird.

2.6.1. Beispielhafte OID-Hierarchie

Die Adressierung von einzelnen KT kann hierarchisch organisiert werden und spiegelt dann die hierarchische Kommunikationsstruktur wider:



Diese Struktur ermöglicht Implementierung einer einfachen und einheitlichen Routing-Funktion, die in jeder Systemkomponente (Leitstellenmodul - UCRM, evtl. Messagebroker, UCRI-Gateway) die Weiterleitung von übermittelten Nachrichten unterstützt (TODO Routing-Konzept).

Die Systemkomponente Gateway stellt einen speziellen 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 internen OID. Ein externes System bekommt dabei einen entsprechend reservierten OID-Bereich (Unterbaum) und das Gateway bekommt die Wurzel-OID-Adresse dieses Unterbaums.

2.6.2. Beispielhafte OID-Nomenklatur

Alle Kommunikationsteilnehmer in einem UCRI-System bekommen eine in diesem System eindeutige OID zugewiesen, die sich in einem OID-Adressraum befindet. Dieser OID-Adressraum wird durch eine Wurzeladresse (im weiteren Verlauf als bezeichnet) festgelegt. Die Strukturierung des UCRI-OID-Adressierungsraums (UCRI-OID-Nomenklatur) ermöglicht Adressierung von KT über die staatlichen Grenzen hinaus.

Dabei steht jedem Land frei, ein eigenes Adressierungsschema unterhalb des Landes-OID-Unterbaums zu definieren. Wurzeladresse eines Landes-OID-Unterbaums ist wie folgt definiert:

1. <Root-OID>.1.<Kennzahl des Landes> (Beispiel: .1.276 für Deutschland)

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

1. <Root-OID>.1.276.<Kennzahl des Bundeslandes> (Beispiel: .1.276.5 für NRW)

Unterschiedliche Organisationen können auch unterschiedliche Adressierungsschemata verwenden. Wurzel-Adresse eines nPOLGA-OID-Unterbaums ist wie folgt definiert:

1. <Root-OID>.1.276.5.<Kennzahl von POL/nPOLGA, etc.> (Beispiel: .1.276.5.1 für nPOLGA in NRW)

Die OID-Adresse der einzelnen Kommunikationsteilnehmer (KT) wird nach dem amtlichen Gemeindeschlüssel (AGS) gebildet:

#	Ebene	Beispiel
1	Root-OID	1.2.3 - Festlegung in einem geschlossenen UCRI-System
2	Satzart	1 - Einzeladresse, 2 - Gruppe (aktuell werden in UCRI keine Gruppenadressen unterstützt)
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. Definition folgt. Annahme für Beispiel: 1 - nPOLGA, 99 - Test/Pilot
6	Kennzahl des Regierungsbezirks	1 - Regierungsbezirk Düsseldorf nach AGS
7	Kennzahl des Landkreises oder der kreisfreien Stadt ("0" bei zentralen Einrichtungen auf der Regierungsbezirksebene)	58 - Landkreis Mettmann nach AGS
8	Gemeinde ("0" bei kreisfreien Städten)	28 - Stadt Ratingen, Stadtbezirk Ratingen nach AGS
9	Leitstellenmodul	1 - erstes Leitstellenmodul, fortlaufende Nummerierung von Leitstellenmodulen
10	Kommunikationsteilnehmer	1 - z.B. ELS, fortlaufende Nummerierung von KT

Beispiel OID Feuerwehr ELS in Ratingen: .1.276.5.1.1.58.28.1.1

2.7. Versionierung

Die Versionierung für die Transportschicht und die UCRI2-Apps erfolgen voneinander getrennt.

Diese Spezifikation beschreibt die Version 2.0.0 der UCRI2-Transportschicht.

2.7.1. Transportschicht-Versionierung

Die Transportschicht-Versionierung umfasst dieses Dokument sowie die OpenAPI-Spezifikationen für die Client- und die P2P-Schnittstellen.

Die Versionsnummer besteht aus drei numerischen Teilen:

GEN.MAJOR.MINOR

Für die drei Versionsbestandteile gelten folgende Festlegungen:

1. GEN: Für UCRI2 auf 2 festgelegt
2. MAJOR: Hauptversion der Transportschicht innerhalb der UCRI2-Versionierung. Eine Änderung dieser Version erfolgt, wenn Änderungen an den Endpunktdefinitionen erfolgen, die entweder bestehende Endpunkte bezüglich obligater Felder verändern oder neue obligate Felder hinzufügen.
3. MINOR: Unterversion der Transportschicht. Eine Änderung dieser Version erfolgt, wenn neue optionale Felder zu bestehenden Endpunkte hinzugefügt oder neue optionale Endpunkte hinzugefügt werden.

Somit sind Änderungen an der MINOR-Version aufwärtskompatibel, sodass Systeme mit übereinstimmenden GEN.MAJOR-Versionen untereinander kommunizieren können, auch wenn sie unterschiedliche MINOR-Versionen aufweisen.

2.7.1.1. Vorgabe der transport_layer_messages-App-Version

Da die transport_layer_messages-App Nachrichten beschreibt, die auf der Transportschicht erstellt und von verbundenen UCRM sowie Clients konsumiert werden, MUSS eine spezifischen Version dieser App durch alle Clients sowie UCRM, welche die Version 2.0.0 der Transportschicht implementieren, ZWINGEND unterstützt werden.

Die hierfür zu unterstützende Version der transport_layer_messages-App ist 1.0.

2.7.2. App-Versionierung

Die App-Versionierung erfolgt für jede App individuell. Die Versionsnummer einer App besteht aus zwei numerischen Teilen:

MAJOR.MINOR

Für die drei Versionsbestandteile gelten folgende Festlegungen:

1. MAJOR: Hauptversion der App. Eine Änderung dieser Version erfolgt, wenn neue obligate Nachrichten hinzugefügt werden oder in bestehenden Nachrichten obligate Felder hinzugefügt oder verändert werden.
2. MINOR: Unterversion der App. Eine Änderung dieser Version erfolgt, wenn neue optionale Nachrichten hinzugefügt werden oder in bestehenden Nachrichten optionale Felder hinzugefügt werden.

Obwohl so Änderungen an der MINOR-Version aufwärtskompatibel sind, müssen auch MINOR-Veränderungen in den supportedApps eines Teilnehmers explizit als unterstützt gekennzeichnet werden.

3. !!! TODO: In Systemarchitektur-Kapitel Verschieben!!! UCRI2 Vermittlungsebene

Die Vermittlungsebene umfasst technische Aspekte der Nachrichtenübermittlung zwischen technischen Systemen der KT. Dabei können unterschiedliche Kommunikationstopologien unterstützt werden, von dezentralen Peer-To-Peer (P2P) Protokollen bis zu zentralisierten Broker-Architekturen.

Die aktuelle UCRI2 Version spezifiziert nur ein P2P-Protokoll für die Kommunikation zwischen den UCRM-Modulen.

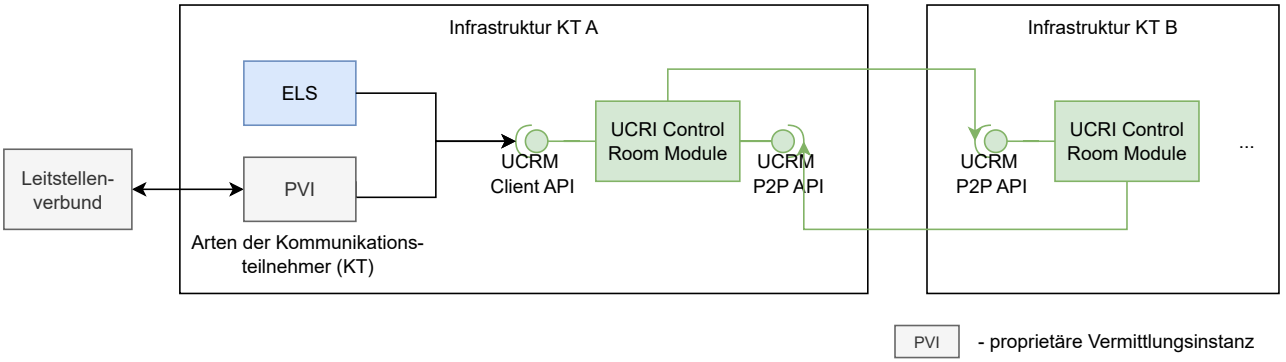
Im Weiteren werden einzelne Aspekte der Nachrichtenübermittlung detailliert beschrieben:

1. Adressierungskonzept
2. UCRI Leitstellenmodul
3. Kommunikationsprotokoll

4. !!! TODO Aufteilen und Verschieben nach Systemarchitektur und API-Kapitel!!! UCRI Leitstellenmodul

4.1. Überblick

Das UCRI Leitstellenmodul (UCRI Control Room Module, UCRM) stellt die API bereit - die einzige Kommunikationsschnittstelle für verbundene Kommunikationsteilnehmer wie zum Beispiel Einsatzleitstellensysteme, proprietäre Vermittlungsinstanzen (PVI), sowie andere technische Systeme:



4.2. UCRM API Technologie

Das Ziel des UCRI-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 UCRM API die technologische Variante REST API mit Polling festgelegt. REST API mit Polling von Nachrichten hat folgende Vorteile:

1. einfach zu implementieren
2. einfach zu konsumieren

Um die Auswirkung des Pollings auf die Systemreaktionszeit bei Meldungsaustausch (die maximale Zeit zwischen den Meldungssende- und Meldungsempfangszeitpunkten) zu minimieren, kann bei Meldungsaustausch ein Long Polling vorgesehen werden.

4.2.1. Protokoll

!!!TODO Details sind bereits in der Client-API beschrieben!!!

Wegen der Anforderung zur sicheren Nachrichtenzustellung wird eine technische Nachrichtenempfangsbestätigung in der UCRM API vereinbart.

Das Prinzip der sicheren Nachrichtenzustellung ist ein E2E-Prinzip, das auch die Verarbeitungslogik bis zum Persistieren der Nachrichtendaten auf der Seite des KT-Systems einschließt. Um die Ausfälle in dieser Empfangs- und Verarbeitungslogik 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 aus der Vermittlungsebene verworfen und stehen beim erneuter Meldungsabfrage nicht mehr zur Verfügung.

Zum Signalisieren der Nachrichtenzustellung werden technische Quittungen für eine gesendete Nachricht implementiert (positive sowie negative Zustellbestätigungen).

Ein Nachrichtensender bekommt folgende Quittungen:

1. Zustellbestätigung: nachdem der Empfänger die Entgegennahme einer Nachricht bestätigt hat
2. Benachrichtigung über fehlgeschlagene Zustellung: 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.

!!!TODO Details sind bereits in der Client-API beschrieben ENDE!!!

4.2.2. Empfehlungen zur technischen Umsetzung

!!!TODO Inhalte wurden in client-api.md übernommen!!!

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 in Sekundenbereich (3 - 5 Sekunden) scheint optimal zu sein. Um die Auswirkung des Pollings auf die Systemreaktionszeit bei Meldungsaustausch zu minimieren, wird Verwendung eines Long Polling empfohlen.

4.2.2.1. Polling bei Nachrichtenabfragen

Folgende Schleife ist bei Nachrichtenabfragen bei Polling zu empfehlen:

1. Nachrichten abfragen mit Angabe maximaler Nachrichtenzahl Nmax
2. Nachrichten verarbeiten
3. Nachrichtenempfang bestätigen
4. Ist die Anzahl von empfangenen Nachrichten gleich Nmax - sofort zum Schritt 1 übergehen
5. Konfigurierte Pause einlegen

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

4.2.2.2. Long Polling bei Nachrichtenabfragen

Folgende Schleife ist bei Nachrichtenabfragen bei Long Polling zu empfehlen:

1. Nachrichten abfragen mit Angabe maximaler Nachrichtenzahl Nmax und maximaler Wartezeit Tmax
2. Nachrichten verarbeiten
3. Nachrichtenempfang bestätigen

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

!!!TODO Inhalte wurden in client-api.md übernommen ENDE!!!

4.3. UCRM REST API

!!!TODO Inhalte wurden in apis.md übernommen!!!

UCRM API Design verwendet REST API Design Richtlinien erarbeitet bei TM Forum.

UCRM API Spezifikation verwendet Standards erarbeitet bei der OpenAPI Initiative.

UCRM API ist in folgende fachliche Bereiche aufgeteilt:

!!!TODO Inhalte wurden in apis.md übernommen ENDE!!!

1. KT-Registry - Abfragen von Eigenschaften der registrierten Kommunikationsteilnehmer
2. Messaging - Versenden und Empfangen von Nachrichten
3. Info - Die Info API liefert Informationen über die Version und den Betreiber der Schnittstelle
4.

4.3.1. Berechtigungskonzept

Die UCRM API wird sowohl KT-seitig als auch für die Inter-CRM-Kommunikation verwendet.

Es wird ein einfaches Berechtigungskonzept verwendet, das folgende Rollen vorsieht:

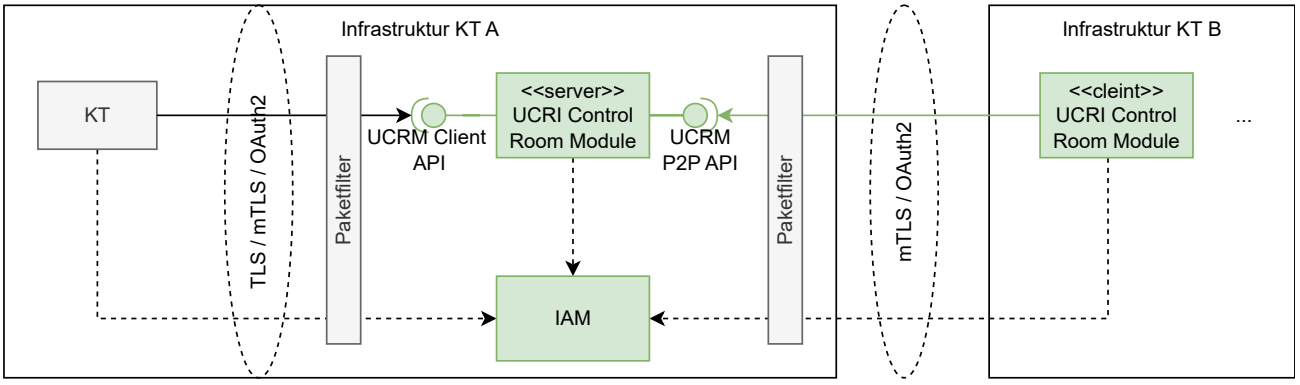
1. KT - für die Kommunikation von KT-Systemen zu UCRM
2. UCRM - für Inter-CRM-Kommunikation

Die Rolle wird an die API-Implementierung mittels eines HTTP-Headers übergeben.

5. !!!TODO Aufteilen und Verschieben nach Systemarchitektur und API-Kapitel!!! UCRI2 Kommunikationsprotokoll

UCRI2 unterscheidet zwei Kommunikationsdomäne, siehe Abbildung:

1. Kommunikation zwischen einem Leitstellensystem (KT-System) und einem UCRI Leitstellenmodul (UCRM). Diese Kommunikation findet anhand der UCRM Client API statt und erfolgt typischerweise innerhalb einer durch einen Leitstellenbetreiber kontrollierten Infrastruktur.
2. Kommunikation zwischen zwei UCRM. Die Inter-UCRM-Kommunikation verwendet UCRM P2P API. Diese Kommunikation kann über das öffentliche Internet stattfinden und braucht dementsprechend besondere Sicherheitsmaßnahmen.



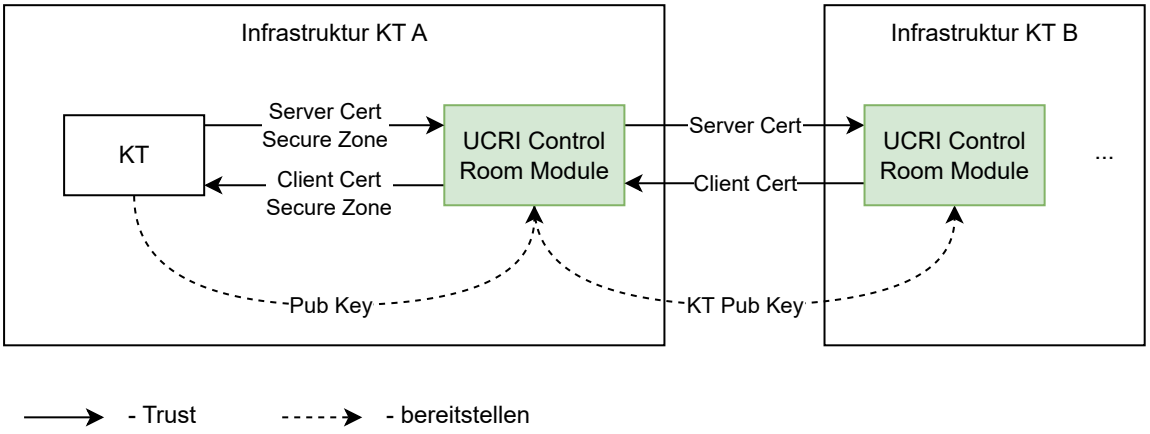
Die Infrastruktur eines KTs sollte eine P-A-P-Struktur aufweisen, die aus Paketfilter als Trennung zu vertrauenswürdigen internen Systemen (Leitstelle), Application-Layer-Gateway (UCRM) und Paketfilter als Trennung zu nicht vertrauenswürdigem Netz (Internet) besteht. Vgl. Netzarchitektur und -design - BSI.

UCRM API Implementierung verwendet in beiden Kommunikationsdomänen OAuth 2.0 mit Client Credentials Grant Type zur sicheren Authentifizierung und Autorisierung von Clients auf der Applikationsebene.

5.1. Trust Konzept

Zwischen zwei UCRM wird mTLS als Sicherung der Inter-UCRM-Kommunikation verwendet. Sowohl Client als auch Server bauen eine gegenseitige Vertrauensbeziehung über digitale Zertifikate auf. Dabei kann der Paketfilter auf der Seite des nicht vertrauenswürdigen Netzes die mTLS-Verbindung terminieren. Dazu kann der Paketfilter eine Liste von zugelassenen Domainnamen beinhalten (White Listing) und somit die Kommunikation auf eine geschlossene Gruppe der Teilnehmer beschränken.

Die Kommunikation zwischen einem KT-System und dem UCRM-Modul kann je nach lokalen Sicherheitsanforderungen wahlweise per TLS oder mTLS erfolgen.



Die Vertrauensbeziehungen zwischen den Systemkomponenten bei der Client/Server-Kommunikation basieren dabei auf folgenden Mechanismen (siehe Abbildung):

1. Server-Authentifizierung. Die Vertrauensbeziehung zu UCRM als Server basiert auf dem Domainname (DNS) des Servers und wird durch Server-Zertifikat abgesichert. Der Domainname des Servers ist im Server-Zertifikat verankert.
2. Client-Authentifizierung. Die Vertrauensbeziehung zu KT oder UCRM in der Client-Rolle basiert auf der Object Identifier (OID) des Clients (siehe Adressierungskonzept) und wird durch Client-Zertifikat abgesichert. Die OID des Clients ist im Client-Zertifikat verankert.
3. Sowohl Client als auch Server haben eine Liste vertrauenswürdiger Zertifizierungsstellen (CAs), sogenannte Root-CAs, die als Vertrauensanker dienen. Die ausgestellten Zertifikate von Client und Server müssen von einer CA signiert sein, die jeweils in der vertrauenswürdigen Liste des Gegenübers enthalten ist.

Anmerkung 1: In den konkreten Kundensituationen können unterschiedliche Mechanismen des Zertifikatsmanagements umgesetzt werden - basierend sowohl auf zentraler Authority als auch auf spezifischen Vereinbarungen zwischen einzelnen KTs.

Anmerkung 2: Als ALternative kann das Vertrauen zwischen KT-Systemen und dem UCRM in einer durch einen Leitstellenbetreiber kontrollierten Infrastruktur durch das Bilden von einer Sicherheitszone auf der Netzwerkebene hergestellt werden.

Um den sichernen Ursprung und unverfälschten Inhalt von Applikationsmeldungen zu garantieren, können die Meldungen durch Sender-KT signiert werden. Zur Signaturvalidierung erstellt der KT ein kryptographisches Schlüsselpaar und stellt sein Public Key dem lokalen UCRM-Modul über sicheren Kanal bereit. Bei der Inter-UCRM-Kommunikation wird das KT Public Key an die fremden UCRM mittels UCRM P2P API übermittelt und steht anschließend den potentiellen Meldungsempfängern über die UCRM Client API zur Signaturvalidierung zur Verfügung.

5.2. Nachrichtenübermittlung

Die ausgehenden Nachrichten übergibt das KT-System an das UCRM-Modul m.H.v. Client API-Endpunkt /send. Die Nachrichten werden dann durch lokales UCRM mittels P2P API-Endpunkt /send gegen Empfänger-UCRM gepuscht. Für das Handling von Nichtverfügbarkeit der Partner-UCRMs wird in dem lokalen UCRM ein Ausgangspuffer implementiert.

5.3. KT-Register

Jeder UCRM baut einen lokalen Cache des KT-Registers durch regelmäßige Aufrufe der KT-Register API /registry an allen bekannten Partner-UCRM.

Die Umsetzung des UCRM API-Endpunktes /registry unterscheidet sich in beiden UCRM API:

1. Der Client API-Endpunkt /registry liefert alle bekannten KT.
2. Der P2P API-Endpunkt /registry liefert nur eigene KT.

Da die Umgebung dynamisch ist, d.h. neue KT können dazukommen, existierende KT können abgebaut werden, sollen KT-Register-Abfragen sowohl KT-seitig als auch in der Inter-UCRM-Kommunikation regelmäßig durchgeführt werden. Empfehlung: mindestens 1 Mal pro Stunde, maximal alle 5 Minuten.

5.4. E2E Verschlüsselung und Datenintegrität

Als Richtlinie für die Auswahl kryptographischer Verfahren für die Verschlüsselung und Signieren von Nachrichten dient die BSI Technische Richtlinie (https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf?__blob=publicationFile&v=9).

Das asymmetrische kryptographische RSA-Verfahren RSASSA-PKCS1-v1_5 (RFC 3447: Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1) wird sowohl zum Verschlüsseln als auch zum digitalen Signieren der Nachrichten verwendet.

Das Verfahren verwendet ein für jeden KT generiertes Schlüsselpaar, bestehend aus einem privaten Schlüssel und einem öffentlichen Schlüssel. Der private Schlüssel wird im Keystore des KT-Systems gespeichert.

Die öffentlichen Schlüssel werden über das KT-Register als Teil der verfügbaren KT-Daten in Form von JSON Web Key (JWK, RFC 7517: JSON Web Key (JWK)) ausgetauscht.

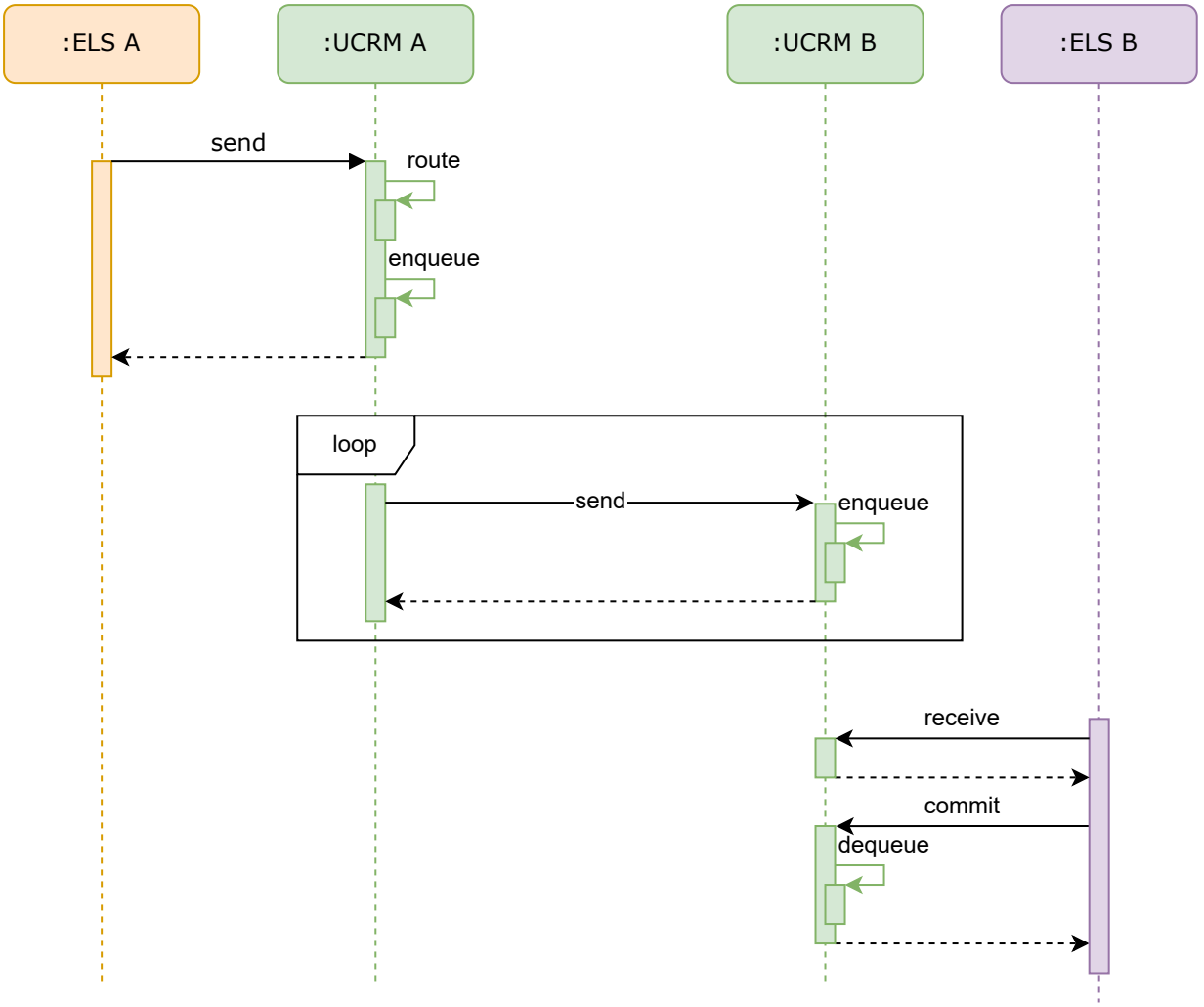
5.4.1. Verschlüsselung

Da jegliche Kommunikation zwischen KT und UCRM und zwischen UCRMs TLS verschlüsselt stattfindet, kann es auf eine E2E-Veschlüsselung des Nachrichteninhaltes verzichtet werden. Um die strengeren Sicherheitsanforderungen vor allem bei der Kommunikation über das Internet zu erfüllen, kann es später jedoch sinnvoll sein, den Nachrichteninhalt zusätzlich Ende-Zu-Ende, d.h. zwischen dem Sender-UCRM und dem Empfänger-UCRM oder sogar zwischen den KTs (in diesem Fall ohne Möglichkeit, die Meldungen durch UCRM validieren zu lassen), zu verschlüsseln.

Für die Verschlüsselung wird dann das RSA-Verfahren RSASSA-PKCS1-v1_5 (RFC 3447: Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1) verwendet.

Das konkrete Verschlüsselungsverfahren wird bei Bedarf später spezifiziert.

5.5. Zustellung von Nachrichten



Für die Zustellung von Nachrichten können unterschieldiche Routing-Algorithmen umgesetzt werden:

1. Routing durch Adressierungshierarchie
2. Routing-Tabellen

5.6. Validierung von Meldungen

Meldungen werden in UCRM m.H.v. Meldungs-Schemata validiert. Die Validierung erfolgt synchron beim Senden. KT-Register liefert Auskunft über die durch KT unterstützten Schemata.

6. UCRI2 APIs

In diesem Kapitel wird die konkrete Umsetzung der UCRI2-APIs beschrieben. Gemäß dem Rollenkonzept werden zwei APIs unterschieden:

1. Die **Client-API** wird vom **UCRM** angeboten und von **Clients** (Kommunikationsteilnehmern) konsumiert.
2. Die **P2P-API** wird von **UCRMs** angeboten, welche eine Verbindung mit anderen **UCRMs** herstellen wollen, falls dies durch die Kommunikationstopologie so vorgesehen ist.

Für beide APIs gelten folgende allgemeinen Festlegungen, welche in den jeweiligen Unterkapiteln mit API-spezifischen Festlegungen ergänzt werden:

1. Das grundlegende API-Design verwendet REST API Design Richtlinien erarbeitet bei TM Forum.
2. Die konkrete Ausgestaltung der APIs inklusive der zu verwendenden Endpunkte und Datenformate wird in den zu der Spezifikation gehörenden OPENAPI-Spezifikationen in Version 3.1.0 (festgelegt in OpenAPI Initiative) festgelegt (ucrm-client.yaml sowie ucrm-p2p.yaml). Die Spezifikationen verweisen jeweils auf die im Unterverzeichnis "schemas" vorhandenen yaml-Dateien mit den zu übertragenden Datenobjekten. Zusätzlich existieren noch inhaltsgleiche, gebündelte OPENAPI-Spezifikationen im JSON Format (ucrm-client-bundled.json sowie ucrm-p2p-bundled.json).
3. Beide APIs sind in folgende Funktionsbereiche aufgeteilt:
    1. Authentifizierung per OAUTH Client Credential Grant type.
    2. KT-Registry - Abfragen von Eigenschaften der registrierten Kommunikationsteilnehmer.
    3. Messaging - Versenden und Empfangen von Nachrichten.
    4. Info - Informationen über die Version und den Betreiber der Schnittstelle.
4. Die Datenübertragung erfolgt ausschliesslich per Transportverschlüsselung (TLS).
5. Die zur Übertragung von Fehlerzuständen genutzten Error-Objekte (vgl. error.yaml) werden im Unterkapitel "Fehlerbehandlung" für beide APIs gemeinsam dargestellt, da ein Großteil der genutzten Fehlercodes in beiden APIs vorkommt.
6. Die Generierung von Nachrichtensignaturen wird im Unterkapitel "Signierung von Nachrichten" beschrieben.
7. Falls ein Request nicht der OPENAPI-Spezifikation entspricht, MUSS dieser vom UCRM zurückgewiesen werden (vgl. Kapitel "Fehlerbehandlung").
8. Ein UCRM MUSS die **Client-API** vollständig unterstützen.
9. Ein UCRM, welches die **P2P-API** implementiert, MUSS diese vollständig unterstützen.

6.1. Client-API

Die Client-API dient der Anbindung von KT (clients) an die Transportschicht (UCRM). Im Folgenden werden die verschiedenen Endpunkte vorgestellt:

6.1.1. Endpunkt /token - HTTP GET

Über den Token-Endpunkt werden JSON Web Token (JWT) abgerufen, die bei sämtlichen anderen Endpunkten zur Authorisierung genutzt werden. Hierzu werden eine Nutzerkennung sowie ein Geheimnis per HTTP-Header übergeben.

6.1.2. Endpunkt /info - HTTP GET

Über den Info-Endpunkt können allgemeine Informationen über das UCRM wie die genutzte UCRI2-Transportschicht-Version, UCRM-Hersteller, Produktversion und Betriebsstatus des UCRM abgerufen werden.

6.1.3. Endpunkt /registry - HTTP GET

Über den Registry-Endpunkt können KT Informationen über alle dem UCRM bekannten Kommunikationsteilnehmer (inklusive den UCRMs selbst) abrufen. Die Datenfelder verändern sich hierbei im Normalfall nur selten, eine Ausnahme bildet das **status**-Feld. Dieses gibt für einen KT an, ob dieser aktuell erreichbar ist. Der Wert dieses Feldes verändert sich je nach von der Vermittlungsebene bzw. den beteiligten UCRMs festgestellten Verbindungsstatus relativ dynamisch (vgl. Kapitel !!!TODO Verfügbarkeits-Kapitel!!!). Da die Umgebung dynamisch ist, d.h. neue KT können dazukommen, existierende KT können abgebaut werden, sollen KT-Register-Abfragen regelmäßig durchgeführt werden (mindestens 1 Mal pro Stunde, maximal alle 5 Minuten).

6.1.4. Endpunkt /registry/{id} - HTTP GET

Über den Registry-ID-Endpunkt können KT Informationen über einen konkreten Kommunikationsteilnehmer abrufen, welcher die übergebene OID besitzt.

6.1.5. Endpunkt /messaging/send - HTTP POST

Über den Messaging-Send-Endpunkt können Nachrichten an andere KT versendet werden. Hierbei werden die Anwendungsnachrichten in eine sog. **Envelope** verpackt, welche Transportschicht-Metadaten enthält. Hierbei MUSS das UCRM folgende Prüfungen durchführen und im Fehlerfall einen passendes Error-Objekt zurückgeben:

1. Prüfung, ob der KT zum Senden der Nachricht berechtigt ist (Abgleich des source-Feldes mit dem übergebenen OAuth-Token).
2. Prüfung, ob der adressierte KT (destinations-Feld) bekannt ist.
3. Prüfung, ob die enthaltene Anwendungsnachricht (payload-Feld) eine bekannte Anwendung in einer bekannten Anwendungsversion referenziert und gemäß des Nachrichtenschemas gültig ist.
4. Prüfung, ob der adressierte KT (destinations-Feld) den Empfang der Anwendungsnachricht gemäß dessen KT-Registrierungs-Eintrages unterstützt.

Die Prüfung von Nachrichtensignaturen wird durch das UCRM für von clients gesendete Nachrichten NICHT durchgeführt, dies obliegt dem Empfänger der Nachricht.

Falls das UCRM eine unbekannte destination meldet, sollte der client über den /info-Endpunkt prüfen, ob das UCRM sich in einem normalen Betriebszustand befindet oder aktuell den initialen Abruf der KT-Registrierungsdaten von verbundenen UCRMs durchführt. In diesem Falle sollte abgewartet werden, bis das UCRM einen normalen Betriebszustand meldet.

6.1.5.1. Zustellungsquittierungen

Um die verlässliche Zustellung sicherzustellen, kann ein Client sowohl einen timeout-Wert für die Zustellung der Nachricht (**timeout**-Feld) setzen als auch eine explizite Empfangsbestätigung (**ack**-Feld) anfragen. In diesen Fällen überträgt das UCRM dem Client entweder eine negative Quittierung (falls ein Timeout aufgetreten ist oder bei der Weiterleitung der Nachricht innerhalb der Transportschicht ein Fehler aufgetreten ist) oder eine positive Quittierung (falls dies durch das **ack**-Feld angefragt wurde und der Empfänger den Empfang der Nachricht per Messaging-Commit-Endpunkt bestätigt hat). Hierzu kommt die **message_delivery_status**-Nachricht aus der **transport-layer-messages**-App zum Einsatz. Bezüglich dieser Statusbenachrichtigungen gelten zwei Festlegungen:

1. **message_delivery_status**-Nachrichten dürfen NUR durch UCRMs versendet werden, nicht durch clients. Dies muss durch das UCRM sichergestellt werden.
2. Ein UCRM darf für eine versendete Nachricht MAXIMAL eine **message_delivery_status**-Nachricht an den KT zurückmelden.

6.1.6. Endpunkt /messaging/receive - HTTP POST

Über den Messaging-Receive-Endpunkt können Nachrichten, die an den KT (bzw. eine Liste von OIDs im Feld **destinations**) adressiert sind, abgerufen werden. Diese Operation ist idempotent, kann also mehrmals mit dem gleichen Ergebnis wiederholt werden. Um den wiederholten Empfang einer Nachricht zu verhindern, MUSS der KT vor dem nächsten Abruf von Nachrichten die erfolgreich erhaltenen Nachrichten per Messaging-Commit-Endpunkt bestätigen.

Das UCRM MUSS prüfen, ob der Client zum Abruf der übergebenen OIDs berechtigt ist (durch Abgleich mit dem übergebenen OAuth-Token).

Das UCRM MUSS für jede empfangene Nachricht eine monoton steigende Sequenznummer im Feld **sequenceId** zurückgeben, die sich für eine individuelle Nachricht bei eventuellen mehrfachen Abrufen NICHT verändert.

6.1.6.1. Long-Polling

Da der Messaging-Receive-Endpunkt periodisch abgerufen werden muss, ist für einen schonenden Umgang mit Netzwerkresourcen und zur Sicherstellung möglichst geringer Zustellungsverzögerungen für diesen Endpunkt ein sog. Long-Polling vorgesehen. Dieses MUSS vom UCRM unterstützt werden. Folgende Schleife ist bei Nachrichtenabfragen mittels Long Polling zu verwenden:

1. Nachrichten abfragen mit Angabe maximaler Nachrichtenzahl nMax (Feld **maxMessages**)
2. Falls Nachrichten gemeldet wurden, Nachrichten verarbeiten

3. Falls Nachrichten gemeldet wurden, Nachrichtenempfang bestätigen

Wichtig: bei Programmausfällen zwischen Schritten 2 und 3 kann es zum wiederholten Empfang von gleichen Nachrichten kommen. Die Client-Logik MUSS somit mit Nachrichtenduplikaten umgehen können. Dies kann durch Berücksichtigung der Eindeutigkeit der Nachrichten-ID (Feld **messageId**) geschehen.

Durch die Verwendung von Long Polling kann direkt ohne Pause von Schritt 3 zu Schritt 1 übergegangen werden. Für das Long Polling gelten folgende Festlegungen:

1. Die maximale Verzögerung (dMax) der Antwort beträgt 30 Sekunden.
2. Falls für mindestens eines der angefragten Ziele (Feld **destinations**) mindestens eine unbestätigte Nachricht vorliegt, wird die Anfrage sofort beantwortet.
3. Ansonsten wird die Beantwortung der Client-Anfrage solange verzögert, bis eine der folgenden Bedingungen vorliegt (die Bedingungen sind in der genannen Reihenfolge zu prüfen), dann wird die Anfrage ohne weitere Verzögerung beantwortet:
    1. Für mindestens eines der angefragten Ziele (Feld **destinations**) liegt mindestens eine unbestätigte Nachricht vor.
    2. Die maximale Verzögerung (dMax) der Antwort wurde erreicht.
4. Falls beim Long Polling Probleme auftreten, die sich nicht auf anderem Wege lösen lassen (z.B. durch die clientseitige Verlängerung von TCP-Timeouts), kann der Client mittels des Feldes **maxDelay** ein verkürztes dMax angeben, welches dann vom UCRM anstelle des in 1. genannten dMax zu verwenden ist.

6.1.6.2. Verfügbarkeitsstatus

Der periodische Abruf des Messaging-Receive-Endpunkts dient dem UCRM zusätzlich als Indikator über die aktuelle Erreichbarkeit des Client. Falls ein Client den Messaging-Send-Endpunkt für 60 Sekunden (Verfügbarkeits-Timeout) nicht abruft, MUSS das UCRM den KT-Registrierungs-Verfügbarkeitsstatus (Feld **status**) auf "offline" setzen. Sobald wieder eine Anfrage eingeht, MUSS das UCRM den Verfügbarkeitsstatus auf "online" setzen. So können andere KT den aktuellen Verfügbarkeitsstatus via KT-Registrierung abrufen. Durch die Wahl des Verfügbarkeitsstatus-Timeout (60 Sekunden) auf das Doppelte der maximalen Long-Polling-Verzögerung (30 Sekunden) ist sichergestellt, dass beide Funktionen miteinander harmonieren. Um aus Sicht der Vermittlungsebene als verfügbar zu erscheinen, MUSS ein Client also eine permamanente Abfrage von Nachrichten durchführen (vgl. die im Kapitel "Long Polling" dargestellte Schleife).

6.1.7. Endpunkt /messaging/commit - HTTP POST

Über den Messaging-Commit-Endpunkt kann der erfolgreiche Empfang von Nachrichten für eine einzelne destination bestätigt werden. Hierzu wird eine Referenz-SequenzNummer angegeben (Feld **sequenceId**). Das UCRM entfernt dann alle Nachrichten mit Nachrichten-Sequenznummer <= Referenz-Sequenznummer aus der Liste unbestätigter Nachrichten und löst je nach Konfiguration des **ack**-Feldes die Übersendung einer Zustellbestätigung (**message-delivery-status**) aus. Falls ein Client Nachrichten für verschiedene OIDs abruft, MUSS für jede Nachricht ein separater Aufruf des Messaging-Commit-Endpunktes durchgeführt werden.

Diese Operation ist idempotent, kann also mehrmals mit dem gleichen Ergebnis wiederholt werden.

Das UCRM MUSS prüfen, ob der Client zur Bestätigung von Nachrichten an die übergebenen OIDs (Feld **destination**) berechtigt ist (durch Abgleich mit dem übergebenen OAuth-Token).

6.2. !!!TODO noch leer!!! Peer-to-Peer-API (P2P-API)

6.3. Fehlerbehandlung

Falls als Antwort auf eine Anfrage ein Fehler zurückgegeben wird, enthält dieser IMMER einen numerischen, UCRI2-spezifischen Fehlercode. Dieser Fehlercode ermöglicht es, die Fehlerursache zusätzlich zur groben Fehlereinteilung per HTTP-Statuscode genauer festzustellen. Für jeden Fehlercode wird eine feste Fehlercode-ID vergeben, z.B. für den Fehlercode 460 die Fehlercode-ID REQUEST_INVALID_PER_CLIENT_TRANSPORT_SPEC (s.u.).

Andere als die unten genannten Fehlercodes dürfen NICHT zurückgegeben werden.

Falls dies nicht explizit angegeben ist, können alle Fehlercodes sowohl von der Client- als auch von der P2P-Schnittstelle zurückgegeben werden.

Zusätzlich vom Fehlercode wird IMMER eine menschenlesbare Fehlermeldung im reason-Feld mitübergeben. Die zusätzliche Übermittlung einer detaillierten Fehlerursache im message-Feld ist dagegen freiwillig.

Für jeden Fehlercode wird an dieser Stelle dargestellt, in welchen Situationen dieser Fehlercode zurückgegeben wird. Die Fehlercodes sind dabei nach den HTTP-Statuscodes geordnet, in deren Kontext sie auftreten. Fehlercodes dürfen NUR im Kontext des aufgeführten HTTP-Statuscodes übertragen werden. Eine Ausnahme bildet hierbei die Übertragung eines NAK-Zustellfehlers im Rahmen der message_delivery_status-Nachricht in der transport_layer_messages-App, wo der Fehlercode, der von einem entfernten UCRM über das P2P-Schnittstelle gemeldet wurde an den Client weitergeleitet wird.

6.3.1. Fehlercodes für Antworten mit HTTP-Statuscode 400

6.3.1.1. Fehlercode 460 (REQUEST_INVALID_PER_CLIENT_TRANSPORT_SPEC)

Dieser Fehlercode wird NUR von der Client-Schnittstelle zurückgegeben, falls der Request zwar valide JSON-Daten enthält, diese aber das Client-Transportschicht-Schema für diesen Request verletzen. Der Fehlercode kommt NICHT zum Einsatz, wenn ein enthaltener App-Payload das App-Schema verletzt (vgl. REQUEST_PAYLOAD_INVALID_PER_APP_SPEC).

6.3.1.2. Fehlercode 461 (REQUEST_PAYLOAD_UNKNOWN_APPID)

Dieser Fehlercode wird zurückgegeben, falls die angegebene AppId dem UCRM unbekannt ist.

6.3.1.3. Fehlercode 462 (REQUEST_PAYLOAD_UNKNOWN_APPVERSION)

Dieser Fehlercode wird zurückgegeben, falls dem UCRM zwar die AppId bekannt ist, aber die angegebene AppVersion unbekannt ist.

6.3.1.4. Fehlercode 463 (REQUEST_PAYLOAD_UNKNOWN_SCHEMAID)

Dieser Fehlercode wird zurückgegeben, falls dem UCRM zwar die AppId und AppVersion bekannt ist, aber die angegebene SchemaId (also der Nachrichtentyp) unbekannt ist.

6.3.1.5. Fehlercode 464 (REQUEST_PAYLOAD_INVALID_PER_APP_SPEC)

Dieser Fehlercode wird zurückgegeben, falls der Request-Payload im data-Feld das zugehörige App-Schema verletzt.

6.3.1.6. Fehlercode 465 (REQUEST_PAYLOAD_INVALID_JSON)

Dieser Fehlercode wird zurückgegeben, falls es sich bei den übergebenen Daten nicht um gültiges JSON handelt.

6.3.1.7. Fehlercode 466 (REQUEST_PAYLOAD_UNSUPPORTED_APPID_OR_APPVERSION)

Dieser Fehlercode wird zurückgegeben, falls die appId und appVersion dem UCRM zwar bekannt, aber nicht in den supportedApps der destination als unterstützt aufgeführt sind.

6.3.1.8. Fehlercode 468 (REQUEST_PAYLOAD_UNSUPPORTED_MESSAGE)

Dieser Fehlercode wird zurückgegeben, falls die appId und appVersion dem UCRM bekannt und auch in den supportedApps der destination als unterstützt aufgeführt sind, die schemaId (also der Nachrichtentyp) aber im entsprechenden supportedApps-Eintrag in der liste der unsupportedMessages aufgeführt ist.

6.3.1.9. Fehlercode 467 (REQUEST_PAYLOAD_FORBIDDEN_APPID)

Dieser Fehlercode wird NUR von der Client-Schnittstelle zurückgegeben, falls die gesendete Nachricht zur transport_layer_messages-App gehört (Nachrichten dieser App dürfen nur durch UCRMs versendet werden).

6.3.1.10. Fehlercode 470 (REQUEST_UNKNOWN_DESTINATION_ID)

Dieser Fehlercode wird zurückgegeben, wenn die angegebene destination dem UCRM nicht bekannt ist.

6.3.1.11. Fehlercode 478 (REQUEST_OID_FORBIDDEN)

Dieser Fehlercode wird zurückgegeben, falls der durch das übergebene OAuth-Token identifizierte Benutzer keinen Zugriff auf die angegebene OID hat. Bei der angegebene OID handelt es sich um die OID:

1. im sender-Feld (für den /messaging/sent-Endpunkt)
2. im destinations-Feld (für den /messaging/receive-Endpunkt)
3. im destination-Feld (für den /messaging/commit-Endpunkt)

6.3.1.12. Fehlercode 479 (REQUEST_WRONG_SIGNATURE)

Dieser Fehlercode wird NUR von der P2P-Schnittstelle zurückgegeben, falls es sich bei der gesendeten Nachricht um eine signierte Nachricht aus der transport_layer_messages-App handelt (und das sendende UCRM nicht per transmitsUnsignedMessages in seinem KT-Registrierungseintrag festgelegt hat, dass dieses seine Nachrichten nicht signiert). Auf der Client-Schnittstelle und generell für von Clients gesendete Nachrichten findet KEINE Signaturprüfung in der Transportschicht statt.

6.3.1.13. Fehlercode 480 (REQUEST_INVALID_PER_P2P_TRANSPORT_SPEC)

Dieser Fehlercode wird NUR von der P2P-Schnittstelle zurückgegeben, falls der Request zwar valide JSON-Daten enthält, diese aber das P2P-Transportschicht-Schema für diesen Request verletzen. Der Fehlercode kommt NICHT zum Einsatz, wenn ein enthaltener App-Payload das App-Schema verletzt (vgl. REQUEST_PAYLOAD_INVALID_PER_APP_SPEC).

6.3.2. Fehlercodes für Antworten mit HTTP-Statuscode 401

6.3.2.1. Fehlercode 475 (REQUEST_UNAUTHORIZED)

Dieser Fehlercode wird zurückgegeben, falls das übergebene OAuth-Token fehlt, ungültig oder expired ist. Außerdem wird der Fehlercode durch den /token-Endpunkt zurückgegeben, falls die übergebenen Zugangsdaten nicht korrekt sind.

6.3.3. Fehlercodes für Antworten mit HTTP-Statuscode 500

6.3.3.1. Fehlercode 491 (REQUEST_INTERNAL_ERROR)

Dieser Fehlercode wird zurückgegeben, falls ein interner Fehler aufgetreten ist.

6.4. Signierung von Nachrichten

Nachrichtensignaturen stellen sicher, dass eine Nachricht während der Übertragung nicht verändert wurde und dass sie von der angegebenen Quelle stammt. Hierzu sind zwei Schritte notwendig: Die Erzeugung eines eindeutigen Hash-Wertes für die Nachricht sowie die Signierung dieses Hashwertes durch das sendende System:

1. Für das Hashing wird die Nachricht zuerst gemäß JSON Canonicalization Scheme (JCS, RFC 8785: JSON Canonicalization Scheme (JCS)) in eine kanonische Form gebracht und diese kanonische Form dann mit SHA3-256 gehasht.
2. Für das Signieren und die Prüfung der Signatur wird das Verfahren nach dem IETF Standard JSON Web Signature (JWS, RFC 7517: JSON Web Key (JWK)) in der Variante Compact JWS in Kombination mit dem RSA-Verfahren RSASSA-PKCS1-v1_5 verwendet.

Diese Schritte werden in den folgenden Unterkapiteln genauer dargestellt.

6.4.1. Hashing der Nachrichten

Vor der Anwendung des JCS erstellt das signierende System eine Kopie der Nachricht, in der nur die folgenden Felder enthalten sind:

1. "source"
2. "destinations"

3. "payload"

Somit werden alle anderen Felder nicht beim Hashing berücksichtigt. Dies ist notwendig, da andere Felder freiwillig sind oder vom UCRM gesetzt werden, falls sie nicht vom Client gesetzt wurden. Als Hashing-Algorithmus kommt SHA3-256 zum Einsatz (Use of the SHA3 One-way Hash Functions in the Cryptographic Message Syntax (CMS)).

6.4.2. Signierung der Nachrichten

Als JWS-Header kommt ein Header mit Angabe des RSA-256-Algorithmus zum Einsatz:

```
{
  "typ": "UCRI_PLAIN",
  "alg": "RS256"
}
```

Der berechnete Hashwert der Nachricht wird als JWS-Payload verwendet. Dieser wird mit dem privaten Schlüssel des Nachrichtensenders signiert. Das Ergebnis (digitale Signatur) wird in Form einer JSON Web Signature (JWS, RFC 7517: JSON Web Key (JWK)) im Feld **signatur** übertragen:

```
BASE64(Header) || '.' || BASE64(Hash) || '.' || BASE64(Signatur)
```

6.4.3. Prüfung der Signaturen

Um die übermittelte JWS zu prüfen, MUSS das empfangende System wiederum

1. den Hashwert der empfangenen Nachricht berechnen
2. die übermittelte Signatur auf Gültigkeit prüfen
3. den signierten Hashwert (JWS-Payload) auf Gleichheit mit dem Hashwert der empfangenen Nachrichten prüfen

Die Ermittlung des Hashwertes erfolgt gemäß "Hashing der Nachrichten".

Die Signatur wird aus dem "signature"-Feld der Nachricht extrahiert und gegen den im KT-Register hinterlegten "key" (öffentlichen Schlüssel) validiert.

7. UCRI2 Anwendungen

Detaillierte Beschreibung der UCRI2 Anwendungen befindet sich in dem Verzeichnis docs/apps. Hierbei gibt das Dokument "UCRI2 App im Überblick" einen Überblick über die verschiedenen Apps und die PDF-Dateien in den Unterordnern beinhalten die detaillierte Dokumentation für verschiedenen Apps.