OParl Vokabular und Schnittstellen-Spezifikation (Entwurf)

OParl Team - http://oparl.org/

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument wird bei seiner Fertigstellung die Spezifikation des OParl Schnittstellen-Standards für parlamentarische Informationssysteme[1] darstellen. Es dient damit als Grundlage für die Implementierung von OParl-konformen Server- und Clientanwendungen.

1.1 Status

Die Spezifikation befindet sich in Arbeit. Das Dokument enthält entsprechend viele Ungenauigkeiten und Hinweise auf offene Fragestellungen.

Die Kennzeichnung "TODO" markiert Hinweise, die für die Bearbeiter des Dokuments gedacht sind. Darunter befinden sich auch Fragen an deren Beantwortung durch Reviewer des Dokuments die Autoren interessiert sind.

1.2 Was ist OParl in Kürze?

OParl ist die Gruppierung, die Initiator und Herausgeber der vorliegenden Spezifikation ist. An OParl wirken Verbände, Zivilgesellschaftliche Organisationen und Initiativen und Software-Anbieter sowie interessierte Einzelpersonen mit.

Die vorliegende Spezifikation beschreibt den OParl-Standard. Dieser definiert eine Webservice-Schnittstelle, die den anonymen und lesenden Zugriff auf öffentliche Inhalte aus parlamentarischen Informationssystemen ermöglicht. Wie der Name "Webservice" ausdrückt, setzt diese Schnittstelle auf dem World Wide Web auf. Sie ermöglicht, dass parlamentarishe Informationen maschinenlesbar als Offene Daten (Open Data) veröffentlicht werden.

Die vorligende Version ist die erste verabschiedete Version der Spezifikaiton zum OParl-Standard.

1.3 Zielsetzung von OParl

OParl richtet sich an verschiedene Nutzergruppen und Stakeholder:

- Verwaltung und politische Gremien in Kommunen
- Bürger, politische Parteien und Organisationen
- Open Data Initiativen
- Wissenschaftler
- Anbieter von Server- und Softwareprodukten
- Anbieter von Linked Data-Plattformen oder -Services

TODO: Nutzen für jede Stakeholder-Gruppe

TODO: Linked Data erwähnen oder Verweis auf Linked Data Abschnitt

Die Gründe, warum Betreiber von parlamentarischen Informationssystemen den Zugriff darauf über eine standardisierte Schnittstelle ermöglichen sollten, können vielfältig sein.

[1] In Deutschland hat sich auf kommunaler Ebene der Begriff "Ratsinformationssystem" etabliert. OParl ist in seiner Anwendung jedoch nicht auf Gemeinderäte eingeschränkt und verwendet daher den Begriff "parlamentarisches Informationssystem".

Ein zentrales Argument ist die Verpflichtung der Parlamente gegenüber der Bevölkerung, diese über die Fortschritte der parlamentarischen Arbeit zu informieren und auf dem Laufenden zu halten. Ein erster Schritt, der Bevölkerung Einblicke in die Arbeit und Zugriff auf Dokumente zu gewähren, ist vielerorts in den letzten Jahren durch Einführung von Ratsinformationssystemen mit anonymem, lesenden Zugriff über das World Wide Web gemacht worden.

Die damit eingeschlagene Richtung konsequent weiter zu gehen, bedeutet, die Daten der parlamentarischen Informationssystemen gänzlich offen zu legen, sofern die Inhalte es erlauben. Es bedeutet, die Daten und Inhalte so universell weiterverwendbar und so barrierearm wie möglich anzubieten, dass jegliche weitere Verwendung durch Dritte technisch möglich ist. Der seit einiger Zeit etablierte Begriff für dieses Prinzip heißt "Open Data".

Das Interesse an parlamentarischen Informationen und an Anwendungen, die diese nutzbar und auswertbar machen, ist offensichtlich vorhanden. Die Entwickler der alternativen Ratsinformationssysteme wie Frankfurt Gestalten[2], Offenes Köln[3] oder der OpenRuhr:RIS-Instanzen[4] wissen zu berichten, wie viel Interesse den Projekten gerade aus Orten entgegen gebracht wird, in denen derartige Systeme noch nicht verfügbar sind.

Die Anwendungsmöglichkeiten für parlamentarische Informationen, wenn sie über eine Schnittstelle schnell und einfach abgerufen werden können, sind vielfältig. Beispiele sind:

- Apps für den Abruf auf mobilen Endgeräten
- Möglichkeiten zur Wiedergabe für Nutzerinnen und Nutzer mit Beeinträchtigung des Sehvermögens
- Alternative und erweiterte Suchmöglichkeiten in Inhalten
- Auswertung und Analyse von Themen, Inhalten, Sprache etc.
- Benachrichtigungsfunktionen beim Erscheinen bestimmte Inhalte

Die Standardisierung dieses Zugriffs über die Grenzen einzelner Systeme hinweg erlaubt zudem, diese Entwicklungen grenzüberschreitend zu denken. Damit steigt nicht nur die potenzielle Nutzerschaft einzelner Entwicklungen. Auch das Potenzial für Kooperationen zwischen Anwendungsentwicklern wächst.

Darüber hinaus sind auch Motivationen innerhalb von Organisationen und Körperschaften erkennbar. So sollen parlamentarische Informationssysteme vielerorts in verschiedenste Prozesse und heterogene Systemlandschaften integriert werden. Durch eine einheitliche Schnittstelle bieten sich effiziente Möglichkeiten zur Integration der Daten in anderen Systeme, wie beispielsweise Web-Portale.

Ausführlichere Beschreibungen einiger möglicher Anwendungsszenarien finden sich im Kapitel Nutzungsszenarien.

1.4 Transparenz und Beteiligung durch Open Data

Öffentliche Stellen verfügen über vielfältige Informationen und Daten. Seit einigen Jahren sind zivilgesellschaftliche Organisationen sowie Politik und Verwaltung unter dem Schlagwort "Open Data" international und auch in Deutschland in unterschiedlichem Maße um eine stärkere Öffnung dieser Daten bemüht[5]. Bei dem Ansatz "Open Data"[6] geht es darum diese Daten so bereitzustellen, dass Dritte diese einfacher finden und weiterverwenden können. Die zehn Open-Data-Prinzipien der Sunlight-Foundation[7] beschreiben die Offenheit von Datensätzen. Wesentlich dabei sind vor allem die einfache rechtliche und die technische

[2] Frankfurt Gestalten: http://www.frankfurt-gestalten.de/
[3] Offenes Köln: http://offeneskoeln.de/
[4] OpenRuhr:RIS: http://openruhr.de/openruhrris/
[5] Eine weltweite Übersicht zu Open-Data-Projekten bietet z.B. der Open-Data-Showroom http://opendata-showroom.org/de/
[6] vgl. https://de.wikipedia.org/wiki/Open_data
[7] Ten Principles for Opening Up Open Government Information, https://sunlightfoundation.com/policy/documents/ten-open-data-principles

Offenheit. Bei ersterer geht es darum, dass Datensätze unter Nutzungsbestimmungen bereitgestellt werden, die kurz und verständlich formuliert sind und mindestens jegliche weitere Verwendung inklusive der kommerziellen erlauben unter der Voraussetzung, dass bei der Weiterverwendung die Quelle benannt wird. Bei der technischen Offenheit steht die Bereitstellung von Datensätzen in möglichst maschinenlesbaren Formaten im Vordergrund. Dies bedeutet, stärker strukturierte Datensätze sind in der Bereitstellung zu bevorzugen. Liegen Daten innerhalb einer Organisation in einer Datenbank vor, so bietet es sich an, diese soweit möglich über eine Programmierschnittstelle (API) für Außenstehende bereitzustellen.

Die Erfüllung dieser rechtlichen und technischen Offenheit erlaubt es Dritten, dies können Bürgerinnen und Bürger, Unternehmen, Forschungseinrichtungen oder auch andere Verwaltungseinheiten sein, die Verwaltungsdaten wesentlich unkomplizierter für eigene Vorhaben wie Anwendungen oder Visualisierungen einzusetzen. Mit dem Ansatz offener Verwaltungsdaten soll so erstens mehr Transparenz über Prozesse und Entscheidungen in Politik und Verwaltung erreicht werden. Zweitens können Dritte auf Grundlage dieser Daten leichter eigene Geschäftsmodelle verfeinern oder neue entwickeln. Drittens wird es auch öffentlichen Stellen selbst leichter bereits im öffentlichen Sektor existierende Daten zu finden und weiterzuverwenden.

Wie das Prinzip offener Daten bzw. offener Verwaltungsdaten über die Minimalprinzipien rechtlicher und technischer Offenheit hinaus am besten erreicht werden kann erfordert im Einzelfall häufig eine Zusammenarbeit von Datenbereitstellern und potentiellen Datennutzern. Die bloße Bereitstellung einer OParl-konformen API wird weder die Einhaltung der technischen Prinzipien, noch der weiteren Open-Data-Prinzipien vollständig garantieren. Viele Bestandteile der OParl Spezifikation, die einen weitgehend barrierearmen Zugang zu Informationen ermöglichen, sind optional (Beispiel: Volltexte von Dokumenten über die API abrufbar machen). Andere Bestandteile, die von Interesse wären, sind noch gar nicht von OParl abgedeckt (Beispiel: Abstimmungsergebnisse). Grund dafür ist, dass sich OParl in einem frühen Stadium befindet und primär am Status Quo der parlamentarischen Informationssysteme ausgerichtet ist. Es liegt also auch weiterhin an Verwaltung und Politik, durch einen verantwortungsvollen Umgang mit den Systemen die maximal erreichbare Transparenz zu bieten. Das fängt bei Dokumentenformaten an (ein PDF mit digitalem Text weist weit weniger Barrieren auf, als ein gescannter Brief, der ebenfalls als PDF gespeichert wurde) und hört bei der verwendeten Sprache auf.[8]

1.5 Werdegang von OParl 1.0

Stichpunkte:

- 17. und 18. November 2012: Die Open Knowledge Foundation Deutschland veranstaltet in den Räumen der Heinrich-Böll-Stiftung in Berlin einen Workshop für Entwickler von Anwendungen, die einen gesellschaftlichen Nutzen bringen sollen. Hier ist VITAKO, die Bundes-Arbeitsgemeinschaft der Kommunalen IT-Dienstleister, als Sponsor engagiert. Die Geschäftsführerin, Dr. Marianne Wulff, ist persönlich vor Ort. Auch das Projekt Offenes Köln wird in einem Vortrag von Marian Steinbach präsentiert. Es kommt zum Austausch über die Frage, wie das Prinzip der offenen Ratsinformationen effektiv auf weitere Kommunen ausgeweitet werden könnte.
- 6. Dezember 2012: Anhörung im Landtag NRW in Düsseldorf zu einer Open-Data-Strategie der Landesregierung, wo Jens Klessmann und Marian Steinbach als Sachverständige gehört werden. Danach Gespräch über Möglichkeiten der Standardisierung offener Ratsinformationssysteme.

[8] Weitere generelle Informationen zur Bereitstellung offener Verwaltungsdaten bieten bspw.

- Praktische Informationen: Open-Data-Handbook der Open Knowledge Foundation http://opendatahandbook.org/de/how-to-open-up-data/index.html
- Grundsätzliche Informationen: Die vom Bundesministerium des Innern beauftragte Studie "Open Government Data Deutschland" http://www.bmi.bund.de/SharedDocs/Downloads/DE/Themen/OED_Verwaltung/ModerneVerwaltung/opengovernment.pdf

- Dezember 2012: Dr. Marianne Wulff, Jens Klessmann und Marian Steinbach beginnen mit der Abstimmung über einen Workshop mit Vertreterinnen und Vertretern von Kommunen, kommunalen IT-Dienstleistern, RIS-Anbietern und Zivilgesellschaft. Ziel: Die Bereitschaft zur Zusammenarbeit an einem gemeinsamen Standard ermitteln. Unterdessen beginnt Marian Steinbach mit der Formulierung eines Standard-Entwurfs als Diskussionsgrundlage. Der Entwurf wird von Beginn an öffentlich auf GitHub.com bereit gestellt.

- 17. April 2013: Insgesamt 30 Teilnehmer versammeln sich in Köln, um sich in einem ersten Treffen über Ziele und Chancen einer Standardisierung für offene Ratsinformationen auszutauschen. Als Ergebnis wird ein großes Interesse an der weiteren Zusammenarbeit auf Basis des vorliegenden Standardentwurfs festgestellt. Als Termin für die Fertigstellung der ersten Version der Spezifikation wird der 30. Juni 2013 festgelegt. Die Initiatoren präsentieren den Anwesenden hier erstmals den Namen "OParl", der künftig als Marke für die Bemühungen der Gruppe stehen soll.

- 22. Januar 2014: Nachdem sich die verteilte Zusammenarbeit am Standard-Entwurf seit April 2013 als nicht zielführend erwiesen hat, laden Jens Klessmann und Marian Steinbach und VITAKO zu einem eintägigen OParl-Workshop in Bielefeld ein. Das Ziel ist, die Spezifikation so weit wie möglich voran zu treiben und eine gute Basis für die baldige Fertigstellung zu legen.

- 26. Januar 2014: In Düsseldorf findet ein weiterer Workshop zur Arbeit am Entwurf der Spezifikation statt.

- April 2014: Verfeinerung des Vokabular-Teils durch Andreas Kuckartz, finanziert durch das FP7-Projekt Fusepool aus Mitteln der Europäischen Union.

- 6. Mai 2014: Beginn der Review-Phase. Interessierte sind aufgerufen, den vorliegenden Entwurf bis Ende Mai zu kommentieren.

- Ende Mai 2014: Telefonkonferenz zum eingegangenen Feedback aus der Review-Phase

- KW 23 (2. bis 6. Juni): Geplante Veröffentlichung der Spezifikation 1.0

1.6 Zukunft von OParl

TODO: - Verfeinerung, Lücken schliessen - Erweiterung über die kommunale Ebene hinaus (Land, Bund) - Vereinheitlichung von Kategorien (Drucksachentypen, Arten von Gremien) - Erweiterung von Personendaten, z.B. mit Social Media URLs - Mehr Abfragekriterien - Suchfunktionen (Volltextsuche) - Abstimmungsverhalten und maschinenlesbare Protokolle - Verknüpfung mit verteilten Social Media Plattformen - Schreibender Zugriff. Auch dazu muss das Rad nicht neu erfunden werden. Bestehende bzw. in Entwicklung befindliche Spezifikationen und Techniken aus der Linked Data-Welt können verwendet werden. Dazu gehören insbesondere die Linked Data Platform des W3C und Hydra.

1.6.1 Globalisierung

Es gibt in sehr vielen Ländern Gebietskörperschaften mit politischen Gremien, deren Prozesse ähnlich strukturiert sind wie diejenigen in Deutschland. Auch dort besteht Bedarf an standardisierten Vokabularen zur Veröffentlichung parlamentarischer Informationen. Deshalb sind - teilweise noch vor OParl - auch weitere entsprechende Initiativen entstanden. Eine enge Zusammenarbeit mit diesen Initiativen mit dem Ziel der Wiederverwendung von Arbeitsergebnissen wird deshalb angestrebt. Auch aus diesem Grund wurde bereits in OParl 1.0 die Möglichkeit der Verwendung mit anderen Sprachen und Mehrsprachigkeit eingebaut.

TODO: Popolo, UK, KB Niederlande

1.7 Nomenklatur der Spezifikation und Satzkonventionen

1.7.1 Zwingende, empfohlene und optionale Anforderungen

Dieses Spezifikationsdokument nutzt die Modalverben müssen, können und sollen in einer Art und Weise, die bestimmte Anforderungen möglichst unmissverständlich in drei verschiedene Abstufung einteilen lässt. Um ihre normative Bedeutung zu unterstreichen, werden diese Wörter grundsätzlich in Großbuchstaben gesetzt.

Diese Konvention ist angelehnt an die Definitionen der Begriffe MUST, SHOULD und MAY (bzw. MUST NOT, SHOULD NOT und MAY NOT) aus RFC2119.[9]

Die Bedeutung im Einzelnen:

MÜSSEN/MUSS bzw. ZWINGEND: Die Erfüllung einer Anforderung, die explizit vom Modalverb MÜSSEN bzw. MUSS Gebrauch macht, ist zwingend erforderlich.

Die Entsprechung in RFC2119 lautet “MUST”, “REQUIRED” oder “SHALL”.

NICHT DÜRFEN/DARF NICHT: Dieses Stichwort kennzeichnet ein absolutes Verbot.

Die Entsprechung in RFC2119 lautet “MUST NOT” oder “SHALL NOT”.

SOLLEN/SOLL bzw. EMPFOHLEN: Mit dem Wort SOLLEN bzw. SOLL sind empfohlene Anforderungen gekennzeichnet, die von jeder Implementierung erfüllt werden sollen. Eine Nichterfüllung ist als Nachteil zu verstehen, beispielsweise weil die Nutzerfreundlichkeit dadurch Einbußen erleidet, und sollte daher sorgfältig abgewogen werden.

Die Entsprechung in RFC2119 lautet “SHOULD” oder “RECOMMENDED”.

NICHT SOLLEN/SOLL NICHT bzw. NICHT EMPFOHLEN: Diese Formulierung wird verwendet, wenn unter gewissen Umständen Gründe existieren können, die ein bestimmtes Verhalten akzeptabel oder sogar nützlich erscheinen lassen, jedoch die Auswirkung des Verhaltens vor einer entsprechenden Implementierung verstanden und abgewogen werden sollen.

Die Entsprechung in RFC2119 lautet “SHOULD NOT” oder “NOT RECOMMENDED”.

DÜRFEN/DARF bzw. OPTIONAL: Mit dem Wort DÜRFEN bzw. DARF oder OPTIONAL sind optionale Bestandteile gekennzeichnet. Ein Anbieter könnte sich entscheiden, den entsprechenden Bestandteil aufgrund besonderer Kundenanforderungen zu unterstützen, während andere diesen Bestandteil ignorieren könnten. Implementierer von Clients oder Servern DÜRFEN in solchen Fällen NICHT davon ausgehen, dass der jeweilige Kommunikationspartner den entsprechenden, optionalen Anteil unterstützt.

Die Entsprechung in RFC2119 lautet “MAY” oder “OPTIONAL”.

1.7.2 Geschlechterspezifische Begrifflichkeiten

Um bei Begriffen wie Nutzer, Anwender, Betreiber etc. die sonst übliche Dominanz der männlichen Variante zu vermeiden, werden in diesem Dokument männliche oder weibliche Varianten gemischt. Es wird also beispielsweise mal von einer Nutzerin gesprochen und mal von einem Nutzer. Gemeint sind selbstverständlich in allen Fällen immer weibliche wie auch männliche Personen.

[9] RFC2119 http://tools.ietf.org/html/rfc2119

1.7.3 Codebeispiele

Die in diesem Dokument aufgeführten Codebeispiele dienen der Veranschaulichung der beschriebenen Prinzipien. Es handelt sich in der Regel um frei erfundene Daten.

Codebeispiele erheben insbesondere bei JSON-Code nicht den Anspruch auf hundertprozentige syntaktische Korrektheit. Es kann in Einzelfällen aus Gründen der besseren Lesbarkeit zur Verwendung von Umlauten etc. kommen, die in JSON nicht erlaubt wären.

1.8 Initiatoren

TODO

1.9 Unterstützer

TODO

1.10 Autoren

An diesem Dokument haben mitgewirkt:

Felix Ebert, Jan Erhardt, Jens Klessmann, Andreas Kuckartz, Babett Schalitz, Ralph Sternberg, Marian Steinbach, Bernd Thiem, Thomas Tursics, Jakob Voss

2 Architektur

In diesem Abschnitt werden grundlegenden Konzepte, die von OParl abgedeckt werden, erläutert. Die Erläuterungen sind nicht im engeren Sinne Teil der Spezifikation, sondern dienen dazu, die Anwendungsbereiche von OParl und die Funktionen einer OParl-konformen API verständlicher und konkreter beschreiben zu können.

Da die Architektur auf der generellen Architektur des World Wide Web (WWW) aufbaut, sind einzelne Konzepte direkt den Begriffen der Architekturbeschreibung des W3-Konsortiums entlehnt.[10]

2.1 Überblick

2.2 Parlamentarisches Informationssystem

Parlamentarische Informationssysteme sind Software-Systeme, die von verschiedensten Körperschaften eingesetzt werden, um die Zusammenarbeit von Parlamenten zu organisieren, zu dokumentieren und öffentlich nachvollziehbar zu machen. Zu den Körperschaften können beispielsweise Kommunen, Landkreise, Regierungsbezirke und Zweckverbänden gehören.

Diese Systeme unterstützen in der Regel mehrere der folgenden Funktionen:

- Das Erzeugen, Bearbeiten und Darstellen von Sitzungen und deren Tagesordnung
- Das Erzeugen und Abrufen von Sitzungsprotokollen
- Das Erzeugen, Bearbeiten und Anzeigen von Drucksachen
- Das Erzeugen, Bearbeiten und Anzeigen von Gremien und deren Mitgliedern

[10] Architecture of the World Wide Web, Volume One. http://www.w3.org/TR/webarch/

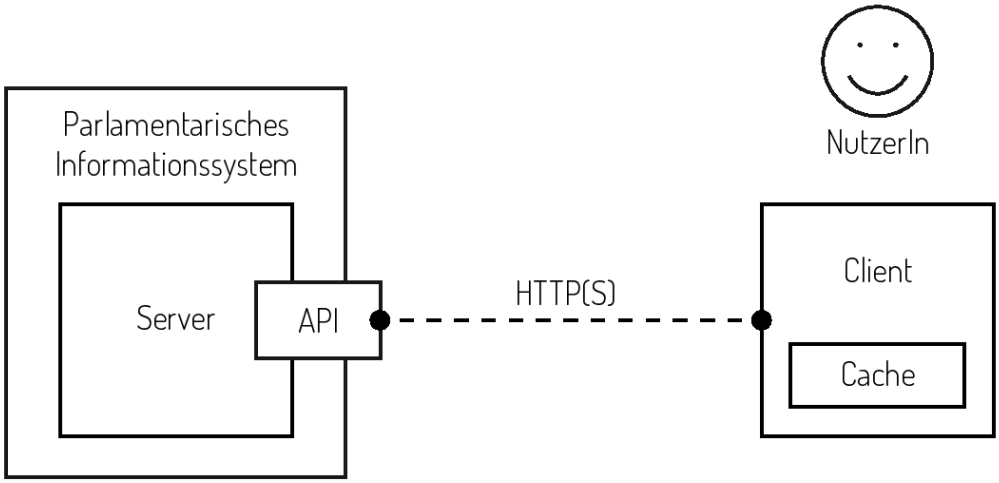

Abbildung 1: Architekturdiagramm

Funktionen, die die Eingabe und Bearbeitung von Daten betreffen, sind in der Regel einem geschlossenen Nutzerkreis vorbehalten. Die Darstellung und der Abruf von Informationen und Dokumenten hingegen ist in vielen Fällen für die Öffentlichkeit freigegeben.

Die OParl Spezifikation beschreibt eine Schnittstelle, die den maschinellen, lesenden Zugriff auf derartige Informationen ermöglicht.

2.3 Server

Der Server im Sinne dieser Spezifikation ist ein Software-Dienst, der auf einem mit dem Internet verbundenen Rechnersystem läuft. Dieser Dienst ist eine spezielle Form eines WWW- bzw. HTTP(S)-Servers. Entsprechend beantwortet der Server HTTP-Anfragen, die an ihn auf einem bestimmten TCP-Port gestellt werden.

Der Server ist als Bestandteil des parlamentarischen Informationssystems zu verstehen. Der Betrieb des Servers steht damit üblicherweise in der Verantwortung desjenigen, der das parlamentarischen Informationssystem betreibt.

Von einem Server, der die OParl-Spezifikation erfüllt, wird erwartet, dass er bestimmte parlamentarische Informationen in einem bestimmten Format zur Verfügung stellt und auf bestimmte Anfragen von so genannten Clients über die OParl API entsprechend dieser Spezifikation reagiert.

2.4 API

Der Begriff API steht in diesem Dokument für die Webservice-Schnittstelle, die der Server anbietet. Die Schnittstelle basiert auf dem HTTP-Protokoll. Mittels HTTPS ist wahlweise auch die verschlüsselte Nutzung der API möglich, sofern Server dies unterstützt.

Die API steht im Mittelpunkt dieser Spezifikation. Server und Clients sind als Kommunikationspartner zu verstehen, die über das Internet als Kommunikationskanal mit einander kommunizieren können. Die API-Spezifikation stellt dabei die nötige Grammatik und das Vokabular bereit, anhand dessen eine sinnvolle Kommunikation erfolgen kann.

2.5 Client

Der Begriff “Client” steht für eine Software, die über die OParl API mit dem Server kommuniziert. Da die API auf dem HTTP-Protokoll aufbaut, handelt es sich bei dem Client

um eine spezielle Form eines HTTP-Clients.

2.6 Cache

Ein Cache ist ein Speicher, der einem Client dazu dienen kann, von einem Server abgerufene Informationen längerfristig vorzuhalten. Dies kann beispielsweise dazu dienen, mehrfache Anfragen der selben Informationen zu vermeiden, wodurch sowohl Ressourcen auf Seite des Servers geschohnt als auch die Nutzung von Netzwerkbandbreite reduziert werden kann. Die Nutzung eines Cache kann auch zur Verbesserung der Nutzerfreundlichkeit eines Clients beitragen, indem Wartezeiten zur Bereitstellung einer Ressource verkürzt werden.

2.7 Nutzerin oder Nutzer

Mit einer Nutzerin oder einem Nutzer ist in diesem Fall eine natürliche Person gemeint, die mittels eines OParl-Clients auf parlamentarische Informationen zugreift.

2.8 Objekt

Der Server beantwortet Anfragen eines Clients im Regelfall, indem bestimmte Objekte ausgegeben werden. Objekte sind im Fall einer OParl-konformen API JSON-Objekte, die das Schema einhalten, das in der vorliegenden Spezifikation beaschrieben wird. Antworten des Servers können einzelne Objekte, Listen von Objekten oder Listen von URLs von Objekten enthalten.

3 Nutzungsszenarien

Die nachfolgenden Nutzungsszenarien dienen dazu, die Architektur und die Anwendungsmöglichkeiten anhand konkreter Beispiele zu verdeutlichen. Sie erheben keinen Anspruch auf Vollständigkeit.

3.1 Szenario 1: Mobile Client-Anwendung

Eine Client-Anwendung für mobile Endgeräte wie Smartphones und Tablets, nachfolgend "App" genannt, könnte das Ziel verfolgen, Nutzern unterwegs sowie abseits vom Desktop-PC auf die Gegebenheiten mobiler Endgeräte optimierten Lesezugriff auf Dokumente aus parlamentarischen Informationssystemen zu bieten. Die möglichen Kontexte und Nutzungsmotivationen sind vielfältig:

- Teilnehmer einer Sitzung greifen während der Sitzung auf die Einladung dieser Sitzung und die zur Tagesordnung der Sitzung gehörenden Drucksachen zu, außerdem auf die Protokolle vorheriger Sitzungen.
- Eine Redakteurin der Lokalpresse geht unterwegs die Themen der nächsten Sitzungen bestimmter Gremien, für die sie sich besonders interessiert, durch.
- Eine Gruppe von Studierenden erkundet zusammen mit ihrem Dozenten die lokalpolitischen Aktivitäten des Viertels rund um ihre Hochschule. Dazu nutzen sie die GPS-Lokalisierung ihrer Smartphones in Verbindung mit den Geodaten, die an vielen Drucksachen des lokalen RIS zu finden sind. Direkt vor Ort an einer Baustelle öffnen sie Beschlüsse, Pläne und Eingaben aus dem Planfeststellungsverfahren, die dieser Baustelle voran gegangen sind.

Zur Realisierung derartiger Szenarien können die Fähigkeiten von OParl-kompatiblen Servern mit den besonderen Eigenschaften der mobilen Endgeräte verknüpft werden.

Smartphones und Tablets verfügen beispielsweise, je nach Aufenthaltsort, über sehr unterschiedlich gute Internetanbindung. In einem Büro oder zuhause können Nutzer über ein WLAN Daten mit hoher Bandbreite austauschen, in Mobilfunknetzen vor allem außerhalb der Ballungsgebiete jedoch sinken die Bandbreiten deutlich. Einige Tablets werden sogar ohne Möglichkeit zur Mobilfunk-Datenübertragung genutzt. In solchen Fällen kann ein Cache auf dem Endgerät dazu dienen, Inhalte vorzuhalten, die dann auch bei langsamer oder fehlender Internetverbindung zur Verfügung stehen. Sobald dann wieder eine Verbindung mit hoher Bandbreite bereit steht, kann die App im Hintergrund, entweder über die Feeds der OParl API oder über den einzelnen Abruf von Objekten, die gecachten Inhalte aktualisieren.

Eine Stärke eines mobilen Clients ist auch die Möglichkeit der Personalisierung, also der Anpassung auf die Bedürfnisse und Interessen der Nutzerin oder des Nutzers. Es wäre beispielsweise denkbar, dass eine Nutzerin die parlamentarischen Informationssysteme, für die sie sich interessiert, dauerhaft in der App einrichtet und eine Favoritenliste der Gremien, die ihre bevorzugten Themengebiete behandeln, hinterlegt. Die App könnte aufgrund dieser Favoritenliste eigenständig über die API nach neuen Sitzungsterminen, Tagesordnungspunkten, Drucksachen und Dokumente suchen. Taucht dabei ein neues Objekt auf, wird die Nutzerin darüber benachrichtigt. Sie kann dann beispielsweise entscheiden, Dokumente direkt zu öffnen oder für den späteren Offline-Zugriff zu speichern.

Einem derartigen Szenario kommt das Graph-orientierte Datenmodell der OParl API entgegen. Ausgehend von einer Sitzung eines bestimmten Gremiums beispielsweise ist es damit einfach möglich, die in Verbindung stehenden Mitglieder des Gremiums, Teilnehmer der Sitzung, Tagesordnungspunkte der Sitzung oder Drucksachen zu den Tagesordnungspunkten und letztlich Dokumente zu Drucksachen und Sitzung abzurufen.

Für die Nutzer einer mobilen Client-Anwendung könnte es sich als besonders hilfreich erweisen, wenn Dokumente auf dem Server in verschiedenen Formaten zur Verfügung gestellt werden. Denn nicht jedes Endgerät mit kleinem Bildaschirm bietet eine nutzerfreundliche Möglichkeit, beispielsweise Dokumente im weit verbreiteten PDF-Format darzustellen. Hier könnte schon der Entwickler der mobilen App Mechanismen vorsehen, die, sofern vorhanden, besser geeignete Formate wie z.B. HTML abrufen.

Neben dem kleinen Display kann für einige mobile Endgeräte auch die im Vergleich zu einem zeitgemäßen Desktop-PC geringere CPU-Leistung eine Einschränkung darstellen. Solchen Geräten kommt es besonders entgegen, wenn der Server zu allen Dokumenten auch den reinen Textinhalt abrufbar macht, der dann beispielsweise für eine Volltextsuche auf dem Endgerät indexiert werden kann. So wiederum kann auf dem Client eine Suchfunktion realisiert werden, welche die OParl-API selbst nicht zur Verfügung stellt.

Eine solche Suchfunktion kann auch über die reine Volltxtsuche und über die Suche mittels Text- oder Spracheingabe hinaus gehen. Denn ein Client könnte von einem Server-System, das Drucksachen mit Geoinformationen anbietet, diese abrufen und räumlich indexieren. Anhand der Position des Geräts, die mittels GPS genau bestimmt werden kann, könnte so der lokale Cache nach Objekten in der Umgebung durchsucht werden. Das Ergebnis könnte auf einer Karte dargestellt oder in einer Ergebnisliste angezeigt werden, die nach Distanz zum Objekt sortiert werden könnte.

3.2 Szenario 2: Integration in Web-Portal

Portallösungen bieten den Betreibern die Möglichkeit, Inhalte auf einer einheitlichen Weboberfläche zu veröffentlichen, die aus verschiedensten Quellen und Plattformen bereitgestellt werden. Inhalte werden dabei häufig als sogenannte “Portlets” in Seiten integriert.

Ein Beispiel für die Realisierung eines solchen Integrations-Ansatzes wäre eine Kommune, die für ihre allgemeine Website eine Portallösung einsezt und hier auch Inhalte aus dem kommunalen Ratsinforamtionssystem einspeisen und darstellen möchte. Die Inhalte könnten

als Module mit anderen Inhalten, beispielsweise aus einem Web Content Management System (WCMS) gemeinsam auf einer Seite dargestellt werden.

Eine Seite über den Gemeinderat beispielsweise könnte durch ein Portlet ergänzt werden, in dem die nächsten Sitzungstermine des Gemeinderats aufgelistet werden. Eine Pressemeldung über ein bestimmtes Bauvorhaben, in dem ein Beschluss erwähnt wird, könnte direkt ein Portlet mit einer Detailsansicht der entsprechenden Drucksache einbinden.

Die Portlets, die von einem Portalserver zur Verfügung gestellt werden, stellen damit im Sinne der OParl-Architektur Clients dar. Je nach Performanz und Anforderungen im Einzelfall könnten diese Client mit eigenen Caches arbeiten oder aber direkt auf den jeweiligen OParl-Server zugreifen.

Vorteil einer solchen Einbindung, also der kontextbezogenen Darstellung von parlamentarischen Informationen im Gegensatz zu einem monolitischen parlamentarischen Informationssystem könnte sein, dass Nutzer in einer gewohnten und akzeptierten Oberfläche jeweils die relevanten Informationen erhalten, ohne sich an die ungewohnte Umgebung eines parlamentarischen Informationssystems gewöhnen zu müssen.

Die denkbaren Szenarien einer solchen Integration beschränken sich nicht auf anonyme Nutzer von öffentlichen Websites. In einem authentifizierten Umfeld wie beispielsweise einem kommunalen Intranet lassen sich weitere Arten von Portlets und damit Mehrwerte für die Nutzer realisieren. So könnte beispielswiese eine eingeloggte Nutzerin eine personalisierte Liste der Sitzungstermine, zu der sie eingeladen ist, angezeigt bekommen.

Die Standardisierung durch OParl sorgt im Rahmen der Portal-Szenarios dazu, dass Portlets, die für ein bestimmtes parlamentarisches Informationssystem entwickelt wurden, leichter auf andere Systeme - auch verschiedener Anbieter - ausgeweitet werden könne, sofern diese ebenfalls OParl-konform sind. Dies ermöglicht es beispielsweise verschiedenen Kommunen, ihre bei der Entwicklung von Portlets zusammen zu arbeiten und ihre Ergebnisse auszutauschen. Denkbar sind auch Portlet-Entwicklungen als Open-Source-Projekte.

3.3 Szenario 3: Meta-Suche

Die Ermöglichung einer nutzerfreundliche Suche, die damit verbundene Indexierung von verschiedensten Dokumenteninhalten und die Kategorisierung von Inhalten kann eine sowohl konzeptionell als auch technisch Anspruchsvolle Aufgabe sein. Auch im Hinblick auf die Server-Ressourcen sind damit nennenswerte Aufwände verbunden. Andererseits liegt auf der Hand, dass die effiziente Arbeit mit großen Informationsmengen nach intelligenten Möglichkeiten der Einschränkung von Informationsmengen auf das jeweils im Anwendungsfall relevante Treffer verlangt. Beispiel wäre ein Nutzer, der sich für alle Dokumente zum Thema Kreisverkehre interessiert. Die OParl-Spezifikation sieht keine Methoden vor, wie die Ausgabe des Servers schon bei der Anfrage von Dokumenten derart beschränkt werden können. Damit ist die Realisation von Such- und Filtermechanismen im OParl-Umfeld eine Aufgabe, die bis auf weiteres lediglich auf Seite der Clients angeboten werden kann.

Angelehnt an das seit den Anfängen des Web etablierte Modell der externen Web-Suchmaschine sind spezielle Suchmaschinen für OParl-konforme parlamentarische Informationssysteme denkbar. Diese können auch von dritten, beispielsweise zivilgesellschaftlichen Organisationen betrieben werden, die nicht Betreiber des Server-Systems sind. Solche Plattformen treten gegenüber dem OParl-Server als Client auf und rufen bestimmte oder sämtliche Informationen, die das System bereit hält, ab. Vorbild sind die Robots oder Spider von Web-Suchmaschinen. Die abgerufenen Informationen können dann indexiert und je nach Anforderungen für eine gezielte Suche weiterverarbeitet werden.

Dieses Modell ist grundsätzlich nicht auf einzelne OParl-Server oder einzelne Körperschaften beschränkt. Vielmehr könnte der Betreiber einer solchen Suchmaschine sich entschließen, die Informationen aus mehreren OParl-konformen Systemen zu indexieren. Nutzern könnte entweder angeboten werden, die Suche auf bestimmte Körperschaften, beispielsweise auf

eine bestimmte Kommune, zu beschränken, oder ohne Beschränkung über alle angebotenen Körperschaften zu suchen.[11]

Daraus ergeben sich vielfältige Anwendungsszenarien, die hier beispielhaft beschrieben werden:

- Eine Mitarbeiterin eines regionalen Zweckverbands hat die Aufgabe, Ratsvorgänge in den Mitgliedskommunen mit Relevanz für die Aufgaben des Verbandes im Blick zu behalten. Sie nutzt dafür ein regionales Internetportal, in dem die Inhalte der OParl-konformen parlamentarischen Informationssysteme der Mitgliedskommunen durchsuchbar sind. Um die Suche zu vereinfachen, hat sie einzelne Schlagwörter abonniert, zu denen sie automatisch über neue Vorgänge informiert wird.
- Ein Einwohner eines Ballungsraums will sich über aktuelle Vorgänge rund um seine Mietwohnung in Stadt A, sein Gartengrundstück in einer Kleingartenkolonie in der Nachbarstadt B und seinen Arbeitsplatz in Stadt C auf dem laufenden halten. Dazu abonniert er im regionalen Meta-Such-Portal parlamentarische Vorgänge mit räumlichem Bezug zu diesen drei Standorten und wird so automatisch über neue Aktivitäten informiert, die Relevanz für ihn haben könnten.
- Eine Landespolitikerin möchte einfacher über die politischen Aktivitäten ihrer Parteikollegen in den Rathäusern des Bundeslandes informiert werden. Dazu nutzt sie ein Internetportal, in dem die Informationen aus den parlamentarischen Informationssystemen mit OParl-Schnittstelle im Land zusammengeführt werden. Dort hat sie sich Abonnements zu einzelnen Lokalpolitikern eingerichtet und wird automatisch über ihre Teilnahme an Gremiensitzungen und die Themen dieser Sitzungen informiert.

3.4 Szenario 4: Forschungsprojekt Themen- und Sprachanalyse

In einem Forschungsprojekt sollen Pro- und Contra-Argumentationen bei Ratsdiskussionen zum Ausbau von Stromtrassen identifiziert werden. Über die Analyse in mehreren Gebietskörperschaften sollen die gefundenen Argumentationen zu wiederkehrenden Mustern verdichtet und festgestellt werden, wie diese Muster regional abweichen.

Dazu nutzen die Mitarbeitenden des Forschungsprojektes die OParl-Schnittstellen der parlamentarischen Informationssysteme aller Kommunen entlang der geplanten überregionalen Trassen. Über diese einheitlichen Schnittstellen können sie insbesondere die relevanten Wortprotokolle abrufen und zum Beispiel in einem Werkzeug zur qualitativen Datenanalyse lokal verarbeiten. Im Ergebnis ließe sich auch erkennen, wie ähnlich oder wie unterschiedlich die Argumente in rhetorischer Hinsicht vorgetragen werden.

4 Prinzipien und Funktionen des Vokabulars und der API

TODO

(In diesem Kapitel werden die Zugriffsmethoden der OParl-konformen Schnittstelle beschrieben. Hierzu gehören alle chapter-Dateien, deren Nummerierung mit der Ziffer 6 beginnnt.)

Stichpunkte:

- Optional gzip Encoding und andere Kodierungen, wenn Client und Server dies unterstützen
- Authentifizierung wird nicht benötigt.

[11] Daher der Begriff Meta-Suche

4.1 Designprinzipien

4.1.1 Aufbauen auf gängiger Praxis

Grundlage für die Erarbeitung der OParl-Spezifikation in der vorliegenden Version ist eine Analyse von aktuell (2012 bis 2014) in Deutschland etablierten parlamentarischen Informationssystemen und ihrer Nutzung. Erklärtes Ziel für diese erste Version ist es, mit möglichst geringem Entwicklungsaufwand auf Seite der Softwareanbieter und Migrationsaufwand auf Seite der Betreiber zu einer Bereitstellung von parlamentarischen Informationen über eine OParl API zu gelangen. Hierbei war es von entscheidender Bedeutung, dass sich die Informationsmodelle der einschlägigen Softwareprodukte stark ähneln. Für die OParl-Spezifikation wurde sozusagen ein Datenmodell als "gemeinsamer Nenner" auf Basis der gängigen Praxis beschrieben.

4.1.2 Verbesserung gegenüber dem Status Quo wo möglich

Dort, wo es dem Ziel der einfachen Implementierbarkeit und der einfachen Migration nicht im Weg steht, erlauben sich die Autoren dieser Spezifikation, auch Funktionen aufzunehmen, die noch nicht als gängige Praxis im Bereich der Ratsinformationssysteme bezeichnet werden können oder welche nur von einzelnen Systemen unterstützt werden. Solche Funktionen sind dann so integriert, dass sie nicht als zwingende Anforderung gelten.

Ein Beispiel für eine derartige Funktion ist die Abbildung von Geodaten im Kontext von Drucksachen (`oparl:Paper`), um beispielsweise die Lage eines Bauvorhabens, das in einer Beschlussvorlage behandelt wird, zu beschreiben. Zwar ist den Autoren nur ein einziges parlamentarisches Informationssystem[12] in Deutschland bekannt, das Geoinformationen - und zwar in Form von Punktdaten, also einer Kombination aus Längen- und Breitengradangaben - mit Dokumenten verknüpft. Der Vorteil dieser Funktion ist jedoch anhand zahlreicher Anwendungsszenarien belegbar. Somit ist der vorliegenden OParl-Spezifikation die Möglichkeit beschrieben, Geodaten-Objekte einzubetten.

Die Angabe eines einzelnen Punktes ist dabei nur ein einfacher Sonderfall. Die Spezifikation erlaubt auch die Kodierung von mehreren Objekten, die Punkte, Linien oder Polygone repräsentieren können. Vgl. dazu `oparl:Location`.

Auch die Ausgabe einer Nur-Text-Version im Kontext des Dokuments (`oparl:Document`), das den barrierefreien Zugriff auf Inhalte oder Indexierung für Volltextsuchfunktionen deutlich vereinfacht, ist eine Möglichkeit, die in der gängigen Praxis noch nicht zu finden ist. Ebenso die Möglichkeit, Beziehungen zwischen einzelnen Dokumenten herzustellen, um so von einem Dokument zu anderen Dokumenten mit identischem Inhalt, aber in anderen technischen Formaten zu verweisen, etwa von einer ODT-Datei zu einer PDF-Version.

4.1.3 RESTful

Die Bezeichnung "REST" (für "Representational State Transfer") wurde im Jahr 2000 von Roy Fielding eingeführt[13]. Die Definition von Fielding reicht sehr weit und berührt viele Details. In der Praxis wird der Begriff häufig genutzt, um eine Schnittstelle zu beschreiben,

- die auf WWW-Technologie aufbaut, insbesondere dem HTTP-Protokoll
- die darauf beruht, dass mittels URL einzelne Ressourcen oder Zustände vom Client abgerufen werden können.
- die zustandslos ist. Das bedeutet, die Anfrage eines Clients an den Server enthält alle Informationen, die notwendig sind, um die Anfrage zu verarbeiten. Auf dem Server wird kein Speicher zur Verfügung gestellt, um beispielsweise den Zustand einer Session zu speichern.

[12] Das Ratsinformationssystem BoRis, eine Eigenentwicklung der Stadt Bonn http://www2.bonn.de/bo_ris/ris_sql/agm_index.asp

[13] Fielding, Roy: Architectural Styles and the Design of Network-based Software Architectures, http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Diese Prinzipien macht sich auch OParl zunutze. Damit gilt prinzipiell, dass eine OParl-konforme Server-Schnittstelle auch als "RESTful" gelten darf.

4.1.4 Selbstbeschreibungsfähigkeit

Ausgaben des Servers sollten so beschaffen sein, dass sie für menschliche NutzerInnen weitgehend selbsterklärend sein können. Dies betrifft besonders die Benennung von Objekten und Objekteigenschaften.

Um den Kreis der Entwicklerinnen und Entwickler, die mit einer OParl-API arbeiten können, nicht unnötig einzuschränken, wird hierbei grundsätzlich auf englischsprachige Begrifflichkeiten gesetzt.

4.1.5 Erweiterbarkeit

Implementierer sollen in der Lage sein, über eine OParl-konforme Schnittstelle auch solche Informationen auszugeben, die nicht im Rahmen des OParl-Schemas abgebildet werden können. Dies bedeutet zum einen, dass ein System Objekttypen unterstützen und ausliefern darf, die nicht (oder noch nicht) im OParl Schema beschrieben sind. Das bedeutet auch, dass Objekttypen so um eigene Eigenschaften erweitert werden können, die nicht im OParl Schema beschrieben sind.

Ein weiterer Aspekt betrifft die Abwärtskompatiblität, also die Kompatibilität von OParl-Clients mit zukünftigen Schnittstellen. So können beispielsweise zukünftige Erweiterungen des OParl Schemas, etwa um neue Objekttypen, genau so durchgeführt werden wie die Erweiterungen um herstellerspezifische Objekttypen. Ein Client muss diese Anteile nicht auswerten, sofern sie nicht für die Aufgabe des Clients relevant sind.

Diese angestrebte Erweiterbarkeit wird durch weitgehend durch das JSON-LD-Format gewährleistet. Es erlaubt die Verflechtung von Objekttypen-Definitionen aus verschiedenen Schemata.

4.1.6 Browseability/Verlinkung

Klassische Webservice-Schnittstellen erfordern von den Entwicklern vollständige Kenntnis der angebotenen Einstiegspunkte und Zugriffsmethoden, gepaart mit sämtlichen unterstützten URL-Parametern, um den vollen Funktionsumfang der Schnittstelle ausschöpfen zu können.

Parlamentarische Informationen sind weitgehend graphartig aufgebaut. Das bedeutet, dass Objekte häufig mit einer Vielzahl anderer Objekte verknüpft sind. So ist eine Person beispielsweise Mitglied in mehreren Gremien, das Gremium hat mehrere Sitzungen abgehalten und zu diesen Sitzungen gibt es jeweils zahlreiche Drucksachen, die ihrerseits wieder zahlreiche Dokumente enthalten.

Eine OParl-Schnittstelle gibt jedem einzelnen Objekt eine eindeutige Adresse, eine URL. Somit kann die Schnittstelle den Verweis von einem Objekt, beispielsweise einem Gremium, auf ein anderes Objekt, etwa ein Mitglied des Gremiums, dadurch ausgeben, dass im Kontext des Gremiums die URL des Mitglieds ausgeben wird. Der Client kann somit ausgehend von einem bestimmten Objekt die anderen Objekte im System finden, indem er einfach den angebotenen URLs folgt. Dieses Prinzip wird auch "Follow Your Nose" genannt[14].

4.1.7 Linked Data

Der Begriff "Linked Data" steht für die Beschreibung von Daten in einer Form, die diese über ihren ursprünglichen Kontext hinaus verständlich macht.[15]

[14] http://patterns.dataincubator.org/book/follow-your-nose.html

[15] vgl. Bundesministerium des Innern (Herausg.): Open Government Data Deutschland, Seite 433f., 2012 http://www.bmi.bund.de/SharedDocs/Downloads/DE/Themen/OED_Verwaltung/ModerneVerwaltung/opengovernment.pdf

Kern von Linked Data ist die Möglichkeit, alle Bestandteile von Daten in Form von Tripeln zu beschreiben, das sind dreiteilige Informationseinheiten aus einem Subjekt, einem Prädikat und einem Objekt. Alle drei Bestandteile können in Form global eindeutiger "Uniform Resource Identifier" (URI) abgebildet werden.

Nach dem Linked-Data-Prinzip könnte beispielsweise der Vorname einer Person mit dem folgenden Tripel beschrieben werden:

```
Subjekt: http://dbpedia.org/resource/John_Doe_(musician)
Prädikat: http://xmlns.com/foaf/0.1/givenName
Objekt: http://dbpedia.org/resource/John_(given_name)
```

Hierbei macht man von der Tatsache Gebrauch, dass das Subjekt, also die Person, um die es geht, bereits mittels ihrer URI eindeutig identifiziert werden kann und dass bestenfalls unter dieser URI weitere Informationen zu der Person abrufbar sind.[16] Auch für das Prädikat "Person hat den Vornamen" liegt bereits eine Beschreibung in einem gebräuchlichen Vokabular vor, auf das hier verwiesen werden kann. Und schließlich kann sogar der eigentliche Vorname in Form einer URI abgebildet werden, nämlich als Verweis auf eine umfangreiche Beschreibung dieses Namens.

Das **Ziel** von OParl ist es, mit der vorliegenden Version 1.0 der Spezifikation, die Nutzung solcher allgemeingültigen Vokabulare für die Veröffentlichung von parlamentarischen Informationen zu begünstigen und die automatisierte Verarbeitung und Verknüpfung von Informationen, auch über die Grenzen verschiedener Informationssysteme hinweg, zu erleichtern.

Beispiele, wo dies sinnvoll ist, sind in der Praxis leicht zu finden. So finden sich beispielsweise in vielen lokalen Parlamenten immer wieder Fraktionen der selben Parteien, beispielsweise CDU und SPD. Mittels Linked Data wäre es möglich, jede dieser Fraktionen mit einer externen URL zu verknüpfen[17] und somit erkennbar zu machen, zu welcher Partei diese Fraktion gehört. Ebenso finden sich viele inhaltliche Ähnlichkeiten bei Gremien wie zum Beispiel Ausschüssen (z.B. Hauptausschuss, Verkehrsausschuss etc.) oder bei Arten von Drucksachen (z.B. Anträge, Anfragen, Mitteilungen, Beschlussvorlagen).

OParl lässt in Version 1.0 der Spezifikation noch viele Aufgaben, die die Vereinheitlichung dieses Vokabulars betreffen, offen. Jedoch wird durch die Verwendung von JSON-LD als Serialisierungsformat der Grundstein für eine Vereinheitlichung im Sinne von Linked Data gelegt.

4.2 Zukunftssicherheit

Wie unter Designprinzipien beschrieben, ist diese erste Version der OParl-Spezifikation bereits im Wesentlichen von den Zielen der einfachen Implementierbarkeit und Migration geleitet.

Der Aufwand, den die Betreiber von parlamentarischen Informationssystemen bei der Bereitstellung von OParl-konformen Schnittstellen betreiben, soll auch bei der zukünftigen Weiterentwicklung dieser Spezifikation berücksichtigt werden. Ebenso soll den Entwicklern von Client-Software zukünftig entgegen kommen, dass ihre bestehenden Clients auch mit Servern kommunizieren können, die eine neuere Version der OParl-Spezifikation unterstützen. Dieser Wunsch ist bereits im Designprinzip Erweiterbarkeit ausformuliert.

Mit anderen Worten: die Autoren der OParl-Spezifikation beabsichtigen größtmögliche Zukunftssicherheit und zukünftige Abwärtskompatibilität. Dieses Ziel wird in Zukunft natürlich abgewägt werden müssen mit dem Wunsch, sich an Veränderungen und neue Erkenntnisse anzupassen. Eine Garantie für Zukunftssicherheit kann insofern niemand aussprechen.

[16] Ein Aufruf der URL `http://dbpedia.org/resource/John_Doe_(musician)` im herkömmlichen Web-Browser führt zu einer Weiterleitung auf die URL `http://dbpedia.org/page/John_Doe_(musician)`. Siehe dazu auch der Abschnitt Content Negotiation

[17] beispielsweise `http://dbpedia.org/resource/Christian_Democratic_Union_(Germany)` und `http://dbpedia.org/resource/Social_Democratic_Party_of_Germany`

Ein fiktives Szenario soll verdeutlichen, dass es zweckmäßig ist, schon beim Betrieb eines OParl 1.0 Servers die zukünftige Entwicklung im Blick zu haben:

- Die Kommune *Beispielstadt* betreibt ihren OParl-1.0-Server unter der URL `https://oparl.bstadt.de/1.0/`.
- Verschiedene Clients, die für OParl Version 1.0 entwickelt wurden, kommen bei Nutzerinnen und Nutzern, die sich für den Stadtrat in Beispielstadt intressieren, zum Einsatz. Jeder Client-Nutzer hat dazu lediglich die URL `https://oparl.bstadt.de/1.0/` des OParl-Servers in der Client-Konfiguration hinterlegt.
- Die OParl-Spezifikation wird aktualisiert, es erscheint Version 1.1. Das Schema enthält Erweiterungen gegenüber Version 1.0, jedes gültige Objekt aus Version 1.0 behält jedoch auch weiterhin seine Gültigkeit. Und Objekte, die nach Version 1.1 gültig sind, sind auch für Clients gültig, die für Version 1.0 entwickelt wurden.
- Die Firma, die den OParl-Server von Beispielstadt entwickelt hat, liefert ein Update.
- Der OParl-Server von Beispielstadt ist nun über eine neue URL `https://oparl.bstadt.de/1.1/` zu erreichen. Alle Anfragen an `https://oparl.bstadt.de/1.0/...` werden auf die entsprechende URL unter `https://oparl.bstadt.de/1.1/` mit HTTP-Redirects und Status-Code 301 weiter geleitet.
- Die Nutzer der Clients, die mit dem OParl-Server von Beispielstadt arbeiten, können weiter arbeiten wie bisher. Sie erhalten vom Client höchstens einmalig eine Information, dass sich die Server-URL geändert hat.
- Einzelne Client-NutzerInnen werden von den Anbietern ihrer Clients darauf aufmerksam gemacht, dass eine neue Version ihres Produkts für eine neue OParl-Version zur Verfügung steht. Mit dieser Version könnten die Nutzer in den Genuss der Vorteile von OParl Version 1.1 kommen.
- Nach einiger Zeit erscheint eine neue Version 2.0 der OParl Spezifikation. Hier haben sich größere Änderungen ergeben. Das Schema ist nicht kompatibel mit dem von Version 1.0 und 1.1. Clients, die für eine Version 1.* entwickelt wurden, werden nicht sinnvoll mit einem Server der Version 2 kommunizieren können.
- Der Server-Entwickler bietet das entsprechende Produkt zu OParl Version 2 an, Beispielstadt entschließt sich zum Einsatz der neuen Version. Da das Server-Produkt gleichzeitig OParl 1.* und OParl 2.0 bedienen kann, kann Beispielstadt gleichzeitig einen Endpunkt für 1.1 und einen für 2.0 betreiben. Die URL des neuen Endpunkts lautet `https://oparl.bstadt.de/2.0/`.

Das Szenario verdeutlicht, wie insbesondere zwei Aspekte für eine möglichst sanfte Migration zwischen den OParl-Versionen sorgen können:

1. Dedizierte API-Endpunkt-URLs für jede OParl-Version
2. HTTP-Weiterleitungen auf die neue URL, sofern diese kompatibel mit der alten ist, erspart den Parallelbetrieb von zwei ähnlichen Endpunkten und kommuniziert den Clients automatisch den Endpunkt der neuen Version

Zu der Art, wie die OParl-Version sich auf die Endpunkt-URL auswirkt, will diese Spezifikation keine Vorgaben machen. Die Pfad-Elemente im obigen Szenario sind Vorschläge, aber in keiner Weise bindend.

Die praktische Umsetzung von HTTP-Weiterleitungen ist besonders dann trivial, wenn die restlichen URL-Bestandteile identisch bleiben. In diesem Fall können Server mit einer einfachen Regel von jeglicher vorherigen auf jegliche neue URL weiter leiten.

4.3 HTTP und HTTPS

OParl-Server und -Client kommunizieren miteinander über das HTTP-Protokoll.

Hierbei SOLL eine verschlüsselte Variante des Protokolls, auch HTTPS genannt, zum Einsatz kommen, alternativ kann jedoch auch unverschlüsseltes HTTP verwendet werden. Welche Verschlüsselungstechnologie im Fall von HTTPS gewählt wird, obliegt dem Betreiber bzw. Server-Implementierer.

Die Wahl des unverschlüsselten oder verschlüsselten HTTP-Zugriffs hat Auswirkung auf die im System verwendeten URLs. Wie im Kapitel URLs beschrieben, verfolgt diese Spezifikation die Festlegung auf genau eine "kanonische" URL je Ressource (URL-Kanonisierung).

Bei unverschlüsseltem Zugriff wird allen URLs, die auf das betreffende System zeigen, das Schema "http://" voran gestellt, beim verschlüsselten Zugriff stattdessen "https://".

Es ist daher ZWINGEND, dass der Server-Betreiber sich zur URL-Kanonisierung für nur eine von beiden Varianten entscheidet. Beantwortet das System regulär Anfragen über HTTPS mit der Auslieferung von Objekten etc., dann MUSS das System bei Anfragen an die entsprechenden URLs ohne "https://" Schema mit einer Weiterleitung antworten (HTTP Status-Code 301).

Gleiches gilt umgekehrt: beantwortet das System regulär Anfragen über unverschlüsseltes HTTP, dann MÜSSEN Anfragen auf die entsprechenden URLs mit "https://"-Schema mit einer HTTP-Weiterleitung (HTTP Status-Code 301) beantwortet werden. TODO: Geht das, wenn HTTPS nicht unterstützt wird?

4.4 URLs, IRIs und URIs

Den URLs (für "Uniform Resource Locators") kommt bei einer OParl-konformen API eine besondere Bedeutung zu und es werden eine Reihe von Anforderungen an die Verarbeitung von URLs gestellt.

Im Rahmen dieses Dokuments wird aus Gründen der Verständlichkeit generell der allgemein gebräuchlichere Begriff "URL" werwendet, auch wenn damit tatsächlich die internationalisierte Variante nach RFC 3987[18], die korrekterweise IRI bzw. "Internationalized Resource Identifier" genannt werden müsste, gemeint ist. In andern Kontexten wiederum wird von URIs bzw. "Uniform Resource Identifier" gesprochen. Das vorliegende Dokumente fasst alle drei Konzepte mit dem Begriff "URL" zusammen und ignoriert damit die Unterschiede der einzelnen Begriffe, da diese im Rahmen dieser Spezifikation ohne bedeutung sind.

Die grundsätzliche Funktionsweise von URLs ist in RFC 3986 beschrieben[19].

Der Aufbau einer beispielhaften URL mit den Bezeichnungen, wie sie in diesem Dokument Verwendung finden:

```
https://refserv.oparl.org/foo/bar/?skip=234
\___/   \_____________/\_______/ \______/
  |         |                |        |
Schema     Host             Pfad  Query-String
```

4.4.1 URL-Kanonisierung

Absicht ist, dass jedes benannte Objekt[20], das ein Server über eine OParl-API anbietet, über genau eine URL identifizierbar und abrufbar ist. Diese Vereinheitlichung der URL nennen wir Kanonisierung.

Die Kanonisierung ist entscheidend, um erkennen zu können, ob zwei URLs das selbe Objekt repräsentieren. Sind zwei URLs identisch, sollen Clients daraus ableiten können, dass diese

[18] RFC 3987: http://tools.ietf.org/html/rfc3987
[19] RFC 3986: http://tools.ietf.org/html/rfc3986
[20] vgl. Benannte und anonyme Objekte

das selbe Objekt repräsentieren. Sind zwei URLs unterschiedlich, soll im Umkehrschluss die Annahme gelten, dass sie zwei verschiedene Objekte repräsentieren.

Der OParl-konforme Server MUSS für jedes benannte Objekt eine kanonische URL bestimmen können.

Die URL-Kanonisierung betrifft sämtliche Bestandteile der URL. Entsprechend beginnt diese schon beim **Schema** und bei der Entscheidung durch den Betreiber, ob eine OParl-API regulär über HTTP oder über HTTPS erreichbar sein soll (vgl. HTTP und HTTPS).

Der **Host**-Teil der URL wird ebenfalls durch die Konfiguration des Betreibers festgelegt. Obwohl technisch auch die Verwendung einer IP-Adresse (z.B. "123.123.123.123") möglich wäre, SOLL der Betreiber einen mit Bedacht gewählten Host-Namen einsetzen. Die Vorteile dieser Lösung gegenüber der Verwendung einer IP-Adresse sind vielfältig:

- NutzerInnen können Host-Namen lesen und interpretieren
- In Kombination mit der richtigen Domain (oder Subdomain) kann der Hostname kommunizieren, wer der Betreiber ist.
- Host-Namen können zwischen verschiedenen technischen Systemen (bzw. von IP-Adresse zu IP-Adresse) migriert werden, was hilft, die Langlebigkeit der URLs zu gewährleisten

Eine URL wie

```
http://oparl.stadtrat.stadt-koeln.de/
```

kommuniziert beispielsweise direkt die Zugehörigkeit zur Stadt Köln als Betreiber des Systems. Die Bezeichnung "stadtrat" in der Subdomain zeigt den Zweck des Systems allgemein verständlich an. Der Host-Name "oparl.stadtrat.stadt-koeln.de" deutet an, dass diese URL zu einer OParl-Schnittstelle zu diesem System gehört.

Um die Kanonisierung zu gewährleisten, sind vom Betreiber alle notwendigen Faktoren auszuschließen, die dazu führen können, dass eine Ressource neben der kanonischen URL noch über andere URLs abrufbar ist. Diese Faktoren könnten sein:

- Der selbe Server antwortet nicht nur über den kanonischen Host-Namen, sondern auch noch über andere Host-Namen. Das könnte zum Beispiel der Fall sein, wenn der Host-Name als CNAME für einen anderen Namen konfiguriert wurde oder wenn ein DNS A-Record für die IP-Adresse des Servers existiert.
- Der Server ist neben dem Host-Namen auch über die IP-Adresse erreichbar.
- Zusätzliche Domains, die einen A-Record auf den selben Server besitzen

Zu der kanonischen Beispiel-URL https://oparl.stadtrat.stadt-koeln.de/ wären eine Reihe von nicht-kanonischen URL-Varianten denkbar, die technischen auf den selben Server führen könnten:

- https://83.123.89.102/
- https://oparl.stadtrat.stadtkoeln.de/
- https://risserv.stadt-koeln.de/

Falls es aus technischen Gründen nicht möglich ist, den Zugang auf das OParl-System über nicht-kanonische URLs zu unterbinden, SOLL eine entsprechende HTTP-Anfrage mit einer Weiterleitung auf die entsprechende kanonische URL beantwortet werden. Dabei ist der HTTP-Status-Code 301 zu verwenden.

Server-Implementierern wird empfohlen, hierfür den Host-Header der HTTP-Anfrage auszuwerten und mit der konfigurierten Einstellung für den kanonischen Hostnamen des Systems abzugleichen.

Beim **Pfad**-Bestandteil der URL MÜSSEN Server-Implementierer darüber hinaus beachten, dass nur jeweils eine Schreibweise als die kanonische Schreibweise gelten kann. Dazu gehört auch die Groß- und Kleinschreibung, die Anzahl von Schrägstrichen als Pfad-Trennzeichen, die Anzahl von führenden Nullen vor numerischen URL-Bestandteilen und vieles mehr.

Die Kanonisierung umfasst auch den **Query-String**-Bestandteil der URL. Wie auch beim Pfad, gilt hier, dass für jeden Parameter und jeden Wert im Query-String nur eine kanonische Schreibweise gelten MUSS.

Darüber hinaus SOLL der Server-Implementierer darauf achten, bei Verwendung von Query-String-Parametern diese in URLs immer nach dem selben Prinzip zu sortieren. Ein Beispiel: die beiden URLs

```
https://oparl.beispielris.de/members?body=1&committee=2
https://oparl.beispielris.de/members?committee=2&body=1
```

unterscheiden sich lediglich in der Reihenfolge der Query-String-Parameter. Da sie jedoch nicht identisch sind, müssen Clients annehmen, dass beide URLs verschiedene Objekte repräsentieren. In der Konsequenz kann es zu vermeidbarer Ressourcennutzung sowohl auf Client- als auch auf Serverseite kommen.

4.4.2 Langlebigkeit

Weiterhin ist es Absicht, dass URLs von Objekten langlebig sind, so dass sie, wenn sie einmal verbreitet wurden, langfristig zur Abfrage des dazugehörigen Objekts verwendet werden können.

Um dies zu gewährleisten, wird den **Betreibern** empfohlen, die Wahl der Domain, eventuell der Subdomain und letztlich des Host-Namens sorgfältig auf seine längerfristige Verwendbarkeit abzuwägen.

Server-Implementierer SOLLEN darüber hinaus dafür sorgen, dass der Pfad-Bestandteil der URLs die Langlebigkeit der URLs unterstützt. Es gelten die folgenden Empfehlungen, die jedoch keinen Anspruch auf Vollständigkeit erheben:

- **Veränderliche Objekt-Eigenschaften nicht als URL-Bestandteil nutzen.** In URLs sollten nur Eigenschaften des Objekts aufgenommen werden, die keinen Veränderungen unterliegen. Ändert sich beispielsweise die Kennung einer Drucksache im Verlauf ihrer Existenz, dann scheidet sie für die Bildung der URL aus.
- **Technische Eigenschaften der Implementierung verbergen.** Ist ein OParl-Server beispielsweise in PHP implementiert, sollte dies nicht dazu führen, dass im Pfad ein Bestandteil wie “oparl.php/” erscheint. Erfahrungsgemäß überdauern solche URLs nur kurz.

Weitere Empfehlungen für langlebige URLs liefern Tim Berners-Lee[21] sowie die Europäische Kommission[22].

4.5 Serialisierung mittels JSON-LD und JSONP

Eine OParl-konforme API gibt Objekte in Form von JSON aus. Die Objekte werden dabei entsprechend der JSON-LD Spezifikation um Kontexte erweitert, welche die Selbstbschreibungsfähigkeit der ausgegebenen Daten verbessert. Auf Anforderung des Clients wird darüber hinaus JSONP unterstützt.

[21] Berners-Lee, Tim: Cool URIs don’t change. http://www.w3.org/Provider/Style/URI.html

[22] Study on persistent URIs, with identification of best practices and recommendations on the topic for the MSs and the EC. (PDF) https://joinup.ec.europa.eu/sites/default/files/D7.1.3%20-%20Study%20on%20persistent%20URIs.pdf

In jedem Fall MUSS ein Server die Anfrage eines Clients unter Verwendung des HTTP `Content-type`-Headers `application/ld+json` beantworten. Die Spezifikation von JSON-LD liefert dazu genauere Informationen[23], auch zu dem optionalen Parameter `profile` für die explizite Anforderung von JSON-LD in einer dieser drei Unterformen: kompakt, expandiert oder flach.

Wenn der Server auch Anfragen nach `application/json` akzeptiert, dann SOLL er expandierte JSON-LD Dokumente liefern (also solche ohne `@context`).

TODO: MUSS der Server solche Legacy-Anfragen akzeptieren? Warum?

4.5.1 JSON

Die Abkürzung JSON steht für "JavaScript Object Notation". Das JSON-Format ist in RFC4627[24] beschrieben. Nachfolgend werden nur die wichtigsten Definitionen übernommen, um eine Terminologie zur weiteren Verwendung in diesem Dokument zu etablieren.

Das JSON-Format unterstützt die Ausgabe von vier verschiedenen primitiven Datentypen:

- *Zeichenkette* (Unicode)
- *Zahl* (sowohl Ganzzahlen als auch Fließkommazahlen)
- *Wahrheitswert* (`true` oder `false`)
- *Null*

Darüber hinaus werden zwei komplexe Datentypen unterstützt:

- *Objekt*: Eine Sammlung von Schlüssel-Wert-Paaren ohne Reihenfolge, wobei der Schlüssel eine Zeichenkette sein muss und der Wert ein beliebiger Datentyp sein kann.
- *Array*: Eine geordnete Liste mit beliebigen Datentypen.

Beispiel eines Objekts in JSON-Notation:

```
{
    "zeichenkette": "Das ist eine Zeichenkette",
    "zahl": 1.23456789,
    "wahrheitswert": true,
    "null": null,
    "objekt": {
        "foo": "bar"
    },
    "array": ["foo", "bar"]
}
```

4.5.2 JSON-LD

Das Kürzel LD im Namen "JSON-LD" steht für "Linked Data"[25]. Entsprechend erweitert die JSON-LD-Spezifikation[26] das JSON-Format um die Möglichkeit,

- Objekte mit anderen Objekten zu verknüpfen,
- Objekte und Eigenschaften bestimmten Typen zuzuordnen und damit
- Auskunft über die semantische Bedeutung von Objekten und Eigenschaften zu geben.

[23] JSON-LD 1.0: IANA COnsiderations: http://www.w3.org/TR/json-ld/#iana-considerations
[24] RFC4627: https://tools.ietf.org/html/rfc4627
[25] siehe dazu Linked Data
[26] http://www.w3.org/TR/json-ld/

Ein Beispiel aus der JSON-LD-Spezifikation illustriert, wie JSON-LD ein Objekt um zusätzliche semantische Informationen erweitert. Als Ausgangspunkt dient eine Personenbeschreibung in gewöhnlichem JSON:

```
{
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/",
  "image": "http://manu.sporny.org/images/manu.png"
}
```

Als menschlicher Betrachter kann man leicht erkennen, dass die Eigenschaft `name` den Namen der Person enthält, dass `homepage` die Website der Person sein könnte und dass `image` die URL einer Bilddatei der Person sein könnte. Ein automatisierter Client jedoch, dem die Objekteigenschaften nicht bekannt sind, kann die Bedeutung dieser Eigenschaften nicht entschlüsseln.

Entsprechend der JSON-LD-Spezifikation kann diese Erläuterung über die `@context`-Eigenschaft direkt im selben Objekt, sozusagen als Unterobjekt, mitgeliefert werden:

```
{
  "@context":
  {
    "name": "http://xmlns.com/foaf/0.1/name",
    "image": {
      "@id": "http://xmlns.com/foaf/0.1/img",
      "@type": "@id"
    },
    "homepage": {
      "@id": "http://xmlns.com/foaf/0.1/homepage",
      "@type": "@id"
    }
  },
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/",
  "image": "http://manu.sporny.org/images/manu.png"
}
```

Hier sind die Eigenschaften wie `image` einer URL wie http://schema.org/image zugewiesen. Ein Client, der diese URL kennt, kann daraus folgern, dass über die Objekteigenschaft `image` immer die URL eines Bildes zu finden ist. Das Schlüssel-Wert-Paar

```
"@type": "@id"
```

sagt darüber hinaus aus, dass der Wert dieser Eigenschaft die URL eines anderen Objekts ist[27]. Mittels `@type`-Deklaration könnte aber auch beispielsweise eine Eigenschaft, die im JSON-Sinn eine Zeichenkette ist, als Datum deklariert werden.

Am obigen Beispiel fällt auf, dass der `@context`-Teil des Objekts schon mehr Daten umfasst, als die eigentlichen Objekteigenschaften. Sinnvollerweise kann jedoch der gesamte Inhalt des `@context`-Teils in eine externe Ressource ausgelagert werden. Das folgende Beispiel verdeutlicht dies:

```
{
  "@context": "http://json-ld.org/contexts/person.jsonld",
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/",
  "image": "http://manu.sporny.org/images/manu.png"
}
```

[27] URLs heißen in der JSON-LD-Spezifikation "IRI" (für "Internationalized Resource Identifier"), wir verwenden hier jedoch weiterhin die Bezeichnung "URL".

Die `@context`-Eigenschaft hat nun als Wert eine URL. Die URL (hier: http://json-ld.org/contexts/person.jsonld) gibt wiederum in JSON kodiert die Beschreibung aller möglichen Attribute des Objekts aus. Die Kontext-Beschreibung des JSON-LD-Objekts wurde somit in eine externe Ressource ausgelagert. Clients SOLLEN davon ausgehen, dass sich diese externen Kontextbeschreibungen nur selten ändern. Somit genügt es, bei Abruf vieler gleichartiger JSON-LD-Objekte vom Server die Kontext-Ressource nur einmal zu laden.

Im Sinne der JSON-LD-Spezifikation sind Objekte mit eingebettetem und externem Kontext identisch. Den Implementierern eines OParl-konformen Servers wird EMPFOHLEN, grundsätzlich die Kontextinformation mittels externer Ressourcen zu übermitteln. Die OParl Autoren werden hierzu die zu dieser Spezifikation passenden Ressourcen auf oparl.org für jegliche Verwendung zur Verfügung stellen (mehr dazu im Anhang). Sollten Server-Implementierer zusätzliche Objekttypen benötigen, die nicht von dieser Spezifikation abgedeckt sind, SOLL entsprechend zusätzlich auf eigene Kontextressourcen unter geeigneten URLs verwiesen werden. Hierbei können herstellereigene und OParl-spezifische URls gemischt werden, wie in einem Beispiel weiter unten verfeutlicht wird.

JSON-LD ermöglicht es auch, für ein Objekt einen **Objekttyp** zu kommunizieren. So könnte passend zu unserem Beispiel ausgedrückt werden, um welche Art von Objekt es sich bei den vorliegenden Daten handelt. Dazu wird die `@type`-Eigenschaft verwendet, deren Wert eine URL ist:

```
{
  "@context": "http://json-ld.org/contexts/person.jsonld",
  "@type": "http://schema.org/Person",
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/",
  "image": "http://manu.sporny.org/images/manu.png"
}
```

Objekte können mehreren Typen zugeordnet sein und damit die Eigenschafen mehrerer Objekttypen nutzen. Im Fall von OParl kann diese Möglichkeit genutzt werden, um über die API Eigenschaften auszugeben, die nicht Teil des OParl-Schemas sind.

```
{
  "@context": {
    "oparl": "http://oparl.org/schema/1.0/",
    "vendor": "http://www.vendor.de/oparl/schema/"
  },
  "@type": ["oparl:Paper", "vendor:Drucksache"],
  "title": "Beschlussvorlage zum Haushalt",
  "created": "2013-05-29T14:17:39+02:00",
  "aktenzeichen": "ABC123"
}
```

Das Beispiel oben zeigt ein Objekt, das über die `@context`-Eigenschaft zwei verschiedene URLs als sogenannte Vokabulare referenziert. Das eine Vokabular wird durch das Namensraum-Präfix `oparl` repräsentiert, das zweite (herstellereigene) durch das Namensraum-Präfix `vendor`.

Durch das Schlüsselwort `@type` wird nun dem Objekt ein oder mehrere Objekttypen zugewiesen. Dabei werden die zuvor beschriebenen Namensraum-Präfixe genutzt. Ein JSON-LD-Client verarbeitet Namensraum-Präfixe und Typenbezeichnung so, dass diese letztlich für jeden Objekttypen eine eindeutige URL ergeben.

- Aus `oparl:Paper` wird `http://oparl.org/schema/1.0/Paper`
- Aus `vendor:Drucksache` wird `http://www.vendor.de/oparl/schema/Drucksache`

TODO: Eventuell hier die Anforderung festhalten, dass jedes Objekt, das über eine OParl API ausgegeben wird, das `@type`-Schlüsselwort haben MUSS. Das ist noch nicht geklärt, da Listen hier eine Ausnahme bilden können.

Eine JSON-LD-konforme Ausgabe stellt noch weitere Anforderungen, von denen nachfolgend die wichtigsten zusammen gefasst werden.

- **Schlüssel MÜSSEN einzigartig sein**: Es ist nicht zulässig, in einem JSON-LD-Objekt mehrmals den selben Schlüssel für ein Attribut zu verwenden.
- **Groß- und Kleinschreibung werden unterschieden**: Groß- und Kleinschreibung MÜSSEN bei allen Bestandteilen eines JSON-LD-Dokuments unterschieden werden, also auch bei den Attributnamen.
- **Listen gelten grundsätzlich als nicht sortiert**: Die JSON-Spezifikation geht bei Listen grundsätzlich davon aus, dass diese eine Sortierung besitzen. Im Unterschied dazu gilt für JSON-LD, dass die Reihenfolge der Werte zwischen zwei eckigen Klammern `[` und `]` als zufällig gilt, sofern nicht anders spezifiziert. Wer einen JSON-LD-Objekttyp spezifiziert, kann jedoch mittels des Schlüsselwortes `@list` kennzeichnen, dass es sich hierbei um eine sortierte Liste handelt.

  Wo immer die OParl-Spezifikation eine stabile, nicht zufällige Sortierung von Listen erwartet, wird dies eigens erwähnt werden. Das OParl-JSON-LD-Vokabular wird an der entsprechenden Stelle das Schlüsselwort `@list` verwenden.
- **Listen DÜRFEN NICHT verschachtelt werden**: JSON-LD erlaubt keine Listen, die wiederum Listen als Werte enthalten.

4.5.3 JSONP

Eine Einschränkung bei der Nutzung von JSON ist das Sicherheitsmodell von Web-Browsern. Die gängigen Browser erlauben es innerhalb von Webanwendungen nicht, JSON-Ressourcen von Domains auszulesen, die nicht der Domain entsprechen, von der die Webanwendung selbst geladen wurde. AnwendungsentwicklerInnen sind dadurch bei der Implementierung von Client-Anwendungen eingeschränkt.

Diese Einschränkung gilt nicht fürt JSONP[28]. Durch JSONP (TODO: Abkürzung erläutern) wird die JSON-Notation so erweitert, dass der ausgegebene Code ausführbarer JavaScript-Code wird. Damit wird erreicht, dass der JSON-Code über die Grenzen von Domains hinweg direkt von Webanwendungen eingebunden werden kann.

Das folgende Beispiel verdeutlicht den Unterschied zwischen JSON und JSONP. Zunächst ein einfaches JSON-Beispiel:

```
{
    "foo": "bar"
}
```

Durch Einbettung in eine sogenannte Callback-Funktion wird daraus JSONP:

```
mycallback({
    "foo": "bar"
})
```

Der Name der Callback-Funktion (im Beispiel “mycallback”) wird grundsätzlich bei der Anfrage vom Client bestimmt, und zwar mittels URL-Parameter.

Für eine OParl-konforme Schnittstelle wird EMPFOHLEN, dass der Server die JSONP-Ausgabe unterstützt. Die JSONP-Ausgabe MUSS in diesem Fall für sämtliche Abfragen möglich sein. Eine JSONP-Unterstzung nur für bestimmte Anfragen ist nicht vorgesehen.

[28] TODO: URL zur Spezifikation

Der URL-Parameter, den Clients zur Aktivierung der JSONP-Ausgabe verwenden, MUSS `callback` lauten. Der Wert des `callback`-URL-Parameters MUSS vom Server unverändert als Callback-Funktionsname verwendet werden.

Aus Sicherheitsgründen MUSS der Client den Wert des `callback`-Parameters aus einem eingeschränkten Zeichenvorrat bilden, erlaubt sind ausschließlich die Klein- und Großbuchstaben von a bis z bzw. A bis Z sowie die Ziffern von 0 bis 9.

Hält sich der Client nicht an diese Einschränkung und wird ein `callback`-Parameter mit nicht erlaubten Zeichen verwendet, SOLL der Server die Anfrage mit einer HTTP XXX (Bad Request) Antwort bedienen. (TODO: Status Code einfügen oder prüfen, welche HTTP-Antwort die geeignetste ist.)

- TODO: Spezifikation finden/verlinken. (RFC gibt es nicht)
- https://github.com/OParl/specs/issues/67

4.6 Benannte und anonyme Objekte

Die JSON-LD-Spezifikation unterscheidet zwischen benannten und anonymen Objekten. Da die Unterscheidung auch für OParl von Bedeutung ist, wird sie hier genauer erläutert.

4.6.1 Benannte Objekte

Benannte Objekte sind innerhalb einer JSON-LD-Ausgabe diejenigen Objekte, die durch eine eigene URL identifiziert werden. Als Beispiel dient ein fiktives Objekt, das ein Client über die URL

`https://oparl.beispielris.de/bodies/0/committees/1`

abruft:

```
{
    "@id": "https://oparl.beispielris.de/bodies/0/committees/1",
    "@type": "http://oparl.org/schema/1.0/committee",
    "name": "Hauptausschuss"
}
```

Das Objekt enthält eine Eigenschaft `@id` mit der URL des Objekts als Wert.

Das benannte Objekt kann über seine URL sowohl eindeutig identifiziert als auch direkt abgerufen werden.

4.6.2 Anonyme Objekte (Blank Nodes)

Im Gegensatz dazu können Objekte existieren, die keine eigene URL haben.

Wenn diese im Semantic Web verwendet werden, dann führen sie zu erheblichen Problemen. Sandro Hawke (W3C) hat diese so zusammengefasst:

> In general, blank nodes are a convenience for the content provider and a burden on the content consumer. Higher quality data feeds use fewer blank nodes, or none. Instead, they have a clear concept of identity and service for every entity in their data.

> If someone in the middle tries to convert (Skolemize) blank nodes, it's a large burden on them. Specifically, they should provide web service for those new URIs, and if they get updated data from their sources, they're going to have a very hard [perhaps impossible] time understanding what really changed. (Zitiert nach http://richard.cyganiak.de/blog/2011/03/blank-nodes-considered-harmful/)

Ein Beispiel dafür findet sich in der Beratungsfolge einer Drucksache. Das nachfolgende Beispiel zeigt eine Drucksache, deren Beratungsfolge über die Eigenschaft `consultations` kodiert ist.

TODO: Nachstehendes Beispiel und Text dazu auf stimmiges Paper Objekt umschreiben.

```
{
    "@id": "https://oparl.beispielris.de/bodies/0/papers/456",
    "@type": "http://oparl.org/schema/1.0/paper",
    "title": "Beschlussvorlage zur Jugendförderung",
    "consultations": [
        {
            "@type": "http://oparl.org/schema/1.0/consultation",
            "committee": "https://oparl.beispielris.de/bodies/0/committees/1",
            "meeting": "https://oparl.beispielris.de/bodies/0/committees/1/meetings/123",
            "agendaitem": "7.2.4",
            "authoritative": false
        },
        {
            ...
        }
    ]
}
```

Die Eigenschaft `consultations` ist eine Liste mit einem oder mehreren Objekten vom Typ `consultation`. Diese Objekte spiegeln wieder, in welchen Sitzungen die vorliegende Drucksache beraten wurde bzw. wird.

Die einzelnen `consultation`-Objekte haben keine `@id`-Eigenschaft, daher handelt es sich dabei um anonyme Objekte, auch *Blank Nodes* genannt. Diese Objekte können nicht einzeln, sondern nur im Kontext verbundener Objekte, wie hier im Beispiel im Kontext einer Drucksache, abgerufen werden.

TODO: Weitere Objekttypen nennen, in denen Blank Nodes vorkommen.

4.7 Objektlisten

Über die OParl-API können entweder einzelne (benannte) Objekte, beispielsweise eine bestimmte Drucksache, oder Listen von Objekten, etwa die Liste aller Sitzungen einer Körperschaft, abgefragt werden.

Fragt ein Client eine Liste von Objekten an, hat der Server mehrere Möglichkeiten, diese Anfrage zu beantworten.

In jedem Fall werden die einzelnen Objekte, die Bestandteile der Liste sind (wie z.B. die einzelnen Drucksachen) durch die URL des jeweiligen Objekts repräsentiert. Objektlisten sind also tatsächlich immer Listen von URLs.

4.7.1 Vollständige Listenausgabe

In der einfachsten Form gibt der Server die Liste als Objekt mit nur einer einzigen Eigenschaft `items` aus. Der Wert dieser Eigenschaft ist die **vollständige Liste** der URLs aller in der Liste enthaltenen Objekte.

Diese einfachste Form der Antwort eignet sich nur für Listen mit einer begrenzten Anzahl von Einträgen, wie beispielsweise die Liste der Mitglieder einer Organisation.

Beispiel:

```
{
    "items": [
        "https://beispielris.de/bodies/0/papers/2",
        "https://beispielris.de/bodies/0/papers/5",
        "https://beispielris.de/bodies/0/papers/7",
    ]
}
```

Die vollständige Listenausgabe SOLL nur für Listen verwendet werden, die bis zu 100 Einträge umfassen.

4.7.2 Paginierung

Für den Abruf von Listen mit vielen Elementen ist eine Blätterfunktion (Paginierung) vorgesehen. Darunter verstehen wir die Aufteilung einer Liste in kleinere Teilstücke, die wir hier als Listenseiten bezeichnen. Jede Listenseite wird vom Client jeweils mit einer eigenen API-Anfrage abgerufen. Das dient dazu, die bei der jeweiligen Anfrage übertragenen Datenmengen zu begrenzen und damit Antwortzeiten und Systemressourcen sowohl beim Server als auch beim Client zu schonen.

Die Entscheidung, ob eine Seite teilweise und daher mit Paginierung ausgegeben wird, liegt allein beim Server. Bei Listen mit mehr als 100 Einträgen ist dies EMPFOHLEN. Die Zahl der Einträge, die der Server dabei je Listenseite ausliefert, SOLL dabei maximal 100 betragen. Die Anzahl der Einträge (Obergrenze) MUSS auf allen Listenseiten der selben Liste einheitlich sein, sofern nicht (beispielsweise auf der letzten Listenseite) weniger Listeneinträge vorhanden sind.

Wird eine Liste in Form von Listenseiten, also mit Paginierung, ausgegeben, teilt der Server dem Client mittels HTTP-Headern in der nachfolgend beschriebenen Form mit, unter welcher URL weitere Listenseiten abgerufen werden können. Dabei wird von dem in RFC5988[29] beschriebenen Header namens `Link` Gebrauch gemacht.

Beispiel eines Link-Headers zur Angabe der URL für den Abruf der folgenden Listenseite:

```
Link: <https://oparl.beispielris.de/bodies/0/papers/?skip_id=7>;rel=next
```

Im oben gezeigten Beispiel besteht der Wert des Link-Headers aus zwei Bestandteilen, die durch Semikolon `;` von einander getrennt sind:

```
<https://oparl.beispielris.de/bodies/0/papers/?skip_id=7>;rel=next
 \_____________________________________________________/  \______/
                           |                                  |
                          URL                          Link-Parameter
```

Die Bestandteile sind:

URL: Die URL zum Abruf der nächsten Listenseite. Die URL wird in spitzen Klammern `<` und `>` ausgegeben. *Hinweis:* Wie diese URL aufgebaut ist, entscheidet allein der Server. Hier wird lediglich ein fiktives Beispiel gegeben.

[29] RFC5988: http://tools.ietf.org/html/rfc5988

Link-Parameter: Gemäß RFC5988 können belibig viele, auch null, Link-Parameter hinter der URL ausgegeben werden, jeweils durch ein Semikolon von der URL getrennt. Für OParl gilt: Es MUSS bei einer Liste mit Paginierung genau ein Link-Header mit dem Link-Parameter `rel=next` gegeben sein, sofern es eine nächste Seite mit weiteren Listenelementen gibt. Stellt die mit der aktuellen Anfrage ausgegebene Listenseite das Ende der Liste dar, DARF die Anfrage NICHT den Link-Header mit Link-Parameter `rel=next` enthalten.

Es ergibt sich eine typische Abfolge, wie Clients bei Bedarf mit mehreren Anfragen ganze Objektlisten vom Server abrufen:

1. Der Server stellt eine URL für eine Liste zur Verfügung.
2. Der Client ruft diese URL der Liste auf.
3. Der Server antwortet mit einer Listenseite und stellt im Link-Header mit Link-Parameter `rel=next` die URL für den Abruf der nächsten Listenseite zur Verfügung.
4. Der Client ruft die im Link-Header übergebene URL für die nächste Listenseite auf.

Die Punkte 3 und 4 können sich nun so oft wiederholen, bis die letzte Listenseite erreicht ist.

5. Der Server liefert die letzte Listenseite ohne Link-Header aus.

Zusätzlich zu dem für die Paginierung obligatorischen Link-Header für die folgende Listenseite (Link-Parameter `rel=next`) können Server OPTIONAL weitere Link-Header zum Abruf bestimmter Listenseiten anbieten:

Erste Listenseite (`first`): Sofern die aktuell abgerufene Listenseite nicht den Anfang der Liste wiedergibt, KANN der Server einen Link-Header mit Link-Parameter `rel=first` und der URL zum Abruf der *ersten* Listenseite ausgeben.

Letzte Listenseite (`last`): Sofern die aktuell abgerufene Listenseite nicht das Ende der Liste wiedergibt, KANN der Server einen Link-Header mit Link-Parameter `rel=last` und der URL zum Abruf der *letzten* Listenseite ausgeben.

Vorherige Listenseite (`prev`): Sofern die aktuell abgerufene Listenseite nicht den Anfang der Liste wiedergibt, KANN der Server einen Link-Header mit Link-Parameter `rel=prev` und der URL zum Abruf der ersten Listenseite ausgeben.

Damit eröffnet der Server dem Client zusätzliche Möglichkeiten, die einzelnen Listenseiten abzurufen.

Server-Implementierer entscheiden selbst, wie die URLs zum Abruf einzelner Listenseiten aufgebaut ist und tragen damit selbst Verantwortung für die Funktionsweise der Paginierung. Bei der Entscheidung für eine Form der Implementierung sollten die folgenden Anforderungen von Clients berücksichtigt werden.

Es ist davon auszugehen, dass Clients für den gesamten Abruf aller Seiten einer Liste längere Zeit benötigen. In der Zwischenzeit kann sich der Inhalt der Liste bereits ändern, etwa durch das Hinzukommen neuer Einträge. Die Paginierung ist idealerweise so zu implementieren, dass sich das Hinzukommen oder Entfernen von Einträgen möglichst nicht auf einen Client auswirkt, der aktuell die Liste paginiert, um alle Einträge abzurufen. Wir bezeichnen dies als **stabile Paginierung**.

Eine wesentliche Anforderung an Listen mit Paginierung ist, dass alle Einträge der Liste in einer konstistenten Reihenfolge sortiert ausgegeben werden SOLLEN. Das bedeutet, dass die Sortierung beim Server im Idealfall anhand einer eindeutigen und unveränderlichen Objekteigenschaft vorgenommen wird. Hierfür eignen sich die Objekt-URLs, da sie genau diese beiden Anforderungen erfüllen sollten.

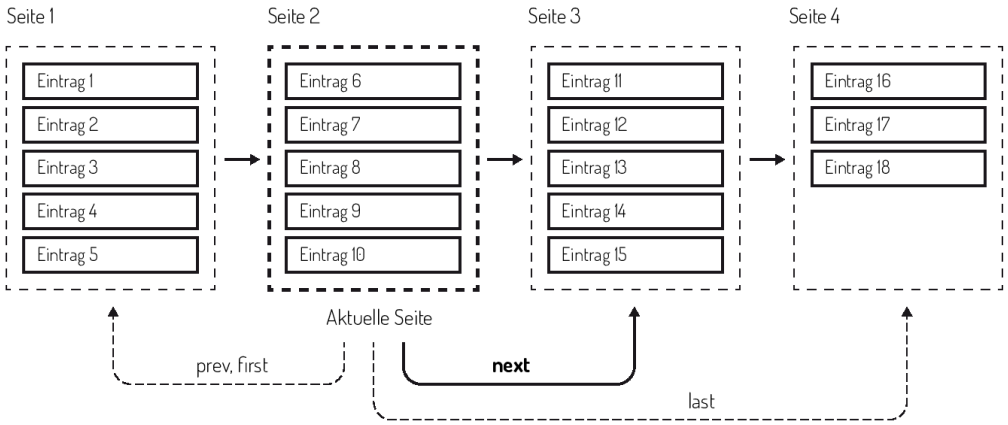

Abbildung 2: Paginierung: Schematische Darstellung

Über die Sortierung hinaus können bei der Implementierung einer stabilen Paginierung auf Server-Seite weitere Überlegungen einbezogen werden. Zur Verdeutlichung soll hier eine ungünstige (unstabile) Form der Implementierung mit Hilfe einer SQL-Abfrage illustriert werden. Gegeben sei eine Tabelle `example`, die einen numerischen Primärschlüssel `id` enthält. Nehmen wir an, die erste Seite der Liste wird mit der Abfrage

```
SELECT * FROM example ORDER BY id LIMIT 10 OFFSET 0
```

abgerufen und würde 10 Datensätze mit den `id`s 1 bis 10 zurück liefern. Dann wird die zweite Seite mit der Abfrage

```
SELECT * FROM example ORDER BY id LIMIT 10 OFFSET 10
```

abgerufen. Sollte nach der ersten, aber vor der zweiten Abfrage beispielsweise der Datensatz mit der `id=1` gelöscht worden sein, liefert die zweite Abfrage Datensätze mit `id` > 9. In diesem Fall würde dies nur dazu führen, dass ein Datensatz (`id=10`) zweimal ausgegeben wird. Bei ungünstigeren Konstellationen wäre auch denkbar, dass eine unstabile Paginierung bewirkt, dass einzelne Datensätze beim Paginieren übergangen werden.

Besser wäre es, bei der Paginierung die Eintragsgrenze, bei der eine Listenseite beginnen soll, explizit zu benennen. Wurden auf der ersten Listenseite die Datensätze mit den IDs 1 bis 10 ausgegeben, so könnte der Folgeaufruf, um beim SQL-Beispiel zu bleiben, so aussehen:

```
SELECT * FROM example WHERE id > 10 ORDER BY id LIMIT 10
```

Die zuvor beschriebenen Anforderungen für die Paginierung von Listen gelten auch unverändert, wenn der Umfang der Liste durch Abfrageparameter vom Client eingeschränkt wurde.

4.7.3 Listen als Eigenschaften von Objekten

Listen von Objekten können auch als Werte von Objekteigenschaften auftreten. Hierbei kann die oben beschriebene Paginierung nicht angewendet werden, sondern es MÜSSEN die URLs aller Listeneinträge aufgelistet werden.

Ein Beispiel dafür könnte die Eigenschaft `bodies` des `oparl:System` Objekts sein, also die Liste der Körperschaften, die auf einem OParl-Server abgebildet werden:

```
{
    "@type": "oparl:System",
    "@id": "https://oparl.beispielris.de/",
    "bodies": [
        "https://oparl.beispielris.de/bodies/1",
        "https://oparl.beispielris.de/bodies/2"
    ],
    ...
}
```

Diese Listenausgabe direkt im Objekt ist nur zu empfehlen, wenn die Anzahl der Einträge klein (weniger als 100 Einträge) ist. Sind mehr Einträge zu erwarten, SOLL die entsprechende Liste über eine eigene URL angeboten werden.

Das folgende Beispiel verdeutlicht, wie dies die Eigenschaft `bodies` am vorangegangenen Beispiel beeinflusst:

```
{
    "@type": "oparl:System",
    "@id": "https://oparl.beispielris.de/",
    "bodies": "https://oparl.beispielris.de/bodies/"
    ...
}
```

Statt einer JSON-Liste ist der Wert der Eigenschaft nun eine einzige URL.[30]

4.8 Feeds

Feeds sind spezielle Arten von Objektlisten, für die besondere Anforderungen gelten. Es werden drei verschiedene Feeds spezifiziert:

- Der Feed *Neue Objekte*
- Der Feed *Geänderte Objekte*
- Der Feed *Entfernte Objekte*

Der Begriff "Feed" ist eine Anlehnung an die weit verbreiteten RSS-[31] oder Atom-Feeds[32], deren Publikationslogik im Wesentlichen auf der chronologischen Sortierung beruht. Im Unterschied zu Atom oder RSS ist hier jedoch keine XML-Ausgabe beabsichtigt.

Die Feeds sollen es Clients ermöglichen, schnell und ressourcenschonend abzufragen, welche Objekte auf dem Server neu hinzugefügt, geändert oder entfernt wurden. Damit können Clients beispielsweise schnell und einfach neue Dokumente auffinden und verarbeiten oder entfernte Objekte aus ihren Caches entfernen und dabei nur ein Mindestmaß an Anfragen ausführen.

Ein OParl-Server SOLL jeden der nachfolgend beschriebenen Feeds anbieten, sofern möglich.

Für alle drei Feeds wird EMPFOHLEN, dass mindestens ein Zeitraum von 365 Tagen abgedeckt wird.

Da Feeds üblicherweise eine große und stetig steigende Anzahl von Objekten beinhalten können, ist hier die Paginierung anzuwenden, wie sie im vorigen Abschnitt über Objektlisten beschrieben wird.

[30] Wie in vielen anderen Fällen ist diese URL hier wieder nur beispielhaft. Die tatsächliche Gestaltung des Pfads bestimmt der Server-Implementierer.
[31] RSS 2.0 Specification: http://cyber.law.harvard.edu/rss/rss.html
[32] Atom ist in RFC4287 spezifiziert: http://www.ietf.org/rfc/rfc4287.txt

4.8.1 Der Feed "Neue Objekte"

Der Feed für neue Objekte listet die URLs neu hinzugekommener Objekte in der Reihenfolge des Datums ihrer Erstellung, wobei die jüngsten Objekte zuerst ausgegeben werden.

Die Definition, was ein "neues" Objekt bzw. die "Erstellung" bedeutet, kann zwischen Systemen und Objekttypen variieren. So werden bestimmte Objekte in einigen Systemen zunächst erstellt und erst dann für die Öffentlichkeit freigegeben. In diesem Fall ist im Sinne dieses Feeds die Freigabe als Zeitpunkt der Erstellung zu verwenden.

Der Feed SOLL sämtliche Objekttypen umfassen, die in einem System geführt werden.

Das nachstehende Beispiel zeigt die mögliche Ausgabe des Feeds:

```
{
    "items": [
        {
            "@id": "https://oparl.beispielris.de/bodies/0/papers/21/documents/3",
            "created": "2014-01-07T12:59:01.038+0100"
        },
        {
            "@id": "https://oparl.beispielris.de/bodies/0/papers/21",
            "created": "2014-01-05T18:29:37.123+0100"
        },
        {
            "@id": "https://oparl.beispielris.de/bodies/0/papers/20/documents/5",
            "created": "2014-01-04T11:26:48.638+0100"
        },
        ...
    ]
}
```

Wie im Beispiel zu sehen ist, enthält die Eigenschaft `items` eine Liste mit unbenannten Objekten. Dies ist ein Unterschied zu herkömmlichen Objektlisten, bei denen an dieser Stelle lediglich URLs als Listeneinträge erwartet werden.

Jedes der Objekte in der `items`-Liste MUSS seinerseits wiederum zwei Eigenschaften besitzen:

- `@id`: Die URL des neuen Objekts
- `created`: Der Zeitpunkt der Erzeugung des Objekts

Der jeweils in der Eigenschaft `created` ausgegebene Zeitpunkt SOLL vom Server als Sortierkriterium der Liste genutzt werden. So können Clients den jeweils am Anfang der Liste vorgefundenen Zeitpunkt als Begrenzung für die zukünftige Abfrage des Feeds nutzen. Ein Beispiel zur Erläuterung:

Am 1. April 2014 ruft ein Client den Feed ab und findet im ersten Listeneintrag den `created`-Zeitpunkt `2014-03-31T18:02:34.058+0200` vor, den er sich als Grenzwert merkt. Beim nächsten Abruf des Feeds einige Tage später muss der Client die Liste nur so weit abarbeiten, so lange der `created`-Zeitpunkt der Einträge größer oder gleich dem Grenzwert ist.

4.8.2 Der Feed "Geänderte Objekte"

Der Feed für geänderte Objekte listet die URLs geänderter Objekte in der Reihenfolge des Datums ihrer Änderung, wobei das zuletzt geänderte Objekt zuerst ausgegeben wird.

Die Definition einer "Änderung" kann sich zwischen den Objekttypen unterscheiden. Tendenziell soll die Definition eher weiter ausgelegt werden, als enger. Als Änderung einer Organisation könnte es beispielsweise verstanden werden, wenn ein neues Mitglied zur Organisation hinzukommt. Das Erstellen eines Objekts (im Sinne des Feeds "Neue Objekte")

sollte hingegen nicht als Änderung gewertet werden, um das redundante Erscheinen eines neuen Objekts sowohl im Feed "Neue Objekte" als auch im Feed "Geänderte Objekte" zu vermeiden.

Auch hier SOLL der Feed sämtliche Objekttypen umfassen, die in einem System geführt werden.

```
{
    "items": [
        {
            "@id": "https://oparl.beispielris.de/bodies/0/papers/0/documents/2",
            "lastModified": "2014-01-08T14:28:31.568+0100"
        },
        {
            "@id": "https://oparl.beispielris.de/bodies/0/papers/0",
            "lastModified": "2014-01-08T12:14:27.958+0100"
        },
        {
            "@id": "https://oparl.beispielris.de/bodies/0/papers/0/documents/1",
            "lastModified": "2014-01-06T17:01:00.402+0100"
        },
        ...
    ]
}
```

Das Ausgabeformat entspricht weitgehend dem des Feeds "Neue Objekte", jedoch heißt hier die Eigenschaft für den Zeitpunkt der letzten Änderung `lastModified`. Auch hier gilt, dass der als `lastModified` ausgegebene Zeitpunkt auch als Sortierkriterium der Liste gelten SOLL.

4.8.3 Der Feed "Entfernte Objekte"

Der Feed für entferne Objekte listet die URLs entfernter Objekte in der Reihenfolge des Datums ihrer Entfernung auf, wobei die zuletzt entfernten Objekte zuerst ausgegeben werden.

Mit "Entfernung" ist im Sinne dieses Feeds die Löschung eines Objekts, aber auch die Depublikation oder das Beenden der öffentlichen Verfügbarkeit gemeint.

Client-Implementierer sind angehalten, diesen Feed zu nutzen, um beispielsweise depublizierte Dokumente aus ihren lokalen Caches zu entfernen.

```
{
    "items": [
        {
            "@id": "https://oparl.beispielris.de/bodies/0/people/22",
            "removed": "2013-11-11T11:11:00.000+0100"
        },
        ...
    ]
}
```

Die Eigenschaft zur Angabe des Entfernungszeitpunkts heißt hier `removed` und SOLL, analog zu den beiden anderen Feeds, als Sortierkriterium der Liste verwendet werden.

4.9 Dateizugriff

Mit dem Begriff "Datei" sind im Sinne dieser Spezifikation alle Ressourcen gemeint, die von einem OParl-Server zur Verfügung gestellt werden und deren Metadaten über die JSON-API

als `oparl:Document` abgerufen werden können. Es handelt sich dabei beispielsweise um Textdokumente im PDF-Format, Abbildungen im JPEG- oder PNG etc., die wesentliche Inhalte der parlamentarischen Informationen im OParl-System ausmachen.

In Bezug auf die Datenvolumen, die der Verkehr zwischen OParl Servern und Clients ausmacht, kommt dem Dateizugriff eine besondere Bedeutung zu. Daher formuliert OParl diesbezüglich einige Anforderungen, die helfen sollen, unnötigen Datentransfer zu vermeiden.

Detail zu sämtlichen angesprochenen Mechanismen sind in der HTTP-1.1-Spezifikation[33] zu finden.

4.9.1 GET und HEAD Anfragen

Grundsätzlich gilt, dass jede Datei mittels HTTP-Anfrage unter Verwendung der HTTP-Methode `GET` abrufbar sein MUSS. Um Clients zusätzlich die Überprüfung einer Datei zu ermöglichen, MUSS vom Server außerdem die HTTP-Methode `HEAD` unterstützt werden. Gemäß HTTP-Spezifikation gibt der Server in diesem Fall nur die Antwort-Header, nicht aber den eigentlichen Inhalt der angefragten Ressource, aus.

Die URLs zum Abruf der einzelnen Datei (wahlweise mittels GET oder HEAD) stellt der Server dem Client in den Daten des Metadaten-Objekts zur Verfügung. Details finden sich in der Schema-Beschreibung zu `oparl:Document`.

4.9.2 Allgemeiner Zugriff und expliziter Download

Mit der im `oparl:Document` ZWINGEND anzugebenden Eigenschaft `accessUrl` liefert der Server dem Client eine URL, die wir hier nachfolgend als *Zugriffs-URL* bezeichnen. Diese URL dient dem allgmeinen Zugriff auf die Datei. Wie der Client dem Endnutzer diesen Zugriff genau ermöglicht, ist nicht Sache der OParl-Spezifikation.

Im Unterschied dazu KANN der Server dem Client in der Eigenschaft `downloadUrl` eine weitere URL anbieten, hier *Download-URL* genannt. Diese dient im Gegensatz zur Zugriffs-URL speziell zum Herunterladen und Speichern der Datei in einem Dateisystem des Endnutzers. Bei Zugriff auf die Download-URL MUSS der Server in der HTTP-Antwort einen `Content-Disposition` Header senden.[34] Dieser Header MUSS als ersten Parameter den Typ `attachment` sowie den `filename`-Parameter mit dem Namen der Datei enthalten.

Beispiel:

```
Content-Disposition: attachment; filename="2014-08-22 Rat Wortprotokoll.pdf"
```

Der in diesem Header kommunizierte Dateiname ist als Vorschlag an die Nutzerin zu verstehen, die Datei unter diesem Namen zu speichern. Entsprechend sind Abwägungen bezüglich der Verständlichkeit, Leserlichkeit und Einzigartigkeit des Dateinamens, aber auch in Hinblick auf den verwendeten Zeichenumfang zu berücksichtigen. Es wird EMPFOHLEN, den Dateinamen ausschließlich aus dem ASCII-Zeichenvorrat zu bilden.

Im Unterschied zum Zugriff auf die Download-URL DARF der Server beim Zugriff auf die Zugriffs-URL KEINEN `Content-Disposition` Header mit Parameter `attachment` senden.

4.9.3 Obligatorische und empfohlene Header

Ziel ist, dem Client möglichst flexible Möglichkeiten zu geben, einen Cache zu überprüfen bzw. zu aktualisieren und vermeidbare Anfragen einer Ressource zu vermeiden. Um dies zu unterstützen, können laut HTTP-Spezifikationen unterschiedliche Header zum Einsatz kommen.

[33] vgl. http://www.w3.org/Protocols/rfc2616/rfc2616.html
[34] vgl. RFC2138 http://www.ietf.org/rfc/rfc2183

Die Auslieferung eines `Last-Modified`-Headers gilt für alle OParl-Server beim Zugriff auf eine Datei-URL, sei es Download- oder Zugriffs-URL, als ZWINGEND.

Darüber hinaus EMPFEHLEN wir, bei Anfrage einer Datei die folgenden Header auszuliefern:

- `Content-Length`: Die Größe des Dateiinhalts
- `ETag`: Entity Tag

4.9.4 Conditional GET

Unter einem "Conditional GET" versteht man im HTTP-Kontext die Möglichkeit des Clients, die Anfrage einer Ressource mit einer Bedingung zu verknüpfen. Der Server beantwortet die Anfrage nur dann mit einer vollständigen HTTP-Antwort, wenn die Bedingung erfüllt ist. Andernfalls enthält die Anfrage lediglich Header, der HTTP Status-Code SOLL in diesem Fall "304" lauten (für "nicht geändert"). Dies dient der Schonung von Ressourcen.

Für einen OParl-Server wird EMPFOHLEN, die nachstehenden Varianten des Conditional GET zu unterstützen:

- `If-Modified-Since`: Der Client sendet mit der Anfrage als Bedingung ein Datum. Nur wenn die angefragte Datei zuletzt *nach* diesem Datum geändert wurde, wird der Dateiinhalt mit der Antwort ausgeliefert.
- `If-None-Match`: Erlaubt die Formulierung der Bedingung anhand eines Entity-Tags.

4.9.5 Zustandsloser Dateizugriff

Die Anforderung, dass die OParl API zustandslos arbeitet (vgl. RESTful{#restful}), hat ZWINGEND auch für den Abruf von Dateien zu gelten. Es DÜRFEN daher keine Session-spezifischen URLs oder ähnliches für den Dateizugriff gebildet werden.

Damit wird erreicht, dass Clients die Zugriffs-URLs aus dem `oparl:Document` für längere Zeit speichern bzw. cachen können

4.9.6 Weiterleitungen

Es ist im Rahmen dieser Spezifikation problemlos möglich, die Anfrage an eine Datei-URL mit einer HTTP-Weiterleitung zu beantworten, um dem Client eine andere URL zum Zugriff mitzuteilen.

In diesem Fall wird dringend EMPFOHLEN, die Unterscheidung der Bedeutung der HTTP-Status-Codes `301` und `307` zu beachten.

- `301` SOLL verwendet werden, wenn die vom Client angefragte URL auch zukunftig nicht mehr gültig sein wird. Clients erhalten damit das Signal, die bisherige URL zu verwerfen und zukünftig die neue, vom Server in der Antwort mitgeteilte zu verwenden.
- `307` SOLL verwendet werden, wenn die vom Client genutzte URL nur temporär auf eine bestimmte andere URL weiter leitet. Clients werden so aufgefordert, die vorhandene URL auch bei zukünftigen Anfragen zu nutzen.

4.9.7 Entfernte Dateien

Beim Zugriff auf eine Datei, die zuvor einmal abrufbar war, es inzwischen jedoch nicht mehr ist, SOLL die HTTP-Antwort des Servers den spezifischen Status-Code `410` tragen.

4.10 Content Negotiation

Das Prinzip "Content Negotiation" wird in RFC2295[35] beschrieben und bedeutet, dass WWW-Server eine Ressource in verschiedenen Formaten bereithalten können und Clients die Möglichkeit haben, eine Vorliebe für ein bestimmtes Format zu übermitteln. Auch die HTTP-1.1-Spezifikation[36] schließt Content Negotiation ein.

Die Idee hinter Content Negotiation ist, dass ein Client die von ihm akzeptierten Repräsentationen im `Accept`-Header der HTTP-Anfrage mitsendet, damit der Server gemäß Spezifikation die am besten passende und von ihm unterstützte Repräsentation an den Client ausliefert.

Grundanforderung der vorliegenden Spezifikation an OParl Clients ist, dass sie bei jeder Anfrage an einen OParl-Server einen Accept-Header mit dem Mime-Type `application/ld+json` senden MÜSSEN, es sei denn, es handelt sich um einen Dateizugriff.

Im Kontext von OParl soll durch Unterstützung von Content Negotiation ermöglicht werden, dass die URLs von OParl-Objekten auch von WWW-Clients aufgerufen werden können, die nicht unmittelbar in Kenntnis der OParl-Spezifikation entwickelt wurden.

Ein Beispiel für einen solchen Client wäre ein üblicher Browser. Ruft dieser die URL einer Drucksache (OParl-Objekttyp `oparl:Paper`) auf, sendet er entweder keinen `Accept`-Header oder aber einen solchen, der eine Bevorzugung von Inhaltstypen wie HTML angibt.

Der Server DARF nun, da kein Accept-Header mit dem Typ `application/ld+json` gesendet wurde, dem Client eine alternative Version der Information über die Drucksache ausliefern, beispielsweise eine HTML-Ansicht.

Ein Server DARF eine alternative Inhaltsform auch in Form einer HTTP-Weiterleitung anbieten.

4.10.1 httpRange-14

In dem Zusammenhang sei informell auf ein für Linked Data relevantes Detail hingewiesen - auf welches man gelegentlich unter dem Kürzel `httpRange-14` stößt. Wenn man in einem üblichen Web-Browser diese URL aufruft:

`http://dbpedia.org/resource/John_Doe_(musician)`

dann erfolgt eine Weiterleitung auf diese URL:

`http://dbpedia.org/page/John_Doe_(musician)`

Dabei identifiziert die erste URL den abstrakten Begriff des Musikers, während die zweite eine Repräsentation dieses Begriffs identifiziert. Ein anderer HTTP-Client, der statt HTML-Seiten z.B. JSON-LD bevorzugt, würde bei Zugriff auf die erste URL wiederum zu einer anderen URL weiter geleitet:

`http://dbpedia.org/data/John_Doe_(musician).jsonld`

Dabei handelt es sich um eine Repräsentation des Musikers in Form von JSON-LD.

Diese drei Links sind nicht nur verschieden, sondern haben unterschiedliche Bedeutungen. Eine Angabe z.B. der zweiten URL (HTML-Seite) oder der dritten URL (JSON-LD) kann von Clients generell nicht verwendet werden, um auf einfache Weise zu den anderen beiden URLs zu gelangen. Deshalb soll bei Linked Data der erste Link, also der des abstrakten Begriffs angegeben werden. Der Zugriff auf diesen Link wird "Dereferenzierung" genannt. Von dem ersten Link gelangt man zu den beiden anderen mittels Content Negotiation.

So kann der Server beim Zugriff eines Client auf

`https://dbpedia.org/resource/John_Doe_(musician)`

[35] RFC2295: http://tools.ietf.org/html/rfc2295
[36] RFC2616: http://tools.ietf.org/html/rfc2616

z.B. entweder diese URL sowie HTML-Inhalt liefern

`https://dbpedia.org/page/John_Doe_(musician)`

oder aber diese URL mit JSON-LD als Inhalt

`https://dbpedia.org/data/John_Doe_(musician).jsonld`

Die Entscheidung darüber, welche der URLs und welche der Inhaltsformate der Server liefert, wird zwischen Client und Server mittels Content Negotiation ausgehandelt.

4.11 Ausnahmebehandlung

TODO:

(Diskussion hierzu unter https://github.com/OParl/specs/issues/89)

4.12 Liste reservierter URL-Parameter

Die in dieser Liste enthaltenen Zeichenketten haben eine reservierte Bedeutung und stehen bei Implementierungen eines OParl-Servers nicht mehr für die freie Verwendung in URLs zur Verfügung.

callback: Mit diesem Parameter wird die JSONP-Ausgabe aktiviert. Mehr dazu im Abschnitt JSONP.

startdate: Parameter für die Einschränkung einer Abfrage anhand eines Datums bzw. einer Zeitangabe.

enddate: Parameter für die Einschränkung einer Abfrage anhand eines Datums bzw. einer Zeitangabe.

- (Parameter für Datums-/Zeitbereichsfilter)

5 Schema

Dieses Kapitel beschreibt das Schema von OParl. Das Schema bildet das Datzenmodell der OParl-Architektur ab. Es definiert, welche Objekttypen über eine OParl-API abgerufen werden können und welche Eigenschaften diese Objekttypen haben dürfen und müssen. Darüber hinaus ist im Schema auch festgelegt, in welcher Beziehung verschiedene Objekttypen zu einander stehen.

5.1 Übergreifende Aspekte

5.1.1 Unicode-Zeichenketten als Standard

Wenn in der nachfolgenden Schema-Beschreibung nicht anders angegeben, werden bei den Werten grundsätzlich Unicode-Zeichenketten (Strings) erwartet.

5.1.2 null-Werte

JSON erlaubt es grundsätzlich, dass Eigenschaften den Wert `null` haben können. Im Rahmen dieser Spezifikation DARF das jedoch nur bei Eigenschaften der Fall sein, die als OPTIONAL oder EMPFOHLEN gekennzeichnet sind. ZWINGENDE Eigenschaften müssen einen Wert ungleich `null` besitzen.

5.1.3 Datums- und Zeitangaben

Für Datum und Zeit werden die in XML Schema festgelegten Typen verwendet (was nicht bedeutet, dass in OParl XML verwendet wird).

Für ein Datum wird http://www.w3.org/TR/xmlschema-2/#date verwendet und für eine Zeit http://www.w3.org/TR/xmlschema-2/#dateTime. Dabei wird ein Datum (ein Tag ohne Uhrzeit) ohne Zeitzone und ein Datum mit Zeit mit Zeitzone angegeben, denn nur damit ist die Uhrzeit weltweit eindeutig ohne zusätzlich auf den Ort einer Sitzung o.ä. Bezug nehmen zu müssen.

Diese Spezifikationen stützen sich auf RFC 3339 (http://www.ietf.org/rfc/rfc3339.txt) und RFC 3339 wiederum auf ISO 8601.

Im JSON-LD Kontext von OParl ist der Präfix 'xsd' so spezifiziert, dass Datums- und Zeittyp durch 'xsd:date' bzw. 'xsd:dateTime' abgekürzt werden können.

5.1.4 Mehrsprachigkeit

Für Texte ist durchgehend vorgesehen, dass diese mehrsprachig sein können. Kommunale Anbieter von OParl-Daten in Deutchland müssen aus gesetzlichen Gründen auf jeden Fall die deutsche Sprache unterstützten. Die Unterstützung anderer Sprachen ist dagegen optional. Deshalb wird grundsätzich durch ~~ "@language": "de", ~~ im Kontext die deutsche Sprache als Vorgabe eingestellt.

Es gibt aber möglicherweise auch Zeichenketten, für die keine Mehrsprachigkeit vorgesehen wird. Dazu gehören z.B. Personennamen.

TODO: Stimmt nicht. Tamilische Namen verwenden in der Originalschreibweise sogar ein vollkommen anderes Alphabet.

5.1.5 Präfixe in Kontexten

Die Beispiel-Kontexte verwenden eine Reihe von Präfixen. Diese sind hier zusammengestellt und werden in den einzelnen Beispiel-Kontexten nicht jeweils wiederholt:

```
"beispielris": "https://oparl.beispielris.de",
"oparl": "http://oparl.org/TODO/",
"dc": "http://purl.org/dc/terms/",
"foaf": "http://xmlns.com/foaf/0.1/",
"skos": "http://www.w3.org/2004/02/skos/core#",
"vcard": "http://www.w3.org/2006/vcard/ns#",
"xsd": "http://www.w3.org/2001/XMLSchema#",
"ogc": "http://www.opengis.net/ont/geosparql#",
```

5.1.6 Herstellerspezifische Erweiterungen

Diese sind - falls tatsächlich erforderlich - mit den JSON-LD Mitteln einfach möglich. z.B.

```
"herstellera:newWonderProperty": "Dies ist ein Feature
    welches noch kein anderer Hersteller bietet!"
```

5.1.7 URL-Pfade in den Beispielen

OParl-Clients wissen *nichts* vom Aufbau von Pfaden innerhalb von URLs, müssen dies nicht wissen und es gibt deshalb in der OParl-Spezifikation *keine* Festlegungen dazu.

Wenn der Betreiber eines OParl-Systems beispielsweise meint, dass eine Person eine eigene Domain verdient, dann ist dies aus Sicht der OParl-Spezifikation völlig in Ordnung:

`https://ratsmitglied-max-mustermann.beispielris.de/mein-oparl-datensatz`

Noch etwas extremer: selbst eine eigene Domain für jedes einzelne OParl-Objekt würde der OParl-Spezifikation nicht widersprechen.

Wenn also in einer Beispiel-URL so etwas wie

`bodies/0/peoples/`

auftaucht, dann bedeutet das nicht, dass genau solche Pfade durch die OParl-Spezifikation vorgeschrieben sind.

Auch dies wäre als absoluter Link z.B. für eine Person verwendbar:

`https://www.ratsinfomanagement.net/personen/?__=LfyIfvCWq8SpBQj0MiyHaxDZwGJ`

Dies käme dann als relativer Link für die Person in Frage:

`personen/?__=LfyIfvCWq8SpBQj0MiyHaxDZwGJ`

oder auch z.B. dies

LfyIfvCWq8SpBQj0MiyHaxDZwGJ

Gleichzeitig ist aber aus verschiedenen Gründen ein strukturierter Aufbau der Pfade durchaus sinnvoll, der sich an der Hierarchie der Objekte orientiert (nicht zuletzt, weil dies Softwareentwicklern während der Entwicklung helfen kann). Dennoch wird eine solche Struktur bewusst nicht in OParl festgelegt.

5.2 Eigenschaften mit Verwendung in mehreren Objekttypen

5.2.1 `@id`

URL des Objekts und eindeutiges Identifikationsmerkmal. Siehe dazu auch “Benannte Objekte”. Dies ist ein ZWINGENDES Merkmal für jedes Objekt.

5.2.2 `@type`

Objekttypenangabe des Objekts. ZWINGEND für jedes Objekt.

5.2.3 `name` und `nameLong`

Beide Eigenschaften können bei vielen Objekttypen genutzt werden, um den nutzerfreundlichen Namen des Objekts anzugeben. Üblicherweise ist `name` eine Pflichteigenschaft, während nameLong optional angegeben werden kann. Dies ist dann zu empfehlen, wenn zu einem Namen eine kurze bzw. kompakte und eine längere, aber weniger nutzerfreundliche Variante existieren. Ein Beispiel wäre die Kurzform “CDU” für den Parteinamen “Christlich Demokratische Union Deutschlands”.

In keinem Fall sollten die Werte von `name` und `nameLong` identische sein.

5.2.4 `license`

Die Eigenschaft `license` erlaubt es, am jeweiligen Objekt die URL einer Lizenz anzugeben. Damit wird gekennzeichnet, welche Lizenz der Veröffentlicher der Daten für das jeweilige Objekt vergibt.[37]

Eine besondere Bedeutung hat die Eigenschaft `license`, wenn sie am `oparl:System` Objekt oder am `oparl:Body` Objekt vergeben wird. Die hier angegebene Lizenzinformation sagt aus, dass alle Objekte dieses Systems bzw. der Körperschaft unter der angegebenen Lizenz veröffentlicht werden, sofern dies nicht am jeweiligen Objekt mit einer anders lautenden Lizenz-URL überschrieben wird. Daher wird dringend EMPFOHLEN, die Lizenzinformation global am `oparl:System` Objekt mitzuteilen und auf redundante Informationen zu verzichten.

An Objekten vom Typ `oparl:Document` auftretend, bezieht sich die Lizenzinformation nicht nur auf die strukturierten Metadaten, die über die API bezogen werden, sondern auch auf den eigentlichen Inhalt der Dateien, die über die angebotene(n) URL(s) abgerufen werden können.

Lesenswert zum Thema Lizensierung von Linked Data ist auch er Abschnitt “Licenses, Waivers and Norms for Data” im online zugänglichen Linked Data Book.[38]

5.2.5 `created`

Datum und Uhrzeit der Erstellung des jeweiligen Objekts.

5.2.6 `modified`

Diese Eigenschaft kennzeichnet stets Datum und Uhrzeit der letzten Änderung des jeweiligen Objekts.

In Einzelfällen unterliegt die Frage, was als Änderung eines Objekts bezeichnet werden kann, einem gewissen Interpretationsspielraum. Beispielsweise ist zu entscheiden, ob eine Gruppierung (`oparl:Organization`) als geändert gilt, wenn ein neues Mitglied hinzugefügt wurde.

Diese Frage sollte aus Sicht des OParl-Clients beantwortet werden. Wenn beispielsweise eine Gruppierung vom Server grundsätzlich mit der Liste der URLs aller Mitglieder ausgegeben wird, umfasst das Objekt aus Sicht des Clients eben auch die Liste der Mitglieder. In diesem Fall wäre eine Veränderung der Liste der Mitglieder als Änderung des Objekts zu verstehen, die im `modified` Zeitstempel widerspiegeln sollte.

5.2.7 `classification`

Dient der Verschlagwortung und verweist dazu auf ein oder eventuell auch mehrere `skos:Concept`-Objekte mit einer `skos:prefLabel`-Eigenschaft. Der Wert der Eigenschaft ist jeweils eine Zeichenkette. Die verwendete Sprache (in der Regel “de” für Deutsch) MUSS dabei angegeben werden.

TODO: SKOS darstellen

TODO: Beispiel

Siehe u.a.:

`https://github.com/OParl/specs/issues/41`

[37] Verzeichnisse für Lizenz-URLs sind unter anderem unter http://licenses.opendefinition.org/ und https://github.com/fraunhoferfokus/ogd-metadata/blob/master/lizenzen/deutschland.json zu finden.

[38] Tom Heath, Christian Bizer: Linked Data: Evolving the Web into a Global Data Space (1st edition), http://linkeddatabook.com/editions/1.0/#htoc48

Geoportal.de und DeStat.de werden bereits auf Basis von Metadaten-Standards verlinkt. Bei einer Verwendung von Metadaten-Standards für RIS könnten ebenfalls interessante Potenziale durch Verlinkung entstehen. Der GovData-Metadaten-Standard z.B. kann genutzt werden

`http://htmlpreview.github.com/?https://github.com/fraunhoferfokus/ogd-metadata/blob/master/OGPD`

und die 14 GovData-Kategorien können dabei durch eine zusätzliche Verschlagwortung auf Basis von Standard-Schlagwortkatalogen für RIS eine Ergänzung erfahren (LeiKa, Bremer-Katalog, DBpedia, RAMON).

Die 14 Werte des GovData-Metadaten-Standard sind bisher nicht als Linked Data existent. TODO: eventuell können diese in OParl als Schlagwort-`skos:Concept`-Objekte als nicht abschliessende Menge von Schlagworten vorgegeben werden.

5.3 oparl:System (System)

Der Objekttyp `oparl:System` bildet grundlegende Informationen zum parlamentarischen Informationssystem ab. Das Objekt repräsentiert das technische System, unabhängig von der Frage, welche Körperschaften auf diesem System vertreten sind.

Ein Beispiel:

```
{
    "@type": "oparl:System",
    "@id": "https://oparl.beispielris.de/",
    "oparlVersion": "http://oparl.org/specs/1.0/",
    "name": "Beispiel-System",
    "website": "http://www.beispielris.de/",
    "contactEmail": "mailto:info@beispielris.de",
    "contactName": "Allgemeiner OParl Kontakt",
    "vendor": "http://example-software.com/",
    "product": "http://example-software.com/oparl-server/",
    "bodies": "https://oparl.beispielris.de/bodies/",
    "newObjects": "https://oparl.beispielris.de/new_objects/",
    "updatedObjects": "https://oparl.beispielris.de/updated_objects/",
    "removedObjects": "https://oparl.beispielris.de/removed_objects"
}
```

Ein Kontext:

```
{
    "@language": "de",
    "beispielris": "https://oparl.eispielris.de/",
    "oparl": "http://oparl.org/TODO/",
    "dc": "http://purl.org/dc/terms/",
    "foaf": "http://xmlns.com/foaf/0.1/",
    "skos": "http://www.w3.org/2004/02/skos/core#",
    "vcard": "http://www.w3.org/2006/vcard/ns#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",

    "name": {
        "@id": "skos:prefLabel",
        "@type": "@id"
    },
    "contactEmail": {
        "@id": "foaf:mbox",
        "@type": "@id"
    },
}
```

Und das System-Objekt in kompakter Form unter Verwendung des Kontext:

```
{
    "@type": "oparl:System",
    "@id": "https://oparl.beispielris.de/",
    "oparlVersion": "http://oparl.org/specs/1.0/",
    "name": "Beispiel-System",
    "website": "http://www.beispielris.de/",
    "contactEmail": "mailto:info@beispielris.de",
    "contactName": "Allgemeiner OParl Kontakt",
    "vendor": "http://example-software.com/",
    "product": "http://example-software.com/oparl-server/",
    "bodies": "beispielris:bodies/",
    "newObjects": "beispielris:new_objects/",
    "updatedObjects": "beispielris:updated_objects/",
    "removedObjects": "beispielris:removed_objects"
}
```

Auf jedem OParl Server MUSS ein Objekt vom Typ `oparl:System` vorgehalten werden. Es DARF nur ein einziges solches Objekt je Server existieren.

Für Clients ist das `oparl:System` Objekt ein geeigneter Einstiegspunkt, um grundlegende Informationen über das Sytem zu bekommen und die URLs zum Zugriff auf andere Informationen in Erfahrung zu bringen.

Die URL des `oparl:System` Objekts MUSS per Definition identisch mit der URL des API-Endpunkts des Servers sein.

5.3.1 Eigenschaften

`oparlVersion` Die URL der OParl-Spezifikation, die von diesem Server unterstützt wird. Der Wert MUSS die URL `http://oparl.org/specs/1.0/` sein. Diese Eigenschaft ist ZWINGEND.

`bodies` Liste der URLs der `oparl:Body`-Objekte, also der Körperschaften, die auf dem System vorliegen. Alternativ kann statt einer Liste eine einzelne URL zum Abruf der Liste angeboten werden. Die Eigenschaft ist ZWINGEND.

`name` Nutzerfreundlicher Name für das System, mit dessen Hilfe Nutzer das System erkennen und von anderen unterscheiden können. Diese Eigenschaft wird EMPFOHLEN.

`contactEmail` E-Mail-Adresse für Anfragen zur OParl-API. Diese Eigenschaft wird EMPFOHLEN. Die Angabe einer E-Mail-Adresse dient sowohl NutzerInnen wie auch EntwicklerInnen von Clients zur Kontaktaufnahme mit dem Betreiber.

`contactName` Name des Ansprechpartners oder der Abteilung, die über die `contactEmail` erreicht werden kann. Die Eigenschaft ist EMPFOHLEN. Typ: Zeichenkette.

`newObjects` URL des Feeds "Neue Objekte". Die Eigenschaft ist EMPFOHLEN.

`updatedObjects` URL des Feeds "Geänderte Objekte". Die Eigenschaft ist EMPFOHLEN.

`removedObjects` URL des Feeds "Entfernte Objekte". Die Eigenschaft ist EMPFOHLEN.

`website` URL zur WWW-Oberfläche des parlamentarischen Informationssystem. Diese Eigenschaft ist OPTIONAL.

`vendor` URL des Software-Anbieters, von dem die OParl-Server-Software stammt. Diese Eigenschaft ist OPTIONAL.

`product` URL mit Informationen zu der auf dem System genutzten OParl-Server-Software. Diese Eigenschaft ist OPTIONAL.

5.4 oparl:Body (Körperschaft)

Der Objekttyp `oparl:Body` dient dazu, eine Körperschaft und damit ein Parlament zu repräsentieren, zu dem der Server Informationen bereithält. Eine Körperschaft kann beispielsweise eine Gemeinde, ein Landkreis oder ein kommunaler Zweckverband sein.

Hätte das System beispielsweise den Zweck, Informationen über das kommunale Parlament der Stadt Köln, namentlich den Rat der Stadt Köln, abzubilden, dann müsste dieses System dazu ein Objekt vom Typ `oparl:Body` führen, welches die Stadt Köln repräsentiert.

Ein Kontext:

```
{
    "@language": "de",
    "license": {
        "@id": "dc:license",
        "@type": "@id"
    },
    "exactMatch": {
        "@id": "skos:exactMatch",
        "@type": "@id"
    }
    "licenseValidSinceDay": // TODO: datum
}
```

Ein expandiertes Beispiel:

```
{
    "@type": "http://oparl.org/schema/1.0/Body",
    "@id": "https://oparl.beispielris.de/body/0",
    "system": "https://oparl.beispielris.de/",
    "contactEmail": "mailto:ris@beispielstadt.de",
    "contactName": "RIS-Betreuung",
    "rgs": "053150000000",
    "equivalentBody": [
        "http://d-nb.info/gnd/2015732-0",
        "http://dbpedia.org/resource/Cologne"
    ],
    "name": "Stadt Köln",
    "nameLong": {
        "de": "Stadt Köln, kreisfreie Stadt",
        "en": "City of Cologne"
    }
    "website": "http://www.beispielstadt.de/",
    "license": "http://creativecommons.org/licenses/by/4.0/",
    "licenseValidSinceDay": "2014-01-01",
    "organization": "https://oparl.beispielris.de/body/0/organisation/",
    "meeting": "https://oparl.beispielris.de/body/0/meeting/",
    "paper": "https://oparl.beispielris.de/body/0/paper/",
    "member": "https://oparl.beispielris.de/body/0/person/",
    "classification": "https://oparl.beispielris.de/vocab/landkreis",
    "created": "2014-01-08T14:28:31.568+0100",
    "modified": "2014-01-08T14:28:31.568+0100"
}
```

Vom OParl-Server wird erwartet, dass er mindestens ein Objekt vom Typ `oparl:Body` bereit hält. Teilen sich mehrere Körperschaften das selbe technische System, können auf demselben Server auch mehrere Objekte vom Typ `oparl:Body` beherbergt werden.

Über die Zuordnung zu einem bestimmten `oparl:Body` Objekt zeigen andere Objekte, wie beispielsweise Gremien oder Drucksachen, ihre Zugehörigkeit zu einer bestimmten Körperschaft und damit implizit zu einem bestimmten Parlament an.

5.4.1 Eigenschaften

`system` System zu dem dieses Objekt gehört. Typ: `oparl:System`. ZWINGEND.

`name` Gibt den gebräuchlichen Namen der Körperschaft an. Typ: TODO. ZWINGEND.

`nameLong` Kann bei Bedarf dazu verwendet werden, eine längere Form des Namens der Körperschaft anzugeben. Typ: TODO. OPTIONAL.

`website` URL der allgemeinen Website der Körperschaft. EMPFOHLEN.

`license` URL der Lizenz, die für die Daten, die über diese API abgerufen werden können, gilt, sofern nicht am einzelnen Objekt anders angegeben. Sie dazu auch die übergreifende Beschreibung zur Eigenschaft `license`. EMPFOHLEN.

`licenseValidSinceDay` Tagesdatum, seit dem die unter license angegebene Lizenz gilt. Vorsicht bei Änderungen der Lizenz die zu restriktiveren Bedingungen führen. Typ: TODO EMPFOHLEN

`rgs` Regionalschlüssel der Körperschaft als zwölfstellige Zeichenkette[39]. Typ: TODO EMPFOHLEN

`equivalentBody` Dient der Angabe beliebig vieler zusätzlicher URLs, die die selbe Körperschaft repräsentieren. Hier können beispielsweise, sofern vorhanden, der entsprechende Eintrag der Gemeinsamen Normdatei der Deutschen Nationalbibliothek[40], der DBPedia[41] oder der Wikipedia[42] angegeben werden. Typ: TODO EMPFOHLEN

`contactEmail` Dient der Angabe einer Kontakt-E-Mail-Adresse mit "mailto:"-Schema. Die Adresse soll die Kontaktaufnahme zu einer für die Körperschaft und idealerweise das parlamentarische Informationssystem zuständigen Stelle ermöglichen. Typ: TODO EMPFOHLEN

`contactName` Name oder Bezeichnung der mit `contactEmail` erreichbaren Stelle. Typ: TODO OPTIONAL.

`paper` Drucksachen unter dieser Körperschaft. Typ: `oparl:Paper` ZWINGEND.

`member` Personen in dieser Körperschaft. Typ: `oparl:Person` ZWINGEND.

`meeting` Sitzungen dieser Körperschaft. Typ: `oparl:Meeting` ZWINGEND

`organization` Gruppierung in dieser Körperschaft. Typ: `oparl:Organization` ZWINGEND

`classification` Typ: `skos:Concept` OPTIONAL

`created` Datum/Uhrzeit der Erzeugung des Objekts. Typ: TODO EMPFOHLEN.

`lastModified` Datum/Uhrzeit der letzten Bearbeitung des Objekts. Typ: TODO EMPFOHLEN

[39] Regionalschlüssel können im Gemeindeverzeichnis (GV-ISys) des Statistischen Bundesamtes eingesehen werden
[40] Gemeinsame Normdatei http://www.dnb.de/gnd
[41] DBPedia http://www.dbpedia.org/
[42] Wikipedia http://de.wikipedia.org/

5.5 oparl:Organization (Gruppierung)

Dieser Objekttyp dient dazu, Gruppierungen von Personen abzubilden, die in der parlamentarischen Arbeit eine Rolle spielen. Dazu zählen in der Praxis insbesondere Fraktionen und Gremien.

Ein Beispiel in expandierter Form:

```
{
    "@type": "http://oparl.org/schema/1.0/Organization",
    "@id": "https://oparl.beispielris.de/organization/34",
    "body": "https://oparl.beispielris.de/body/0",
    "nameShort": {
        @language" :"de",
        @value": "Finanzausschuss"
    },
    "nameLong": {
        "@language": "de",
        "@value": "Ausschuss für Haushalt und Finanzen"
    },
    "post": {
        "@list": [
            "https://oparl.beispielris.de/post/chairperson",
            "https://oparl.beispielris.de/post/deputyChairperson"
        ]
    },
    "member": [
        "https://oparl.beispielris.de/person/27",
        "https://oparl.beispielris.de/person/48",
        "https://oparl.beispielris.de/person/57"
    ],
    "organizationType": "https://oparl.beispielris.de/vocab/committee",
    "classification": "https://oparl.beispielris.de/vocab/finance",
    "modified": "2012-08-16T14:05:27+02:00"
}
```

Das selbe Beispiel in kompakter Form.

Ein Kontext: ˷~~ { “@language”: “de”, . . . } ˷~~

```
{
    "@context": "https://oparl.beispielris.de/Pfad/zum/Kontext/organization.jsonld",
    "@type": "oparl:Organization",
    "@id": "beispielris:organization/34",
      // kann eventuell weiter verkürzt werden
    "body": "0",
    "nameShort": "Finanzausschuss",
    "nameLong": "Ausschuss für Haushalt und Finanzen",
    "post:" [
        "beispielris:post/chairperson",
        "beispielris:post/deputyChairperson"
    ],
    "members: [
        "27",
        "48",
        "57"
    ],
    "organizationType": "beispielris:vocab/committee",
    "classification": "beispielris:vocab/finance",
```

```
    "modified": "2012-08-16T14:05:27+02:00"
}
```

5.5.1 Eigenschaften

`body` URL der Körperschaft, zu der diese Gruppierung gehört. ZWINGEND

`nameShort` Der Name der Gruppierung, eine Kurzform. ZWINGEND

TODO: ZWINGEND/OPTIONAL prüfen

`nameLong` Langform des Namens der Gruppierung, der rechtlich korrekte Name. ZWINGEND

TODO: ZWINGEND/OPTIONAL prüfen

`post` Position oder Positionen, die für diese Gruppierung vorgesehen sind. Die Objekte gehören zu der Klasse `org:Post` oder einer ihrer Unterklassen. Die `skos:prefLabel`-Eigenschaften der Objekte SOLLEN sowohl die männliche als auch die weibliche Form enthalten, und zwar in dem Muster "männliche Form | weibliche Form" (genau in der Reihenfolge mit einem Leerzeichen vor und nach dem "|") Wenn sich beide Formen nicht unterscheiden, dann DARF die Form nur einmal verwendet werden: "Mitglied" und nicht "Mitglied | Mitglied". Dadurch kann auch solche Software einen sinnvollen Text anzeigen, die keine Fall-Unterscheidung nach Geschecht der Personen vornimmt. z.B. "Vorsitzender | Vorsitzende", "1. Stellvertreter | 1. Stellvertreterin", "2. Stellvertreter | 2. Stellvertreterin", "Schriftführer | Schriftführerin", "Stellvertretender Schriftführer | Stellvertretende Schriftführerin", "Ordentliches Mitglied", "Stellvertretendes Mitglied"

Siehe https://github.com/OParl/specs/issues/45 TODO: "Ordentliches Mitglied", "Stellvertretendes Mitglied" müssen anders behandelt werden! OPTIONAL

`member` URLs aller Mitglieder dieser Organisation (Objekte vom Typ `[oparl:Person](#oparl_person)`). Auch alle Personen mit Positionen in der Organisation sind hier anzugeben. ZWINGEND (falls es Mitglieder gibt)

`subOrganizationOf` Ggf. URL der übergeordneten Organisation. OPTIONAL.

`created` Datum/Uhrzeit der Erzeugung des Objekts. EMPFOHLEN

`classification` Schlagworte. Dies sind `skos:Concept`-Objekte mit einem `skos:prefLabel`-Attribut (für jede unterstützte Sprache) mit einer Zeichenkette. In einer zukünftigen OParl-Version wird möglicherweise eine Menge solcher Schlagwort-Objekte definiert. Anregungen gibt es u.a. in der Tabelle "Kategorien" im unteren Drittel der Seite http://htmlpreview.github.io/?https://github.com/fraunhoferfokus/ogd-metadata/blob/master/OGPD_JSON_Schema.html OPTIONAL

`organizationType` Objekt mit `skos:prefLabel`, z.B. "Rat", "Hauptausschuss", "Ausschuss" "Beirat", "Projektbeirat", "Kommission", "AG", "Verwaltungsrat" OPTIONAL

`modified` Datum/Uhrzeit der letzten Bearbeitung des Objekts. EMPFOHLEN

5.6 oparl:Person (Person)

Jede natürliche Person, die in der parlamentarischen Arbeit tätig ist und insbesondere Mitglied in einer Gruppierung (oparl:Organization), wird mit einem Objekt vom Typ `oparl:Person` abgebildet.

Es gibt existieren bereits eine ganze Reihe von Vokabularen für Personen. Dazu gehören FOAF (Friend of a Friend) und vCard. Es gibt aber auch der XÖV-Standard für natürliche

Personen, ein XML Schema. Für `oparl:Person` wurde daraus und basierend auf dem Input der OParl-Stakeholder eine Auswahl von Eigenschaften zusammengestellt.

TODO: für Personen-Namen und Titel wird keine Mehrsprachigkeit benötigt. Dies im Kontext berücksichtigen. Dies spricht auch für je einen Kontext pro Klasse.

Ein Beispiel in expandierter Form:

```
{
    "@type": "http://oparl.org/schema/1.0/Person",
    "@id": "https://oparl.beispielris.de/person/29",
    "name": "Prof. Dr. Max Mustermann",
    "familyName": { // könnte mehrsprachig sein, z.B. griechisch, russisch, tamilisch
        "@value": "Mustermann",
        "@language": "de"
    },
    "givenName": { // könnte mehrsprachig sein
        "@value": "Max",
        "@language": "de"
    },
    "title": "Prof. Dr.", // TODO: nicht mehrsprachig?!
    "formOfAddress": "https://oparl.beispielris.de/formofaddress/ratsmitglied",
    "gender": "http://www.w3.org/2006/vcard/ns#Male",
    "email": "mailto:max@mustermann.de",
    "phone": "tel:+493012345678",
    "streetAddress": "Musterstraße 5", // nicht mehrsprachig
    "postalCode": "11111",
    "locality": {
        "de": "Musterort",
        "en": "Sample Town"
    },
    "organization": [
        "https://oparl.beispielris.de/organization/11",
        "https://oparl.beispielris.de/organization/34"
    ],
    "status": "https://oparl.beispielris.de/status/buergermeister",
    "hasMembership": "https://oparl.beispielris.de/membership/34",
    "created": "2011-11-11T11:11:00+01:00",
    "modified": "2012-08-16T14:05:27+02:00"
}
```

Das selbe Beispiel in kompakter Form. Zunächst der verwendete Kontext:

```
{
    "@language": "de",

    // Präfixe siehe Abschnitt 8000

    "gender": "vcard:hasGender",
    "givenName": "foaf:firstName",
    "familyName": "foaf:lastName",
    "academic_degree": {
        "@language": null , // keine Vorgabesprache da nicht mehrsprachig
        "@id": "foaf:title"
    },
    "email": {
        "@id": "foaf:mbox",
        "@type": "@id"
    },
```

```
    "phone": "foaf:phone",
    "streetAddress": "vcard:street-address",
    "locality": {
        "@id": "vcard:locality",
        "@container": "@language" // für eine "language map"
    },
    "created": {
        "@id": "dc:created",
        "@type": "xsd:dateTime"
    },
    "modified": {
        "@id": "dc:modified",
        "@type": "xsd:dateTime"
    }
}
```

```
{
    "@context": "https://oparl.beispielris.de/Pfad/zum/Kontext/person.jsonld",
    "@type": "oparl:Person",
    "@id": "https://oparl.beispielris.de/person/29",
    "name": "Prof. Dr. Max Mustermann",
    "familyName": "Mustermann", // Kontext gibt deutsche Spache vor
    "givenName": "Max",
    "title": "Prof. Dr.",
    "formOfAddress": "beispielris:formofaddress/ratsmitglied",
    "gender": "vcard:Male",
    "email": "mailto:max@mustermann.de",
    "phone": "tel:+493012345678",
    "streetAddress": "Musterstraße 5",
    "postalCode": "11111",
    "locality": "Musterort",
    "locality": {
        "en": "Sample Town" // TODO prüfen, ob Eigenschaft doppelt erscheinen darf
    },
    "organization": ["11", "34"],
    "status": "beispielris:status/buergermeister",
    "hasMembership": "beispielris:membership/34",
    "created": "2011-11-11T11:11:00+01:00",
    "modified": "2012-08-16T14:05:27+02:00"
}
```

5.6.1 Eigenschaften

`name` Der vollständige Name der Person mit akademischem Grad und Vornamen. Typ: TODO ZWINGEND

`familyName` Familienname bzw. Nachname. Typ: TODO OPTIONAL

`givenName` Vorname bzw. Taufname. Typ: TODO OPTIONAL

`formOfAddress` Anrede Begriff mit `skos:prefLabel`. Ähnlich wie `status`. Beispiele für die `skos:prefLabel` sind “Ratsherr | Ratsfrau” und “Herr | Frau”. Typ: `skos:Concept` OPTIONAL

`title` Akademische(r) Titel. TODO: “Dr.”? “Diplom”? Typ: `skos:Concept` OPTIONAL

`gender` Geschlecht. Zulässige Werte sind `vcard:Female`, `vcard:Male`, `vcard:None`, `vcard:Other` und `vcard:Unknown`. Typ: `vcard:TODO` OPTIONAL

phone Telefonnummer mit `tel:` Schema. Typ: TODO OPTIONAL

email E-Mail-Adresse mit `mailto:` Schema. Typ: `foaf:mbox` OPTIONAL

streetAddress Straße und Hausnummer der Kontakt-Anschrift der Person. Typ: TODO OPTIONAL

postalCode Postleitzahl der Kontakt-Anschrift der Person. Typ: TODO OPTIONAL

locality Ortsangabe der Kontakt-Anschrift der Person. Typ: `vcard:locality` OPTIONAL

organization Gruppierung in der die Person aktuell Mitglied ist. Typ: `oparl:Organization` ZWINGEND

status Status. Begriff mit `skos:prefLabel`. Die Zeichenketten SOLLEN sowohl die männliche als auch die weibliche Form enthalten, und zwar in dem Muster "männliche Form | weibliche Form" (genau in der Reihenfolge mit einem Leerzeichen vor und nach dem "|") Wenn sich beide Formen nicht unterscheiden, dann DARF die Form nur einmal verwendet werden: "Mitglied" und nicht "Mitglied | Mitglied". Dadurch kann auch solche Software einen sinnvollen Text anzeigen, die keine Fall-Unterscheidung nach Geschecht der Personen vornimmt. z.B. "Bürgermeister | Bürgermeisterin", "Bezirksbürgermeister | Bezirksbürgermeisterin", "Stadtverordneter | Stadtverordnete", "Bezirksverordneter | Bezirksverordnete", "Sachkundiger Bürger | Sachkundige Bürgerin", "Einzelstadtverordneter | Einzelstadtverordnete" (Mitglieder des Rates die keiner Fraktion/Organisation angehören -> die Zuordbarkeit einer fiktiven Organisation ermöglichen TODO: warum will man das?). Siehe https://github.com/OParl/specs/issues/45 Typ: `skos:Concept` OPTIONAL

hasMembership Mitgliedschaft. TODO: Eventuell Unterklasse von `org:Membership` definieren. Typ: `org:Membership` OPTIONAL.

classification Typ: `skos:Concept` OPTIONAL

created Datum/Uhrzeit der Erzeugung des Objekts. Typ: String mit xsd:dateTime EMPFOHLEN.

lastModified Datum/Uhrzeit der letzten Bearbeitung des Objekts. Typ: String mit xsd:dateTime EMPFOHLEN.

5.7 oparl:Meeting (Sitzung)

Eine Sitzung ist die Versammlung einer oder mehrerer Gruppierungen (oparl:Organization) zu einem bestimmten Zeitpunkt an einem bestimmten Ort.

Die geladenen Teilnehmer der Sitzung sind jeweils als Objekte vom Typ oparl:Person in entsprechender Form referenziert. Verschiedene Dokumente (Einladung, Ergebnis- und Wortprotokoll, sonstige Anlagen) können referenziert werden.

Die Inhalte einer Sitzung werden durch Tagesordnungspunkte (oparl:AgendaItem) abgebildet.

Ein Beispiel in expandierter Form:

```
{
    "@type": "http://oparl.org/schema/1.0/Meeting",
    "@id": "https://oparl.beispielris.de/meeting/281",
    "name": "4. Sitzung des Finanzausschusses",
    "start": "2013-01-04T08:00:00+01:00",
    "end": "2013-01-04T12:00:00+01:00",
    "location": {
        "description": {
            "@value": "Rathaus, Raum 136",
            "@language": "de"
```

```
        },
        "description": {
            "@value": "Town Hall, room 136",
            "@language": "en"
        }
    },
    "organization": "https://oparl.beispielris.de/organization/34",
    "participant": [
        "https://oparl.beispielris.de/person/29",
        "https://oparl.beispielris.de/person/75",
        "https://oparl.beispielris.de/person/94"
    ],
    "invitation": "https://oparl.beispielris.de/document/586",
    "resultsProtocol": "https://oparl.beispielris.de/document/628",
    "verbatimProtocol": "https://oparl.beispielris.de/document/691",
    "auxiliaryDocument": [
        "https://oparl.beispielris.de/document/588",
        "https://oparl.beispielris.de/document/589"
    ],
    "agendaItem": {
        "@list": [
            "https://oparl.beispielris.de/agendaitem/1045",
            "https://oparl.beispielris.de/agendaitem/1046",
            "https://oparl.beispielris.de/agendaitem/1047",
            "https://oparl.beispielris.de/agendaitem/1048"
        ]
    },
    "created": "2012-01-06T12:01:00+01:00",
    "modified": "2012-01-08T14:05:27+01:00"
}
```

Das selbe Beispiel in kompakter Form:

```
{
    "@context": "https://oparl.beispielris.de/Pfad/zum/Kontext/oparl.jsonld"
    "@type": "oparl:Meeting",
    "@id": "https://oparl.beispielris.de/meeting/281",
    "name": "4. Sitzung des Finanzausschusses",
    "start": "2013-01-04T08:00:00+01:00",
    "end": "2013-01-04T12:00:00+01:00",
    "location": {
        "description": "Rathaus, Raum 136", // default-Sprache ist im Kontext als "de" angegebe
        "description": {
            "@value": "Town Hall, room 136",
            "@language": "en"
        }
    },
    "organization": "beispielris:organization/34",
    "participant": [
        "beispielris:person/29",
        "beispielris:person/75",
        "beispielris:person/94"
    ],
    "invitation": "beispielris:document/586",
    "resultsProtocol": "beispielris:document/628",
    "verbatimProtocol": "beispielris:document/691",
    "auxiliaryDocument": [
        "beispielris:document/588",
```

```
        "beispielris:document/589"
    ],
    "agendaItem": [
    // Reihenfolge ist wichtig, deshalb @list im Kontext angeben
        "beispielris:agendaitem/1045",
        "beispielris:agendaitem/1046",
        "beispielris:agendaitem/1047",
        "beispielris:agendaitem/1048"
    ],
    "created": "2012-01-06T12:01:00+01:00",
    "modified": "2012-01-08T14:05:27+01:00"
}
```

5.7.1 Eigenschaften

`start` Datum und Uhrzeit des Anfangszeitpunkts der Sitzung. Bei einer zukünftigen Sitzung ist dies der geplante Zeitpunkt, bei einer stattgefundenen KANN es der tatsächliche Startzeitpunkt sein. Typ: TODO. ZWINGEND

`end` Endzeitpunkt der Sitzung als Datum/Uhrzeit. Bei einer zukünftigen Sitzung ist dies der geplante Zeitpunkt, bei einer stattgefundenen KANN es der tatsächliche Endzeitpunkt sein. Typ: TODO. EMPFOHLEN

`location` Sitzungsort. Typ: `oparl:Location`. EMPFOHLEN

`organization` Gruppierung der die Sitzung zugeordnet ist. Wenn eine Liste angegeben wird, dann ist diese geordnet. Das erste Element ist dann das federführende Gremium. Typ: `oparl:Organization`. ZWINGEND

`participant` Teilnehmer der Sitzung. Bei einer Sitzung in der Zukunft sind dies die geladenen Teilnehmer, bei einer stattgefundenen Sitzung SOLL die Liste nur diejenigen Teilnehmer umfassen, die tatsächlich an der Sitzung teilgenommen haben. Typ: `oparl:Person`. ZWINGEND.

`invitation` Einladungsdokument zur Sitzung. Typ: `oparl:Document`. EMPFOHLEN.

`resultsProtocol` Ergebnisprotokoll zur Sitzung. Diese Eigenschaft kann selbstverständlich erst nach dem Stattfinden der Sitzung vorkommen. Typ: `oparl:Document`. EMPFOHLEN

`verbatimProtocol` Wortprotokoll zur Sitzung. Diese Eigenschaft kann selbstverständlich erst nach dem Stattfinden der Sitzung vorkommen. Typ: `oparl:Document`. EMPFOHLEN

`auxiliaryDocument` Dokumentenanhang zur Sitzung. Hiermit sind Dokumente gemeint, die üblicherweise mit der Einladung zu einer Sitzung verteilt werden und die nicht bereits über einzelne Tagesordnungspunkte referenziert sind. Typ: `oparl:Document`. OPTIONAL

`agendaItem` Tagesordnungspunkte der Sitzung. Die Reihenfolge ist relevant. Es kann Sitzungen ohne TOPs geben. Typ: `oparl:AgendaItem`. OPTIONAL

`classification` Typ: `skos:Concept`. OPTIONAL

`created` Datum und Uhrzeit der Erzeugung des Objekts. Typ: TODO. EMPFOHLEN

`modified` Datum und Uhrzeit der letzten Änderung des Objekts. Typ: TODO. EMPFOHLEN

5.8 oparl:AgendaItem (Tagesordnungspunkt)

Tagesordnungspunkte sind die Bestandteile von Sitzungen (`oparl:Meeting`). Jeder Tagesordnungspunkt widmet sich inhaltlich einem bestimmten Thema, wozu in der Regel auch die Beratung bestimmter Drucksachen gehört.

Ein Beispiel in kompakter Form:

```
{
    "@context": "https://oparl.beispielris.de/Pfad/zum/Kontext/oparl.jsonld"
    "@type": "oparl:AgendaItem",
    "@id": "https://oparl.beispielris.de/agendaitem/3271",
    "meeting": "beispielris:meeting/281",
    "number": "10.1",
    "name": "Satzungsänderung für Ausschreibungen",
    "public": true,
    "consultations": [
        "beispielris:consultation/1034",
        "beispielris:consultation/1235"
    ],
    "result": "besipielris:vocab/decided_modified",
    "resolutionText": "Der Beschluss weicht wie folgt vom Antrag ab: ...",
    "absentParticipant": [
        "beispielris:person/75"
    ],
    "paper": "beispielris:paper/2812",
    "modified": "2012-08-16T14:05:27+02:00"
}
```

5.8.1 Eigenschaften

`meeting` Sitzung, der der Tagesordnungspunkt zugeordnet ist. Typ: `oparl:Meeting` ZWINGEND.

`number` Nummer des Tagesordnungspunktes. Eine beliebige Zeichenkette, wie z.B. “10.”, “10.1”, “C”, “c)” o.ä. Die Reihenfolge wird dadurch nicht festgelegt, sondern durch die Reihenfolge der TOPs im `agendaItem`-Attribut von `oparl:Meeting` Typ: TODO OPTIONAL

`name` Das Thema des Tagesordnungspunktes. Typ: TODO ZWINGEND.

`public` Kennzeichnet, ob der Tagesordnungspunkt zur Behandlung in öffentlicher Sitzung vorgesehen ist/war. Es wird ein Wahrheitswert (`true` oder `false`) erwartet. Typ: boolean EMPFOHLEN.

`consultation` Beratung, die diesem Tagesordnungspunkt zugewiesen ist. Typ: `oparl:Consultation` ZWINGEND.

`result` Kategorische Information darüber, welches Ergebnis die Beratung des Tagesordnungspunktes erbracht hat. Es wird zu einem Objekt verlinkt, welches ein `skos:prefLabel`-Attribut mit einer Zeichenkette hat. In der Praxis sind hier Kategorien wie “Unverändert beschlossen”, “Geändert beschlossen”, “Endgültig abgelehnt”, “Zur Kenntnis genommen”, “Ohne Votum in nachfolgende Gremien überwiesen” und weitere zu erwarten. Alternativ können, sobald dieses zur Verfügung steht, URLs aus einem OParl Vokabular verwendet werden, wie im Beispiel oben zu sehen. Diese dienen dazu, Kategorien über Systemgrenzen hinweg maschinenlesbar zu vereinheitlichen. Typ: `skos:Concept` EMPFOHLEN

`resolutionText` Falls in diesem Tagesordnungspunkt ein Beschluss gefasst wurde, kann der Text hier hinterlegt werden. Das ist besonders dann in der Praxis relevant, wenn der

gefasste Beschluss (z.B. durch Änderungsantrag) von der Beschlussvorlage abweicht. Typ: TODO OPTIONAL.

paper Drucksache. Zwar kann auch das `oparl:Meeting` darauf verweisen, aber hier sind solche Verweise in der Regel präziser, da Drucksachen regelmäßig nur für einen TOP relevant sind und nicht für alle TOPs. Typ: `oparl:Paper` OPTIONAL

auxiliaryDocument eine weitere relevante Datei. Typ: `oparl:Document` OPTIONAL

classification Typ: `skos:Concept` OPTIONAL

created Erzeugungsdatum und -zeit des Objekts. Typ: TODO EMPFOHLEN

lastModified Datum und Uhrzeit der letzten Änderung. Typ: TODO EMPFOHLEN

5.9 oparl:Paper (Drucksache)

Dieser Objekttyp dient der Abbildung von Drucksachen in der parlamentarischen Arbeit, wie zum Beispiel Anfragen, Anträgen und Beschlussvorlagen.

Drucksachen werden in Form einer Beratung (oparl:Consultation) im Rahmen eines Tagesordnungspunkts (oparl:AgendaItem) einer Sitzung (oparl:Meeting) behandelt.

Drucksachen spielen in der schriftlichen wie mündlichen Kommunikation eine besondere Rolle, da in vielen Texten auf bestimmte Drucksachen Bezug genommen wird. Hierbei kommen in parlamentarischen Informationssystemen unveränderliche Kennungen der Drucksachen zum Einsatz.

Zunächst ein Kontext:

```
// consultations als @list deklarieren!
```

Ein Beispiel in kompakter Form:

```
{
    "@context": "https://oparl.beispielris.de/Pfad/zum/Kontext/oparl.jsonld"
    "@type": "oparl:Paper",
    "@id": "https://oparl.beispielris.de/paper/749",
    "reference": "1234/2014",
    "publishedDate": "2014-04-04T16:42:02+02:00",
    "name": "Antwort auf Anfrage 1200/2014",
    "paperType": "beispielris:vocab/answer",
    "relatedPaper": "beispielris:paper/699",
    "mainDocument": "beispielris:document/925",
    "auxiliaryDocument": "beispielris:document/926",
    "location": [
        {
            "description": "Theodor-Heuss-Ring 1",
            "geometry": "POINT(7.148  50.023)"
        }
    ],
    "creator": [
        "beispielris:organization/2000",
        "beispielris:people/1000"
    ],
    "consultation": [
        "beispielris:consultation/5676",
        "beispielris:consultation/5689"
    ]
    "modified": "2013-01-08T12:05:27+01:00"
}
```

5.9.1 Eigenschaften

reference Beispiel: “1234/2014” Typ: TODO OPTIONAL

publishedDate Beispiel: “2014-04-04T16:42:02+02:00” Typ: TODO EMPFOHLEN

name Beispiel: “Antwort auf Anfrage 1200/2014” Typ: TODO EMPFOHLEN

paperType Begriff mit einem `skos:prefLabel`-Attribut, dessen Wert eine Zeichenkette ist und die Art der Drucksache beschreibt, z.B. “Beantwortung einer Anfrage”. Für die URLs kommen als letztes Pfadelement z.B. “draft”, “petition”, “request”, “note” und “answer” in Frage. Denkbar ist auch eine Kategorisierung z.B. in drei Arten von Drucksachen: initiierend, beratend und protokollierend. Eine weitere Liste mit exemplarischen Drucksachentypen gibt es hier: https://wiki.piratenpartei.de/BE:BVVupdates/Glossar Eine zukünftige Version von OParl wird möglicherweise solche Werte spezifizieren. Typ: `skos:Concept` OPTIONAL

relatedPaper Typ: `oparl:Paper` OPTIONAL

mainDocument Typ: `oparl:Document`

auxiliaryDocument Typ: `oparl:Document`

location Typ: `oparl:Location`

creator Typ: `oparl:Person` | `oparl:Organization`

consultation Typ: `oparl:Consultation`

modified Typ: TODO Beispiel: “2013-01-08T12:05:27+01:00”

classification Begriff mit `skos:prefLabel`. Allgemeiner als `paperType` Typ: `skos:Concept` OPTIONAL

TODO: * Festlegen: OPTIONAL / EMPFOHLEN / ZWINGEND * Eigenschaft “locations” im Beispiel ändern

5.10 oparl:Document (Datei)

Ein Objekt vom Typ `oparl:Document` repräsentiert eine Datei, beispielsweise eine PDF-Datei, ein RTF- oder ODF-Dokument, und hält Metadaten zu der Datei sowie URLs zum Zugriff auf die Datei bereit.

Ein Beispiel:


```
{
    "@type": "oparl:Document",
    "@id": "https://oparl.beispielris.de/document/57739",
    "name": "Anlage 1 zur Anfrage",
    "fileName": "57739.pdf",
    "paper": "https://oparl.beispielris.de/paper/2396",
    "mimeType": "application/pdf",
    "date": "2013-01-04T07:54:13+01:00",
    "lastModified": "2013-01-04T07:54:13+01:00",
    "sha1Checksum": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
    "size": 82930,
    "accessUrl": "https://oparl.beispielris.de/document/57739.pdf",
    "downloadUrl": "https://oparl.beispielris.de/document/download/57739.pdf",
    "text": "Der Übersichtsplan zeigt alle Ebenen des ...",
    "masterDocument": "https://oparl.beispielris.de/document/57738",
    "license": "http://www.opendefinition.org/licenses/cc-by",
    "documentRole": "https://oparl.beispielris.de/document-role/evidence",
}
```

Objekt vom Typ `oparl:Document` können mit Drucksachen (`oparl:Paper`) oder Sitzungen (`oparl:Meeting`) in Beziehung stehen. Dies wird durch die Eigenschaft `paper` bzw. `meeting` angezeigt.

Mehrere Objekte vom Typ `oparl:Document` können mit einander in direkter Beziehung stehen, wenn sie den selben Inhalt in unterschiedlichen technischen Formaten wiedergeben. Hierfür werden die Eigenschaften `masterDocument` bzw. `derivativeDocuments` eingesetzt. Das oben angezeigte Beispiel-Objekt repräsentiert eine PDF-Datei (zu erkennen an der Eigenschaft `mimeType`) und zeigt außerdem über die Eigenschaft `masterDocument` an, von welcher anderen Datei es abgeleitet wurde. Umgekehrt KANN über die Eigenschaft `derivativeDocuments` angezeigt werden, welche Ableitungen einer Datei existieren.

5.10.1 Eigenschaften

`fileName` Dateiname, unter dem die Datei in einem Dateisystem gespeichert werden kann. Typ: ASCII-Zeichenkette. ZWINGEND.

`name` Ein zur Anzeige für Endnutzer bestimmter Name für dieses Objekt. Der Wert SOLL NICHT mit dem Wert der Eigenschaft `fileName` identisch sein. Typ: Zeichenkette. EMPFOHLEN.

`mimeType` Mime-Typ des Inhalts[43]. Sollte das System einer Datei keinen spezifischen Typ zuweisen können, wird EMPFOHLEN, hier `application/octet-stream` zu verwenden. Typ: Zeichenkette ZWINGEND.

`date` Datum und Zeit der Erstellung der Datei. Wahlweise, falls dies nicht vom System kommuniziert werden kann oder soll, KANN alternativ der Zeitpunkt der Veröffentlichung ausgegeben werden. Typ: Datum. ZWINGEND.

`lastModified` Datum und Zeit der letzten Änderung der Datei bzw. der Metadaten. Als Änderung der Datei gilt alles, was den Inhalt der Datei verändert und beispielsweise zu einer Veränderung der Prüfsumme führen würde, nicht aber die Änderung des Dateinamens (siehe Eigenschaft `name`). Als Änderung der Metadaten hingegen würde beispielsweise die Änderung des Dateinamens gelten. Hier soll immer das größere der beiden Daten ausgegeben werden, also der am wenigsten lang zurückliegende Änderungszeitpunkt. Typ: Datum. ZWINGEND.

`size` Größe der Datei in Bytes. Typ: ganze Zahl. ZWINGEND.

`sha1Checksum` SHA1-Prüfsumme des Dokumenteninhalts in Hexadezimal-Schreibweise. Typ: Zeichenkette. OPTIONAL.

`text` Reine Text-Wiedergabe des Dateiinhalts, sofern dieser in Textform wiedergegeben werden kann. Typ: Zeichenkette. EMPFOHLEN.

`accessUrl` URL zum allgemeinen Zugriff auf die Datei. Näheres unter Dateizugriff. Typ: URL. ZWINGEND.

`downloadUrl` URL zum Download der Datei. Näheres unter Dateizugriff. Typ: URL. EMPFOHLEN.

`paper` Falls die Datei zu einer Drucksache (oparl:Paper) gehört, MUSS über diese Eigenschaft die URL des Drucksache-Objekts ausgegeben werden. Andernsfalls DARF diese Eigenschaft NICHT vorhanden sein. Typ: `oparl:Paper`.

`meeting` Falls die Datei zu einer Sitzung (oparl:Meeting) gehört, MUSS über diese Eigenschaft die URL des Sitzung-Objekts ausgegeben werden. Andernfalls DARF diese Eigenschaft NICHT vorhanden sein. Typ: `oparl:Meeting`.

`masterDocument` Datei von der das aktuelle Objekt abgeleitet wurde. Typ: `oparl:Document`. OPTIONAL.

[43] vgl. RFC2046: http://tools.ietf.org/html/rfc2046

derivativeDocuments Abgeleitete Datei die von dem aktuellen Objekt abgeleitet wurde. Typ: `oparl:Document`. OPTIONAL.

license Lizenz unter der die Datei angeboten wird. Wenn diese Eigenschaft verwendet wird, dann ist sie anstelle einer globalen Angabe im übergeordneten `oparl:Body` bzw. `oparl:System` Objekt maßgeblich.[44] Typ: URL. OPTIONAL.

documentRole Rolle, Funktion, Sorte des Dokuments. Das Objekt enthält ein `skos:prefLabel`. Dessen Werte können z.B. sein: "Einladung", "Protokoll", "Wortprotokoll" oder "Beschlussprotokoll". In einer zukünftigen OParl-Version wird möglicherweise eine Menge der wichtigsten Kategorien vorgegeben. Typ: `skos:Concept`. OPTIONAL. TODO: Link auf Erkärungs-Kapitel, damit klar ist, dass hier auf externes Vokabular verlinkt wird.

classification Begriff mit `skos:prefLabel`. Hat allgemeinere Bedeutung als `documentRole`. Typ: `skos:Concept`. OPTIONAL. TODO: TODO: Link auf Erkärungs-Kapitel, damit klar ist, dass hier auf externes Vokabular verlinkt wird.

5.11 oparl:Consultation (Beratung)

Der Objekttyp `oparl:Consultation` dient dazu, die Beratung einer Drucksache (`oparl:Paper`) in einer Sitzung abzubilden. Dabei ist es nicht entscheidend, ob diese Beratung in der Vergangenheit stattgefunden hat oder diese für die Zukunft geplant ist.

Die Gesamtheit aller Objekte des Typs `oparl:Consultation` zu einer bestimmten Drucksache bildet das ab, was in der Praxis als "Beratungsfolge" der Drucksache bezeichnet wird.

Ein Beispiel:

```
{
    "@type": "oparl:Consultation",
    "@id": "https://oparl.beispielris.de/consultation/47594",
    "paper": "https://oparl.beispielris.de/paper/2396",
    "agendaitem": "https://oparl.beispielris.de/agendaitem/15569",
    "committee": "https://oparl.beispielris.de/organization/96",
    "authoritative": false,
    "role": "https://oparl.beispielris.de/role/decision"
}
```

Das selbe Beispiel in kompakter Form (ein passender Kontext wird vorausgesetzt):

```
{
    "@context": "https://oparl.beispielris.de/Pfad/zum/Kontext/oparl.jsonld",
    "@type": "oparl:Consultation",
    "@id": "beispielris:consultation/47594",
    "paper": "beispielris:paper/2396",
    "agendaitem": "beispielris:agendaitem/15569",
    "committee": "beispielris:organization/96",
    "authoritative": false,
    "role": "beispielris:role/decision"
}
```

Das Objekt "beispielris:roles/decision" kann so aussehen:

```
{
    "@context": "https://oparl.beispielris.de/Pfad/zum/Kontext/oparl.jsonld"
    "@id": "beispielris:role/decision",
```

[44] vgl. license

```
    "prefLabel": {
        "de": "Entscheidung",
        "en": "decision"
    }
}
```

5.11.1 Eigenschaften

paper Drucksache, die beraten wird. Typ: `oparl:Paper` ZWINGEND.

agendaitem Tagesordnungspunkt, unter dem die Drucksache beraten wird. Typ: `oparl:AgendaItem` ZWINGEND.

committee Gremium, dem die Sitzung zugewiesen ist, zu welcher der zuvor genannte Tagesordnungspunkt gehört. Hier kann auch eine mit Liste von Gremien angegeben werden (die verschiedenen `oparl:Body` und `oparl:System` angehören können). Die Liste ist dann geordnet. Das erste Gremium der Liste ist federführend. Typ: `oparl:Organization` ZWINGEND.

authoritative Drückt aus, ob bei dieser Beratung ein Beschluss zu der Drucksache gefasst wird (`true`) wird oder nicht (`false`). Typ: boolean. Diese Eigenschaft ist OPTIONAL.

role Rolle oder Funktion der Beratung. z.B. Anhörung (hearing), Entscheidung (decision), Kenntnisnahme (notice), Vorberatung (counseling) usw. Es wird empfohlen in den URLs entsprechende englische Bestandteile zu verwenden. Die Rollenobjekte haben nur eine festgelegte Eigenschaft: `skos:prefLabel` für den Namen. In einer zukünftigen Version von OParl können gegebenenfalls die am stärksten benötigten Rollen standardisiert werden. Typ: `skos:Concept` OPTIONAL

classification Schlagwort, Begriff mit `skos:prefLabel`. Allgemeiner verwendbar als `role`. Typ: `skos:Concept` OPTIONAL

5.12 oparl:Location (Ort)

Dieser Objekttyp dient dazu, den Ortsbezug einer Drucksache formal abzubilden. Ortsangaben können sowohl aus Textinformationen bestehen (beispielsweise dem Namen einer Straße/eines Platzes oder eine genaue Adresse) als auch aus Geodaten.

In der Praxis soll dies dazu dienen, den geografischen Bezug eines politischen Vorgangs, wie zum Beispiel eines Bauvorhabens oder der Änderung eines Flächennutzungsplanes, maschinenlesbar nachvollziehbar zu machen.

Dieser Objekttyp kann für Objekte im Kontext des Objekttyps `oparl:Paper` verwendet werden.

Ein einfaches Beispiel:

```
{
    ...
    "location": {
        "description": "Honschaftsstraße 312, Köln",
        "geometry": "POINT (7.03291 50.98249)"
    },
    ...
}
```

Ein Kontext (TODO: Kann das weg?):

```
{
   "geometry":
   {
     "@type": "ogc:wktLiteral"
   }
}
```

Und ein Beispiel unter Verwendung des Kontextes:

```
{
    // ...
    "location": {
        "description": "Honschaftsstraße 312, Köln",
        "geometry": "POINT (7.03291 50.98249)"
    },
    // ...
}
```

OParl sieht bei Angabe von Geodaten ZWINGEND die Verwendung des Well-Known-Text-Formats (WKT) der Simple Feature Access Spezifikation[45] vor. WKT erlaubt die Beschreibung von unterschiedlichen Geometrien wie Punkten (`Point`), Pfaden (`LineString`), Polygonen (`Polygon`) und viele andere mehr.

Zum Zeitpunkt der Erstellung der vorliegenden Spezifikation ist Version 1.2.1 der Simple-Feature-Access-Spezifikation aktuell. OParl stellt keine Anforderungen daran, welche Version von Simple-Feature-Access bei der Ausgabe von WKT zu unterstützen ist.

Für die Ausgabe über eine OParl API MÜSSEN sämtliche Koordinatenangaben solcher Geodaten im System WGS84[46] angegeben werden, und zwar in Form von Zahlenwerten (Fließkommazahlen) für Längen- und Breitengrad.

5.12.1 Eigenschaften

`description` Textliche Beschreibung eines Orts, z.B. in Form einer Adresse. Typ: Zeichenkette. EMPFOHLEN

`geometry` Geodaten-Repräsentation des Orts. Ist diese Eigenschaft gesetzt, MUSS ihr Wert der Spezifikation von Well-Known Text (WKT) entsprechen. Typ: Zeichenkette (TODO: Stimmt das?). OPTIONAL

`classification` Schlagwort mit `skos:prefLabel` Typ: `skos:Concept` OPTIONAL

5.12.2 Weitere Beispiele

Ortsangabe mit Polygon-Objekt:

```
{
    "description": "Rechtes Rheinufer zwischen Deutzer
        Brücke und Hohenzollernbrücke",
    "geometry": "POLYGON ((
                6.9681106 50.9412137,
                6.9690940 50.9412137,
                6.9692169 50.9368270,
                6.9681218 50.9368270,
                6.9681106 50.9412137))"
}
```

[45] Simple Feature Access Spezifikation: http://www.opengeospatial.org/standards/sfa

[46] WGS84 steht für "World Geodetic System 1984", es wird unter anderem auch vom Global Positioning System (GPS) verwendet. In geografischen Informationssystemen ist für das System der EPSG-Code 4326 geläufig.

5.13 org:Membership oder oparl:Membership

TODO. Siehe: https://github.com/OParl/specs/issues/122 https://github.com/OParl/specs/issues/109

5.13.1 Eigenschaften

`person` Person. Typ: `oparl:Person` OPTIONAL

`organization` Organization. Typ: `oparl:Organization` OPTIONAL

`role` Rolle. Das Objekt hat eine `skos:prefLabel`-Eigenschaft, deren Wert eine Funktionsbezeichnung ist, z.B. "1. pers. Vertreter | 1. pers. Vertreterin" oder "2. pers. Vertreter | 2. pers. Vertreterin". Popolo: "The role that the person fulfills in the organization". normale Mitglieder haben in der Regel keine eigene Funktion, aber auch eine Unterscheidung zwischen z.B. "Sachkundige Bürger | Sachkundige Bürgerin" und "Ratsherr | Ratsfrau" bei einfachen Mitgliedern ist hiermit möglich. TODO: was ist mit einem sachkundigen Bürger, der gleichzeitig Vorsitzender ist? Typ: `org:Role` OPTIONAL

`post` The post held by the person in the organization Typ: `org:Post` OPTIONAL

`onBehalfOf` Entsendende Organization - Fraktion, fraktionslos oder externes Gremium Dies entspricht `opengov:onBehalfOf` in Popolo. TODO: wie wird fraktionslos kodiert? Typ: `oparl:Organization` OPTIONAL

`startDate` `schema:validFrom` wie in Popolo. The date on which the relationship began Typ: TODO OPTIONAL

`endDate` `schema:validThrough` wie in Popolo. The date on which the relationship ended Typ: TODO OPTIONAL

5.13.2 Hinweise

http://www.w3.org/TR/vocab-org/#membership-roles-posts-and-reporting

http://popoloproject.com/specs/membership.html

6 Glossar

IRI Internationalized Resource Identifier ist die internationalisierte Form der Uniform Resource Identifier (URI). Diese sind in RFC 3987 spezifiziert (http://tools.ietf.org/html/rfc3987). In der OParl-Spezifikation sind grundsätzlich auch dann IRIs gemeint, wenn die Bezeichnungen URI oder URL verwendet werden. Dies dient der Lesbarkeit, auch wenn es technisch nicht korrekt ist.

JSON JavaScript Object Notation ist ein strukturiertes Datenformat, welches als Teil von JavaScript bzw. ECMAScript spezifiziert ist.

JSON-LD JSON for Linked Data

RIS Ratsinformationssystem

URI Uniform Resource Identifier. In der OParl-Spezifikation sind grundsätzlich auch dann IRIs gemeint, wenn die Bezeichnung URI verwendet wird.

URL Uniform Resource Locator. In der OParl-Spezifikation sind grundsätzlich auch dann IRIs gemeint, wenn die Bezeichnungen URL verwendet wird.

WGS 84 World Geodetic System 1984. Ein weltweites Referenzsystem für die Interpretation von Geokoordinaten-Angaben.

7 JSON-LD-Ressourcen auf oparl.org

TODO: Beschreiben, wo weitere Informationen über JSON-LD-Kontextdokumente zu finden sein werden.