OParl Schnittstellen-Spezifikation (Entwurf)

OParl Team - http://oparl.de/

Contents

1 Einleitung **4**

1.1 Status . . . 4

1.2 Was ist OParl? . . . 4

1.3 Zielsetzung von OParl . . . 5

1.4 Transparenz und Beteiligung durch Open Data . . . 6

1.5 Werdegang von OParl 1.0 . . . 6

1.6 Zukunft von OParl . . . 7

1.7 Nomenklatur der Spezifikation und Satzkonventionen . . . 7

1.7.1 MÜSSEN, SOLLEN und KÖNNEN bzw. ZWINGEND, EMPFOHLEN und OPTIONAL . . . 7

1.7.2 Besondere Hervorhebungen und Satzkonventionen . . . 8

1.8 Initiatoren . . . 8

1.9 Unterstützer . . . 8

1.10 Autoren . . . 8

2 Architektur **8**

2.1 Überblick . . . 9

2.2 Parlamentarisches Informationssystem . . . 9

2.3 Server . . . 9

2.4 API . . . 10

2.5 Client . . . 10

2.6 Cache . . . 10

2.7 Nutzerin oder Nutzer . . . 10

2.8 Objekt . . . 10

3 Nutzungsszenarien **11**

3.0.1 Szenario 1: Mobile Client-Anwendung . . . 11

3.0.2 Szenario 2: Integration in Web-Portal . . . 12

3.0.3 Szenario 3: Meta-Suche . . . 12

3.0.4 Szenario 4: Forschungsprojekt Themen- und Sprachanalyse . . . 12

4 Prinzipien und Funktionen der API **12**

4.1 Designprinzipien . . . 13
4.1.1 Aufbauend auf der gängigen Praxis . . . 13
4.1.2 Verbesserungen gegenüber dem Status Quo wo möglich . . . 13
4.1.3 RESTful . . . 13
4.1.4 Selbstbeschreibungsfähigkeit . . . 14
4.1.5 Erweiterbarkeit . . . 14
4.1.6 Browseability/Verlinkung . . . 14
4.2 Zukunftssicherheit . . . 15
4.3 HTTP und HTTPS . . . 15
4.4 URLs . . . 15
4.4.1 URL-Kanonisierung . . . 16
4.4.2 Langlebigkeit . . . 17
4.5 Serialisierung mittels JSON-LD und JSONP . . . 18
4.5.1 JSON . . . 18
4.5.2 JSON-LD . . . 18
4.5.3 JSONP . . . 21
4.6 Benannte und anonyme Objekte . . . 21
4.6.1 Benannte Objekte . . . 22
4.6.2 Anonyme Objekte (Blank Nodes) . . . 22
4.7 Objektlisten . . . 23
4.7.1 Gezieltes Abfragen von Listen . . . 23
4.7.2 Listen als Eigenschaften von Objekten . . . 25
4.8 Feeds . . . 25
4.8.1 Der Feed “Neue Objekte” . . . 25
4.8.2 Der Feed “Geänderte Objekte” . . . 26
4.8.3 Der Feed “Entfernte Objekte” . . . 27
4.9 Dokumentenabruf . . . 27
4.10 Ausnahmebehandlung . . . 27
4.11 Liste reservierter URL-Parameter . . . 27

5 Schema **28**

5.1 Übergreifende Aspekte . . . 28
5.1.1 null-Werte . . . 28
5.1.2 Vererbung der Lizenzbedingung . . . 28
5.1.3 Die Eigenschaften “created” und “last_modified” . . . 28
5.1.4 Die Eigenschaften “name” und “name_long” . . . 28
5.1.5 Die Eigenschaft “description” . . . 28
5.2 OParlSystem (System) . . . 28
5.2.1 Eigenschaft `oparl_version` . . . 29

5.2.2 Eigenschaft `bodies` . . . 29
5.2.3 Eigenschaft `name` . . . 29
5.2.4 Eigenschaft `contact` . . . 29
5.2.5 Eigenschaft `license` . . . 29
5.2.6 Eigenschaft `new_objects` . . . 29
5.2.7 Eigenschaft `updated_objects` . . . 30
5.2.8 Eigenschaft `removed_objects` . . . 30
5.2.9 Eigenschaft `info` . . . 30
5.2.10 Eigenschaft `vendor` . . . 30
5.2.11 Eigenschaft `product` . . . 30
5.2.12 Beispiel . . . 30
5.3 oparl:Body (Körperschaft) . . . 31
5.3.1 Eigenschaft `system` . . . 31
5.3.2 Eigenschaft `name` . . . 31
5.3.3 Eigenschaft `name_long` . . . 31
5.3.4 Eigenschaft `url` . . . 31
5.3.5 Eigenschaft `rgs` . . . 31
5.3.6 Eigenschaft `gnd_url` . . . 32
5.3.7 Eigenschaft `contact` . . . 32
5.3.8 Eigenschaft `papers` . . . 32
5.3.9 Eigenschaft `people` . . . 32
5.3.10 Eigenschaft `meetings` . . . 32
5.3.11 Eigenschaft `committees` . . . 32
5.3.12 Beispiel . . . 32
5.4 oparl:Committee (Gremium) . . . 33
5.4.1 Eigenschaften . . . 33
5.4.2 Beziehungen . . . 33
5.4.3 Beispiel . . . 33
5.5 oparl:Person (Person) . . . 34
5.5.1 Eigenschaften . . . 34
5.5.2 Beziehungen . . . 34
5.5.3 Beispiel . . . 34
5.6 oparl:Organization (Organisation) . . . 35
5.6.1 Eigenschaften . . . 35
5.6.2 Beziehungen . . . 35
5.6.3 Beispiel . . . 36
5.7 oparl:Meeting (Sitzung) . . . 36
5.7.1 Eigenschaften . . . 36
5.7.2 Beziehungen . . . 36

5.7.3 Beispiel . . . 37
5.8 oparl:AgendaItem (Tagesordnungspunkt) . . . 37
5.8.1 Eigenschaften . . . 37
5.8.2 Beziehungen . . . 38
5.8.3 Beispiel . . . 38
5.9 oparl:Paper (Drucksache) . . . 38
5.9.1 Eigenschaften . . . 39
5.9.2 Beziehungen . . . 39
5.9.3 Beispiel . . . 39
5.10 oparl:Document (Dokument) . . . 40
5.10.1 Eigenschaften . . . 40
5.10.2 Beziehungen . . . 41
5.11 oparl:Consultation (Beratung) . . . 41
5.12 oparl:Location (Ort) . . . 41
5.12.1 Eigenschaften . . . 41
5.12.2 Beziehungen . . . 41
5.13 oparl:Contact (Kontakt) . . . 42

6 Fußnoten **42**

7 Glossar **42**

Lizenz: Creative Commons CC-BY-SA

1 Einleitung

Dieses Dokument wird bei seiner Fertigstellung die Spezifikation des OParl Schnittstellen-Standards für parlamentarische Informationssysteme (Ratsinformationssysteme, RIS) 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.

1.2 Was ist OParl?

(Nachfolgend eine Übernahme aus dem bisherigen Abschnitt “Funktionsumfang der OParl-Schnittstelle”. Der Text sollte deutlich überarbeitet und erweitert werden.)

Die vorliegende Spezifikation soll eine Webservice-Schnittstelle definieren, die den anonymen und lesenden Zugriff auf öffentliche Inhalte aus Parlamentarischen Informationssystemen ermöglicht. Die Zugriffe erfolgen über das Hypertext Transfer Protocol (HTTP). Daten werden als JSON oder als JSONP ausgeliefert.

Die Spezifikation wird obligatorische Bestandteile (MUSS) und optionale Bestandteile (KANN) haben. Der tatsächliche Funktionsumfang kann daher zwischen den Implementierungen variieren.

1.3 Zielsetzung von OParl

- Nutzen für Kommunen, Bürger, politische Parteien
- Nutzen für Anbieter von RIS-Pflegesoftware
- Nutzen für Anbieter von RIS-Darstellungssoftware
- Nutzen für Open Data Initiativen
- Nutzen für die Wissenschaft
- Linked Data erwähnen

(Nachstehen der bisherige Textentwurf aus dem Abschnitt "Motivationen für den standardisierten Datenzugriff", der sicherlich deutlich überarbeitet werden muss.)

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

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[14], Offenes Köln[15] oder der OpenRuhr:RIS-Instanzen[16] 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 könnten sein:

- 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.

1.4 Transparenz und Beteiligung durch Open Data

- Einführung zu Open Data
- "10 Principles" erwähnen https://sunlightfoundation.com/policy/documents/ten-open-data-principles/
- Spürbare Tendenz zu mehr Offenheit
- Open Data ist (nur) zum Teil technisches Problem
- Vier der "10 Principles" haben technische Dimension:
  - 5. Machine readability
  - 6. Non-discrimination
  - 7. Use of Commonly Owned Standards
  - 9. Permanence
- Die bloße Bereitstellung einer OParl-konformen API wird weder die Einhaltung der technischen Prinzipien, noch der weiteren Open-Data-Prinzipien vollständig garantieren. Für echte Transparenz bedarft es mehr.
- 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 den Verwaltungen und Parlamentariern, 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.

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.

- 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.

1.6 Zukunft von OParl

- Verfeinerung, Lücken schliessen
- Globalisierung
- 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
- 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.

1.7 Nomenklatur der Spezifikation und Satzkonventionen

1.7.1 MÜSSEN, SOLLEN und KÖNNEN bzw. ZWINGEND, EMPFOHLEN und OPTIONAL

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.[1]

Die Bedeutung im Einzelnen:

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

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 Besondere Hervorhebungen und Satzkonventionen

1.8 Initiatoren

1.9 Unterstützer

1.10 Autoren

An diesem Dokument haben mitgewirkt:

Felix Ebert, Jan Erhardt, Jens Klessmann, Andreas Kuckartz, Babett Schalitz, Marian Steinbach, 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.[2]

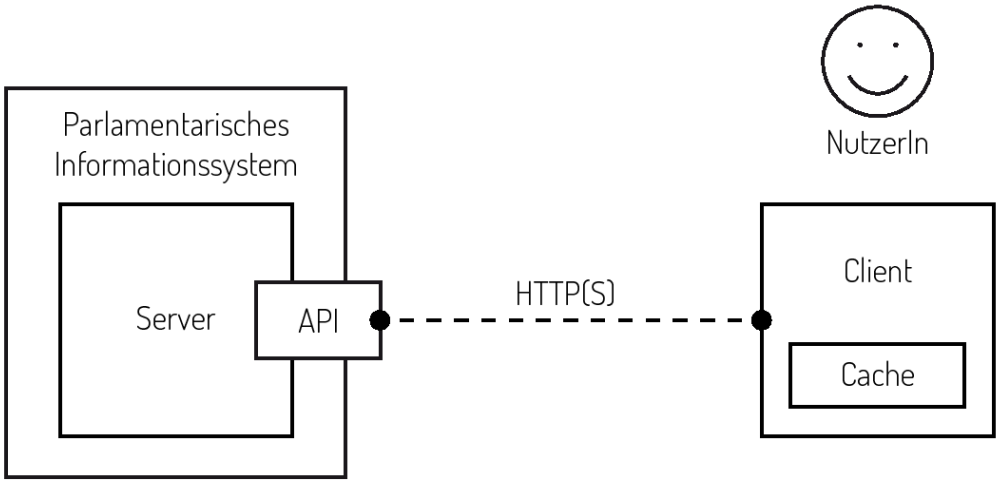

Figure 1: Architekturdiagramm

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.

Im kommunalen Umfeld in Deutschland, wo das Parlament je nach Art der Kommune häufig als Stadtrat oder Gemeinderat bezeichnet wird, hat sich für diese Art von Informationssystem auch der Begriff "Ratsinformationssystem" (kurz "RIS") etabliert.

Parlamentarische Informationssysteme sind jedoch nicht auf die kommunale Ebene begrenzt. Ähnliche Systeme werden auch auf Ebene z.B. von Landkreisen, Regierungsbezirken und diversen Zweckverbänden eingesetzt.

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

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-

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

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.

3.0.1 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 bestmöglichen Lesezugriff auf Dokumente aus Ratsinformationssystemen (RIS) 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 Ratsinformationssysteme, 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 hinaus gehen 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.

3.0.2 Szenario 2: Integration in Web-Portal

Web Portale bieten Nutzern unter anderem die Möglichkeit Anwendungen, Prozesse und Dienste zu integrieren. Die OParl API stellt einen solchen Dienst dar und bereitet so den Weg zu angereicherten Portalseiten. Informationen, die über die API bezogen werden, können in Portlets organisiert und visualisiert werden. Hierbei können

1. angemeldete Benutzer

die eingegrenzten Portlet Parameter für den nächsten Besuch zwischen speichern, während

2. anonyme Benutzer

dies nicht können. In beiden Fällen können Portalnutzer das angezeigte Portlet nach ihren Bedürfnissen anpassen. Beispielsweise kann ein solches Portlet eine Liste der Gremien bereitstellen, aus der sich der Nutzer das interessante Gremium aussucht und aufgrund dieser Auswahl die Informationen zu den vergangenen / nächsten Sitzungsterminen im Rat, etwaiger Drucksachen oder Dokumenten erhält und geeignet visualisiert.

Durch eine solche Integration von RIS Informationen in bestehende Portalsysteme (unter Umständen die kommunale Webseite selbst), ist es möglich Nutzern zusätzliche Informationen in der bereits gewohnten Umgebung zu präsentieren und den bestehenden Informationsgehalt und den Datenbestand aufzuwerten.

3.0.3 Szenario 3: Meta-Suche

3.0.4 Szenario 4: Forschungsprojekt Themen- und Sprachanalyse

4 Prinzipien und Funktionen der API

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

Stichpunkte:

- Grundlage für den Zugriff auf die Schnittstelle ist das Hypertext Transfer Protocol (HTTP).

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

4.1 Designprinzipien

4.1.1 Aufbauend auf der gängigen Praxis

Grundlage für die Erarbeitung der OParl-Spezifikation in der vorliegenden Version ist eine Analyse der aktuell (2012 bis 2014) in Deutschland befindlichen Ratsinformationssysteme und ihrer Nutzung. Erklärtes Ziel für diese 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 Verbesserungen 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 Ratsinformationssystem[3] 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, beliebige Geodaten-Objekte entsprechend der GeoJSON Spezifikation[4] 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[5]. 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

[3] Das System BoRis der Stadt Bonn http://www2.bonn.de/bo_ris/ris_sql/agm_index.asp
[4] GeoJSON http://geojson.org/
[5] Fielding, Roy: Architectural Styles and the Design of Network-based Software Architectures, http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

- 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.

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 (TODO: Verweis einfügen) 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[6].

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

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.

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.

4.4 URLs

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

Die grundsätzliche Funktionsweise von URLs ist in RFC3986 beschrieben[7].

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

```
http://refserv.oparl.org/bodies/0/committees/4/members/?skip=234
\__/   \_____________/\_____________________________/ \_____/
 |         |                 |                          |
Schema    Host              Pfad                    Query-String
```

[7] http://tools.ietf.org/html/rfc3986

4.4.1 URL-Kanonisierung

Absicht ist, dass jedes benannte Objekt, 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 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.ratsinformation.stadt-koeln.de/
```

kommuniziert beispielsweise direkt die Zugehörigkeit zur Stadt Köln als Betreiber des Systems. Die Bezeichnung “ratsinformation” in der Subdomain zeigt den Zweck des Systems allgemein verständlich an. Der Host-Name “oparl.ratsinformation.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 http://oparl.ratsinformation.stadt-koeln.de/ wären eine Reihe von nicht-kanonischen URL-Varianten denkbar, die technischen auf den selben Server führen könnten:

- http://83.123.89.102/
- http://oparl.ratsinformation.stadtkoeln.de/

- http://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

```
http://oparl.meinris.de/members?body=1&committee=2
http://oparl.meinris.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 Ressourcennutzugn 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[8] sowie die Europäische Kommission[9].

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

[9] Study on persistent URIs, with identification of best practices and recommendations on the topic for the MSs and the EC. (PDF) http://goo.gl/JaTq6Z

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.

4.5.1 JSON

Die Abkürzung JSON steht für "JavaScript Object Notation". Das JSON-Format ist in RFC4627[10] 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. Entsprechend erweitert die JSON-LD-Spezifikation[11] 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.

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:

[10] https://tools.ietf.org/html/rfc4627
[11] http://www.w3.org/TR/json-ld/

```
{
  "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[12]. 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 Objects schon mehr Daten umfasst, als die eigentlichen Objekteigenschaften. Sinnvollerweise kann jedoch der gesamte Inhalt des `@context`-Teils in eine externe Ressource ausgelagert werden. Clients, die eine Vielzahl von gleichartigen Objekten laden und interpretieren wollen, müssen diese Ressource dann nur einmal laden. Das Ergebnis könnte so aussehen:

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

[12] 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.

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 Präfix `oparl` repräsentiert, das zweite (herstellereigene) durch das Präfix `vendor`. Ein JSON-LD-Client setzt Präfix und Typenbezeichnung letztlich wieder zu einer URL zusammen.

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

TODO: Ab hier weiter ausformulieren

Darüber hinaus stellt JSON-LD zusätzliche Anforderungen an JSON-Daten, die in diesem Abschnitt weiter ausgeführt werden sollen...

- Einschränkungen von OParl gegenüber JSON-LD
- Schlüssel in einem JSON-LD-Objekt müssen einzigartig sein.
- Unterscheidung von Groß- und Kleinschreibung
- Benannte Objekte (URL als Schlüssel)
- Anonyme Objekte (Blank Nodes)
- Mime Type application/ld+json
- Kompakte Form mit Verwendung externer @context-URL als SOLL-Anforderung
- Verweis auf http://www.bmi.bund.de/SharedDocs/Downloads/DE/Themen/OED_Verwaltung/ModerneVerwa
- Siehe https://github.com/OParl/specs/issues/77
- Siehe https://github.com/OParl/specs/issues/10

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[13]. 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.

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.

[13] TODO: URL zur Spezifikation

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

`http://refserv.oparl.org/bodies/0/committees/1`

abruft:

```
{
    "@id": "http://refserv.oparl.org/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. 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": "http://refserv.oparl.org/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": "http://refserv.oparl.org/bodies/0/committees/1",
            "meeting": "http://refserv.oparl.org/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 Objekte oder Listen von Objekten abgefragt werden.

4.7.1 Gezieltes Abfragen von Listen

Fragt ein Client eine Liste von Objekten an, beispielsweise die Liste aller Drucksachen in einem System, kann der Server innerhalb bestimmter Grenzen entscheiden, wie die Ausgabe aussieht.

In der einfachsten Form gibt der Server ein Objekt mit nur einer Eigenschaft `items` aus. Der Wert dieser Eigenschaft ist die **vollständige Liste** der URLs aller angefragten Objekte. Beispiel:

```
{
    "items": [
        "http://refserv.oparl.org/bodies/0/papers/2",
        "http://refserv.oparl.org/bodies/0/papers/5",
        "http://refserv.oparl.org/bodies/0/papers/7",
    ]
}
```

Diese einfachste Form der Antwort eignet sich nur für Listen mit einer begrenzten Anzahl von Einträgen.

Für längere Listen ist eine Blätterfunktion bzw. Paginierung vorgesehen. Darunter versteht man den seitenweisen Abruf der Einträge einer Liste, wobei die Reihenfolge der Liste vom Server festgelegt ist und zwischen den Seitenabrufen unverändert bleibt.

Listen mit mehr als 100 Einträgen SOLL der Server nur teilweise ausgeben und dem Client dabei eine **Paginierung** anbieten, um weitere Listenteile abzurufen. Dabei wird EMPFOHLEN, die Zahl der jeweils ausgegebenen Listeneinträge wiederum auf maximal 100 zu begrenzen.

Das nachstehende Beispiel zeigt, wie dem Client die URL zum “Blättern”, also zum Aufruf der nächsten Listenseite, angeboten wird.

```
{
    "items": [
        "http://refserv.oparl.org/bodies/0/papers/2",
        "http://refserv.oparl.org/bodies/0/papers/5",
        "http://refserv.oparl.org/bodies/0/papers/7",
    ],
    "nextpage": "http://refserv.oparl.org/bodies/0/papers/?skip=7",
    "count": 7
}
```

Wie oben zu sehen, enthält das Beispiel-Objekt nun eine zusätzliche Eigenschaft `nextpage`. Der Wert dieser Eigenschaft ist eine URL, die dem Client dazu dient, die weiteren Einträge der Liste abzurufen.

Die Eigenschaft `count` DARF bei Listen grundsätzlich ausgegeben werden und SOLL bei mehrseitigen Listen ausgegeben werden. Ihr Wert ist eine Zahl und gibt an, wie viele Einträge die vollständige Liste enthält.

Ruft der Client die unter `nextpage` angegebene URL auf, erhält er wiederum ein Listenobjekt. Dieses Objekt MUSS, sofern noch immer mehr Listeneinträge vorhanden sind, als ausgegeben wurden, wiederum die `nextpage` Eigenschaft mit einer URL enthalten. Um alle Einträge

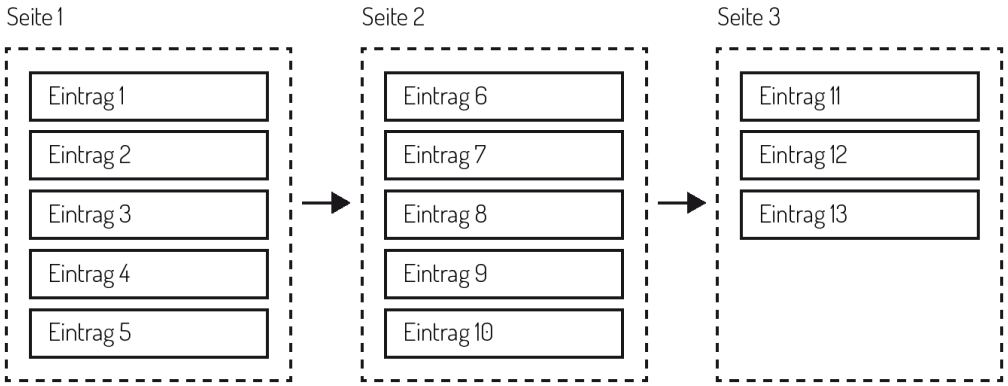

Figure 2: Paginierung: Schematische Darstellung

einer Liste zu erfassen, folgt der Client also jeweils der URL, die in der `nextpage` Eigenschaft angegeben ist.

Server-Implementierer entscheiden selbst, wie die `nextpage`-URL aufgebaut ist und tragen damit selbst Verantwortung für die Funktionsweise der Paginierung. Bei der Entscheidung für eine Form der Implementierung sind weitere Anforderungen zu berücksichtigen:

- 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 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.

Eine ungünstige (unstabile) Form der Implementierung soll hier 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. Sofern sich an der Tabelle zwischen den beiden Abfragen nichts geändert hat, liefert die zweite Abfrage Datensätze mit `id` > 10 aus. Sollte zwischen den beiden Abfragen jedoch beispielsweise der Datensätze 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 instabile 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 `id`s 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
```

TODO: Bestimmte Listen können mit Einschränkung auf einen Datumsbereich abgefragt werden. Mehr dazu in https://github.com/OParl/specs/issues/30 Fraglich ist, ob das in diesem Kapitel behandelt werden sollte oder in einem anderen.

4.7.2 Listen als Eigenschaften von Objekten

TODO: Listen können auch als Werte von Objekteigenschaften auftreten. Hierbei gibt es keine Paginierung, sondern es müssen alle URLs aufgelistet werden. Das ist auszuformulieren und mit Beispielen zu zeigen.

4.8 Feeds

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

Der Begriff "Feed" ist eine Anlehnung an die weit verbreiteten RSS- oder Atom-Feeds, 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, schnelle und ressourcenschonende abzufragen, welche Objekte auf dem Server neu hinzugefügt, geändert oder entfernt wurden. Ziel ist, zu verhindern, dass Clients zur Aktualisierung ihres Caches den gesamten Datenbestand eines Servers abrufen müssen.

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

Für alle Feeds drei gilt, dass mindestens ein Zeitraum von 365 Tagen, gerechnet vom Zeitpunkt der Abfrage, abgedeckt werden SOLL.

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": "http://refserv.oparl.org/bodies/0/papers/21/documents/3",
            "created": "2014-01-07T12:59:01.038+0100"
        },
        {
            "@id": "http://refserv.oparl.org/bodies/0/papers/21",
            "created": "2014-01-05T18:29:37.123+0100"
        },
        {
            "@id": "http://refserv.oparl.org/bodies/0/papers/20/documents/5",
            "created": "2014-01-04T11:26:48.638+0100"
        },
        ...
    ],
    "nextpage": "http://refserv.oparl.org/feeds/new/?t=20140106170100402"
}
```

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

Wie für Objektlisten üblich, SOLL auch für Feeds automatisch eine Aufteilung auf mehrere Seiten vorgenommen und ein Paginierungs-Link angeboten werden, um die übertragenen Datenmengen je Abruf einzugrenzen.

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 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": "http://refserv.oparl.org/bodies/0/papers/0/documents/2",
            "last_modified": "2014-01-08T14:28:31.568+0100"
        },
        {
            "@id": "http://refserv.oparl.org/bodies/0/papers/0",
            "last_modified": "2014-01-08T12:14:27.958+0100"
        },
        {
            "@id": "http://refserv.oparl.org/bodies/0/papers/0/documents/1",
            "last_modified": "2014-01-06T17:01:00.402+0100"
        },
        ...
    ],
    "nextpage": "http://refserv.oparl.org/feeds/updated/?t=20140106170100402"
}
```

Das Ausgabeformat entspricht weitgehend dem des Feeds "Neue Objekte", jedoch heißt hier die Eigenschaft für den Zeitpunkt der letzten Änderung `last_modified`. Auch hier gilt, dass der als `last_modified` 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": "http://refserv.oparl.org/bodies/0/people/22",
            "removed": "2013-11-11T11:11:00.000+0100"
        },
        ...
    ],
    "nextpage": "http://refserv.oparl.org/feeds/updated/?t=20131111111100"
}
```

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

4.9 Dokumentenabruf

- HTTP GET Methode MUSS unterstützt werden
- HEAD-Methode MUSS unterstützt werden
- HTTP Last-Modified Header sowie Conditional GET sind zu unterstützen

4.10 Ausnahmebehandlung

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

4.11 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 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.

5.1.2 Vererbung der Lizenzbedingung

- Jedes Objekt KANN die Eigenschaft "license" besitzen.
- Die genannte Lizenz bezieht sich auf das jeweilige Objekt und auf untergeordnete Objekte, sofern diese keine license-Eigenschaft besitzen.
- Dazu muss die Vererbungshierarchie aufgezeigt werden.
- Empfohlene Minimalvariante: Nur eine license-Angabe auf Ebene von `oparl:System`.
- Auf Ebene des `oparl:Document` bezieht sich die Eigenschaft sowohl auf die Metadaten als auch auf das Dokument selbst.

5.1.3 Die Eigenschaften "created" und "last_modified"

5.1.4 Die Eigenschaften "name" und "name_long"

5.1.5 Die Eigenschaft "description"

5.2 OParlSystem (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.

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

5.2.1 Eigenschaft `oparl_version`

Diese Eigenschaft ist ZWINGEND.

URL der Version der OParl-Spezifikation, die von diesem System unterstützt wird. So lange es nur die Version 1.0 der OParl-Spezifikation gibt, MUSS der Wert dieser Eigenschaft `http://oparl.org/specs/1.0/` sein.

Sofern zukünftig weitere Versionen der Spezifikation vorliegen, können Clients damit in Erfahrung bringen, welches Schema, welche Eigenschaften und Methoden auf Seite des Servers vorausgesetzt werden können.

5.2.2 Eigenschaft `bodies`

Diese Eigenschaft ist ZWINGEND.

Über diese URL sind alle `oparl:Body` Objekte, also die im System geführten Körperschaften, als Liste abrufbar.

TODO: Verweis auf `oparl:Body` einfügen

5.2.3 Eigenschaft `name`

Diese Eigenschaft wird EMPFOHLEN.

Diese Eigenschaft dient dazu, einen nutzerfreundlichen Namen zu kommunizieren, mit dem NutzerInnen das System wiedererkennen und von anderen unterscheiden können.

5.2.4 Eigenschaft `contact`

Diese Eigenschaft wird EMPFOHLEN.

Die Eigenschaft dient dazu, NutzerInnen bzw. EntwicklerInnen von Clients die Kontaktaufnahme mit dem Betreiber des Systems zu ermöglichen. Es wird EMPFOHLEN, hier die Kontaktdaten eines technischen Ansprechpartners bzw. einer allgemeinen Kontaktstelle auszugeben, über die Anfragen verschiedener Art an die richtige Kontaktperson umgeleitet werden können.

Der Wert dieser Eigenschaft MUSS ein Objekt vom Typ `oparl:Contact` sein.

TODO: Verweis auf `oparl:Contact` einfügen.

5.2.5 Eigenschaft `license`

Diese Eigenschaft wird EMPFOHLEN.

Die Eigenschaft dient dazu, darüber zu informieren, unter welcher Lizenz die Daten des aktuell angezeigten Objekts stehen. Zur Vererbung dieser Eigenschaft siehe (TODO: Verweis auf Abschnitt zur Lizenz-Vererbung einfügen).

Der Wert dieser Eigenschaft sollte nach Möglichkeit eine URL sein, unter der genau die entsprechende Lizenz abgerufen werden kann.

5.2.6 Eigenschaft `new_objects`

Diese Eigenschaft ist EMPFOHLEN.

Mit dieser Eigenschaft wird die URL des Feeds für neu hinzugekommene Objekte ausgegeben.

TODO: Verweis auf Feeds > Neue Objekte

5.2.7 Eigenschaft `updated_objects`

Diese Eigenschaft ist EMPFOHLEN.

Mit dieser Eigenschaft wird die URL des Feeds für geänderte Objekte ausgegeben.

TODO: Verweis auf Feeds > Geänderte Objekte

5.2.8 Eigenschaft `removed_objects`

Diese Eigenschaft ist EMPFOHLEN.

Mit dieser Eigenschaft wird die URL des Feeds für entfernte Objekte ausgegeben.

TODO: Verweis auf Feeds > Entfernte Objekte

5.2.9 Eigenschaft `info`

Diese Eigenschaft ist OPTIONAL.

Diese Eigenschaft dient dazu, eine zusätzliche URL zu einer WWW-Seite mit zusätzlichen Informationen zum System anzubieten. So könnten NutzerInnen beispielsweise auf eine Web-Oberfläche eines parlamentarischen Informationssystems geführt werden.

5.2.10 Eigenschaft `vendor`

Diese Eigenschaft ist OPTIONAL.

Diese Eigenschaft dient dazu, über eine URL den Hersteller des Server-Systems zu komunizieren. Die URL sollt nach Möglichkeit zu einer WWW-Seite mit weiteren Informationen zum Hersteller führen.

5.2.11 Eigenschaft `product`

Diese Eigenschaft ist OPTIONAL.

Diese Eigenschaft dient dazu, über eine URL mitzuteilen, welches Softwareprodukt das Server-System bereitstellt. Die URL soll nach Möglichkeit zu einer WWW-Seite mit weiteren Informationen zum Produkt führen.

5.2.12 Beispiel

```
{
    "@id": "http://refserv.oparl.org/",
    "@context": "http://oparl.org/schema/1.0/System",
    "oparl_version": "http://oparl.org/specs/1.0/",
    "name": "OParl Reference Server",
    "info": "https://github.com/OParl/reference-server",
    "contact": {
        "email": "info@oparl.org",
        "name": "Common OParl contact"
    },
    "vendor": "http://oparl.org/",
    "product": "https://github.com/OParl/reference-server",
    "license": "http://creativecommons.org/licenses/by/4.0/",
    "bodies": "http://refserv.oparl.org/bodies/",
    "new_objects": "http://refserv.oparl.org/feeds/new/",
    "updated_objects": "http://refserv.oparl.org/feeds/updated/",
    "removed_objects": "http://refserv.oparl.org/feeds/removed/"
}
```

5.3 oparl:Body (Körperschaft)

Dieser Objekttyp erlaubt es, eine Körperschaft abzbilden. Eine Körperschaft kann beispielsweise eine Gemeinde, ein Landkreis oder ein Zweckverband sein.

Von einem funktionsfähigen 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 an.

Es werden mehrere Eigenschaften angeboten, die dazu dienen, die real existierende Körperschaft, die von einem `oparl:Body` Objekt repräsentiert wird, programmatisch auslesbar zu machen zu können. Insbesondere sind hier die Eigenschaften `url`, `rgs` und `gnd_url` zu nennen.

5.3.1 Eigenschaft `system`

Diese Eigenschaft ist ZWINGEND.

Mit dieser Eigenschaft wird das Objekt dem übergeordneten `oparl:System` Objekt zugeordnet. Wert MUSS der IRI des `oparl:System` Objekts sein.

5.3.2 Eigenschaft `name`

Diese Eigenschaft ist ZWINGEND. Sie transportiert den gebräuchlichen Namen der Körperschaft.

5.3.3 Eigenschaft `name_long`

Diese Eigenschaft ist OPTIONAL und kann bei Bedarf dazu verwendet werden, eine längere Form des Namens der Körperschaft wieder zu geben, sofern dieser für die Eigenschaft `name` zu lang ist.

5.3.4 Eigenschaft `url`

Diese Eigenschaft ist EMPFOHLEN.

Mit dieser Eigenschaft SOLL die URL der offiziellen Website der Körperschaft ausgegeben werden.

TODO: Beschreibung

5.3.5 Eigenschaft `rgs`

Diese Eigenschaft ist EMPFOHLEN.

Handelt es sich bei der Körperschaft um eine Gebietskörperschaft (Landkreis, Kommune etc.) in Deutschland, SOLL für die eindeutige Identifizierung der amtliche Regionalschlüssel verwendet werden.[14] Dieser ist grundsätzlich zwölfstellig.

[14] Regionalschlüssel können im Gemeindeverzeichnis (GV-ISys) des Statistischen Bundesamtes eingesehen werden

5.3.6 Eigenschaft `gnd_url`

Diese Eigenschaft ist EMPFOHLEN.

Sofern die Körperschaft in der GND[15] vertreten ist, SOLL diese Eigenschaft als Wert die URL des Eintrags in der GND enthalten.

5.3.7 Eigenschaft `contact`

Diese Eigenschaft ist EMPFOHLEN.

Über diese Eigenschafte SOLLEN Kontaktinformationen zu einer Stelle bereit gestellt werden, die die inhaltliche Verantwortung für sämtliche zu dieser Körperschaft gehörenden Inhalte im System trägt. Besonders wichtig ist diese Angabe, wenn auf einem System mehrere Körperschaften vertreten sind und damit auf der Ebene des `oparl:System` Objekts ein rein technischer Kontakt ausgegeben wird, der nicht für inhaltliche Fragestellungen im Zuständigkeitsbereich der jeweiligen Körperschaften kontaktiert werden sollte.

5.3.8 Eigenschaft `papers`

Diese Eigenschaft ist ZWINGEND.

Wert dieser Eigenschaft ist die URL der API zum Aufruf einer Liste der Drucksachen (Objekte vom Typ `oparl:Paper`) für diese Körperschaft.

5.3.9 Eigenschaft `people`

Diese Eigenschaft ist ZWINGEND.

Wert dieser Eigenschaft ist die URL der API zum Aufruf einer Liste der Personen (Objekte vom Typ `oparl:Person`) für diese Körperschaft.

5.3.10 Eigenschaft `meetings`

Diese Eigenschaft ist ZWINGEND.

Wert dieser Eigenschaft ist die URL der API zum Aufruf einer Liste der Sitzungen (Objekte vom Typ `oparl:Meeting`) für diese Körperschaft.

5.3.11 Eigenschaft `committees`

Diese Eigenschaft ist ZWINGEND.

Wert dieser Eigenschaft ist die URL der API zum Aufruf einer Liste der Gremien (Objekte vom Typ `oparl:Committee`) für diese Körperschaft.

5.3.12 Beispiel

```
{
    "@id": "http://refserv.oparl.org/bodies/0",
    "@context": "http://oparl.org/schema/1.0/Body",
    "committees": "http://refserv.oparl.org/bodies/0/committees/",
    "contact": {
        "email": "ris@stadt-koeln.de",
        "name": "RIS-Betreuung"
    },
```

[15] Gemeinsame Normdatei http://www.dnb.de/gnd

```
    "created": "2014-01-08T14:28:31.568+0100",
    "gnd_url": "http://d-nb.info/gnd/2015732-0",
    "last_modified": "2014-01-08T14:28:31.568+0100",
    "meetings": "http://refserv.oparl.org/bodies/0/meetings/",
    "name": "Stadt K\u00f6ln",
    "name_long": "Stadt K\u00f6ln, kreisfreie Stadt",
    "organisations": "http://refserv.oparl.org/bodies/0/organisations/",
    "papers": "http://refserv.oparl.org/bodies/0/papers/",
    "people": "http://refserv.oparl.org/bodies/0/people/",
    "rgs": "053150000000",
    "system": "http://refserv.oparl.org/",
    "url": "http://www.stadt-koeln.de/"
}
```


5.4 oparl:Committee (Gremium)

Das Gremium ist ein Personenkreis, üblicherweise von gewählten und/oder ernannten Mitgliedern. Beispiele hierfür sind der Stadtrat, Kreisrat, Gemeinderat, Ausschüsse und Bezirksvertretungen. Gremien halten Sitzungen ab, zu denen die Gremien-Mitglieder eingeladen werden.

5.4.1 Eigenschaften

Schlüssel (`id`) Zur eindeutigen Identifizierung des Gremiums im Kontext einer bestimmten Körperschaft. In der Praxis kommen sowohl numerische IDs als auch Namenskürzel (Beispiel: "STA" für den Stadtentwicklungsausschuss) vor. Beides sollte hier Verwendung finden können.

Name (`name`) Der Name des Gremiums. Beispiele: "Rat", "Hauptausschuss", "Bezirksvertretung 1 (Innenstadt)"

Kurzname (`short_name`) *Optional.* Eine zur Anzeige bestimmte, kürzere Form des Namens.

Zuletzt geändert (`last_modified`) Datum und Uhrzeit der letzten Änderung

5.4.2 Beziehungen

- Objekte vom Typ `oparl:Person` referenzieren auf Gremien, um die Mitgliedschaft/Zugehörigkeit einer Person im/zum Gremium zu kennzeichnen. Diese Beziehung ist datiert. Das bedeutet, sie hat einen Anfangszeitpunkt und ggf. einen Endzeitpunkt.
- Objekte vom Typ "Drucksache" verweisen auf Gremien. Beispielsweise wird eine Anfrage oder ein Antrag dem Rat und/oder einer bestimmten Bezirksvertretung zugeordnet. Details zu dieser Beziehung werden unter "Drucksache" erläutert.
- Das Gremium verweist auf die Körperschaft, zu der das Gremium gehört.

5.4.3 Beispiel


```
{
    "id": "7",
    "name": "Finanzausschuss",
    "short_name": "FA",
    "body": "1",
    "last_modified": "2012-08-16T14:05:27+02:00"
}
```

5.5 oparl:Person (Person)

Jede natürliche Person, die Mitglied eines Gremiums ist, ist als `oparl:Person` im Datenmodell eindeutig identifizierbar.

5.5.1 Eigenschaften

Schlüssel (`id`) Zur eindeutigen Identifizierung sollte jede Person eine Kennung besitzen, die keinen Änderungen unterworfen ist und aus diesem Grund nicht mit dem Namen in Verbindung stehen sollte. Viele RIS nutzen rein numerische Kennungen.

Vorname (`first_name`) Der Vorname der Person.

Nachname (`last_name`) Der Nachname der Person.

Titel (`academic_title`) *Optional.* Akademische Titel wie "Dr." und "Prof. Dr."

Geschlecht (`sex`) *Optional.* Weiblich (Wert `F` für *female*), männlich (Wert `M` für *male*), anderes (Wert `O` für *others*)

Beruf (`profession`) *Optional.* Z.B. "Rechtsanwalt"

E-Mail-Adresse (`email`) *Optional.*

Telefon (`phone`) *Optional.*

Fax (`fax`) *Optional.*

Anschrift (`address`) *Optional.* Straße und Hausnummer, Postleitzahl und Ort

Zuletzt geändert (`last_modified`) Datum und Uhrzeit der letzten Änderung

Anmerkungen

- Das System von Euskirchen scheint Vor- und Nachname (evtl. einschl. Titel) in einem gemeinsamen Feld "Name" zu führen. Ob das System hier technisch differenziert, ist unklar. Falls einzelne Systeme den angezeigten Namen nur als ganzes speichern, sollte dies für den Standard übernommen werden, da es für die meisten Anwendungen ausreichen sollte.
- Das System PROVOX unterscheidet zwischen privaten und geschäftlichen Anschriften.

5.5.2 Beziehungen

- Objekte vom Typ `oparl:Person` können einer Organisation, z.B. einer Fraktion, zugeordnet werden. Diese Beziehung ist datiert.
- Objekte vom Typ `oparl:Person` können einem oder mehreren Gremien zugewiesen werden, um die Mitgliedschaft in diesem Gremium darzustellen. Diese Beziehungen sind ebenfalls datiert.

5.5.3 Beispiel

```
{
    "id": "1000",
    "first_name": "Max",
    "last_name": "Mustermann",
    "academic_title": "Dr.",
    "sex": "M",
    "profession": "Rechtsanwalt",
```

```
    "email": "max@mustermann.de",
    "phone": "+4977777",
    "fax": "+4988888",
    "address": "Musterstraße 5, 11111 Musterort",
    "last_modified": "2012-08-16T14:05:27+02:00",
    "organisations": [
        {
            "id": "2000",
            "start": "2011-03-01",
            "end": "2013-02-28"
        },
        {
            "id": "2001",
            "start": "2013-03-01"
        }
    ],
    "committees": [
        {
            "id": "7",
            "start": "2013-01-01"
        }
    ]
}
```

5.6 oparl:Organization (Organisation)

Organisationen sind üblicherweise Parteien bzw. Fraktionen, denen die Personen angehören können.

5.6.1 Eigenschaften

Schlüssel (`id`) Zur eindeutigen Kennzeichnung einer Organisation innerhalb des Systems

Name (`name`) Der gebräuchliche Name der Organisation, z.B. “SPD” oder “DIE LINKE”.

Zuletzt geändert (`last_modified`) Datum und Uhrzeit der letzten Änderung

Anmerkungen

- Unklar ist bislang, ob Organisationen in der Praxis eher Fraktionen (“SPD-Fraktion im Kölner Rat”, “SPD-Fraktion in Köln-Innenstadt”) abbilden oder ob eher Ortsverbände von Parteien (“SPD Köln”) gemeint sein werden. Einblicke, wie gängige Systeme dies handhaben, sollten evtl. gesammelt und berücksichtigt werden.
- Es wird die Schreibweise “Organization” (und nicht “Organisation”) verwendet, da diese in allen englischen Sprachräumen problemlos verwendet werden kann. Siehe dazu Abschnitt 3, Fussnote 1 auf dieser Seite: http://popoloproject.com/specs/organization.html

5.6.2 Beziehungen

- Jede `oparl:Organization` gehört zu einer Körperschaft.
- Personen können Organisationen angehören (*datiert*).

5.6.3 Beispiel

```
{
    "id": "15",
    "name": "SPD",
    "body": "1",
    "last_modified": "2012-08-16T14:05:27+02:00"
}
```

5.7 oparl:Meeting (Sitzung)

Eine Sitzung ist die Versammlung der Mitglieder eines Gremiums oder mehrerer Gremien zu einem bestimmten Zeitpunkt an einem bestimmten Ort.

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

5.7.1 Eigenschaften

Schlüssel (`id`) Zur eindeutigen Identifizierung der Sitzung innerhalb des Systems. In der Praxis wird ein solcher Schlüssel entweder durch eine numerische ID gebildet oder durch Kombination mehrerer Merkmale wie dem Kürzel des Gremiums, der laufenden Nummer der Sitzung in einem Jahr und der Jahreszahl (z.B. “BV1/0034/2012”).

Nummer (`sequence_number`) *Optional.* Laufende Nummer der Sitzung, üblicherweise innerhalb der Wahlperiode mit 1 beginnend. In der Praxis wird dadurch z.B. die “2. Sitzung des Rats” gekennzeichnet. Ist dieses Feld gesetzt, MUSS ein numerischer Wert enthalten sein.

Anfang (`start`) Datum und ggf. Uhrzeit des Anfangszeitpunkts der Sitzung

Ende (`end`) *Optional.* Datum und Uhrzeit vom Ende der Sitzung

Ort (`address`) *Optional.* Textliche Information zum Ort der Sitzung, z.B. “Rathaus, Raum 136”.

Zuletzt geändert (`last_modified`) Datum und Uhrzeit der letzten Änderung

5.7.2 Beziehungen

- Sitzungen sind mindestens einem Gremium zugeordnet
- Einer Sitzung sind Personen zugeordnet, um die Teilnahme an der Sitzung auszudrücken.
- Dokumente können vom Typ `oparl:Meeting` *optional* zu mehreren Zwecken referenziert werden:
  - Zum Verweis auf die Einladung zur Sitzung
  - Zum Verweis auf das Ergebnisprotokoll zur Sitzung
  - Zum Verweis auf das Wortprotokoll zur Sitzung
- Weiterhin können Sitzungen beliebige weitere Dokumente, die keine eigenständigen Drucksachen sind, referenzieren. Dabei handelt es sich dann um nicht weiter spezifizierte Anlagen.

5.7.3 Beispiel


```
{
    "id": "3271",
    "identifier": "STA/0034/2012",
    "start": "2013-01-04T08:00:00+01:00",
    "end": "2013-01-04T12:00:00+01:00",
    "address": "Rathaus, Raum 136",
    "sequence_number": 1,
    "committees": ["STA"],
    "people": ["1000", "1001"],
    "invitation": "0001/2013",
    "result_minutes": "0002/2013",
    "verbatim_minutes": "0003/2013",
    "attachments": [
        "0004/2013",
        "0005/2013"
    ],
    "last_modified": "2012-01-08T14:05:27+01:00"
}
```


5.8 oparl:AgendaItem (Tagesordnungspunkt)

Der Tagesordnungspunkt wird für eine bestimmte Sitzung angelegt, erhält eine (innerhalb dieser Sitzung eindeutige) Nummer und einen Titel (Betreff). Nach der Sitzung wird dem Tagesordnungspunkt außerdem ein Ergebnis angehängt. Unter Umständen kann dem Tagesordnungspunkt ein bestimmter Beschlusstext beigefügt sein.

Überlicherweise haben Sitzungen mehrere Tagesordnungspunkte.

5.8.1 Eigenschaften

Nummer (`identifier`) Beispiel: “1.2.3”. Diese Nummer gibt an, in welcher Reihenfolge die Tagesordnungspunkte einer Sitzung normalerweise behandelt werden. Im Kontext einer Sitzung ist diese Nummer eindeutig.

Öffentlich (`public`) Kennzeichnet, ob der Tagesordnungspunkt in öffentlicher Sitzung behandelt wird. Kann die Werte `true` (öffentlich) oder `false` annehmen.

Titel (`title`) Das Thema des Tagesordnungspunktes

Ergebnis (`result`) *Optional.* Kategorische Information darüber, welches Ergebnis die Beratung des Tagesordnungspunktes gebracht 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.

Ergebnis Details (`result_details`) *Optional.* Ermöglicht die Angabe zusätzlicher Textinformationen zum Ergebnis, zum Beispiel im Fall der Verweisung an ein anderes Gremium die Angabe, an welches Gremium verwiesen wurde.

Beschlusstext (`resolution_text`) *Optional.* 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.

Zuletzt geändert (`last_modified`) Datum und Uhrzeit der letzten Änderung

Anmerkungen

- Einige Systeme vergeben zu Tagesordnungspunkten intern unveränderliche, numerische IDs. Es ist unklar, ob es zusätzlichen Nutzen bringt, derartige IDs, neben den Nummern, in den Standard zu übernehmen. Dies würde vermutlich nur Sinn ergeben, wenn es als Pflichtfeld gelten könnte.
- Teil der Beratungen über einheitliche Nomenklatur im Standard sollte sein, eine Vereinheitlichung der Werte für die Eigenschaft `result` zu diskutieren.

5.8.2 Beziehungen

- Jeder Tagesordnungspunkt gehört zu genau einem `oparl:Meeting`.
- Der Tagesordnungspunkt kann auf eine Drucksache verweisen, die im Rahmen dieses Tagesordnungspunkt beraten werden soll.
- Es können `oparl:Person` Objekte referenziert werden, die während der Abstimmung zu diesem Tagesordnungspunkt *nicht* anwesend waren.

5.8.3 Beispiel


```
{
    "meeting": "3271",
    "identifier": "3.1.2",
    "public": true,
    "title": "Gemeinschaftsgrundschule Hornschaftsstraße/Höhenhaus. Hier: Anfrage von Herrn Phi
    "result": "Geändert beschlossen",
    "resolution_text": "Der Beschluss weicht wie folgt vom Antrag ab: ...",
    "people_absent": ["1002", "1003"],
    "last_modified": "2012-08-16T14:05:27+02:00"
}
```


5.9 oparl:Paper (Drucksache)

Eine Drucksache bildet Mitteilungen, Antworten auf Anfragen, Beschlussvorlagen, Anfragen, Anträge und weitere Vorlagen ab. Jede Drucksache erhält eine eindeutige Kennung.

Die Drucksache hat im Informationsmodell eine hervorgehobene Bedeutung. Im Fall eines Antrags kann mit einer einzigen Drucksache ein über Monate oder Jahre dauernder politischer Entscheidungsprozess verbunden sein. In dem Zusammenhang entstehen üblicherweise weitere Drucksachen.

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

Jede Drucksache ist über die Eigenschaft “Typ” als eine der folgenden Arten von Drucksachen gekennzeichnet:

- **Beschlussvorlage**: Entscheidungsvorschlag der Verwaltung
- **Antrag**: Entscheidungsvorschlag einer Fraktionen bzw. mehrerer Fraktionen oder einer/mehrerer Einzelperson/en
- **Anfrage**: Frage(n) einer oder mehrerer Fraktion oder Einzelpersonen an die Verwaltung
- **Mitteilung/Stellungnahme der Verwaltung**: Eine Information der Verwaltung an einzelne oder mehrere Gremien. Darunter fallen nicht Beantwortungen von Anfragen.
- **Beantwortung einer Anfrage**: Antwort der Verwaltung auf (mündliche oder schriftliche) Anfragen

5.9.1 Eigenschaften

Schlüssel (`id`) Die Kennung einer Drucksache muss für die jeweilige Körperschaft eindeutig sein. Sie kann sowohl Ziffern als auch Buchstaben enthalten. Einige Systeme (z.B. Köln) verwenden besondere Trennzeichen wie "/", um eine Jahreszahl von einer laufenden Nummer abzutrennen. Weiterhin werden mancherorts führende Nullen verwendet.

Datum (`date`) Datum der Veröffentlichung

Typ (`type`) Art der Drucksache (Erläuterung siehe oben)

Zuletzt geändert (`last_modified`) Datum und Uhrzeit der letzten Änderung

5.9.2 Beziehungen

- Es muss genau ein **Hauptdokument** (`oparl:Document`) referenziert werden.
- Es können beliebig viele weitere Dokumente referenziert werden, die als nachgeordnete **Anlagen** zur Drucksache verstanden werden.
- Die Drucksache ist beliebig vielen Gremien zuzuordnen, in denen diese beraten wird.
- Drucksachen können **Urhebern** zugewiesen werden. Im Fall von Mitteilungen der Verwaltung ist dies oft der Oberbürgermeister. Bei Anträgen oder Anfragen können Organisationen oder Einzelpersonen referenziert werden. Es können stets mehrere Uhrheber verknüpft werden.
- Es können beliebig viele **Orte** (siehe Objekttyp "Ort") referenziert werden, die im Inhalt der Drucksache behandelt werden. Beispiel: Beschlussvorlage zur Freigabe von Mitteln für die Sanierung eines Sportplatzes, wobei der Ort die Lage des Sportplatzes genau beschreibt. (TODO)
- Drucksachen können auf andere Drucksachen referenzieren. Diese Verweise können verschiedene semantische Beziehungen ausdrücken. So kann eine Drucksache auf eine übergeordnete oder eine oder mehrere untergeordnete Drucksachen verweisen. Beim Drucksachen-Typ "Beantwortung einer Anfrage" ist die Drucksache zu referenzieren, die die ursprüngliche **Anfrage** beinhaltet. Denkbar sind auch Verweise auf frühere Drucksachen zum selben Thema. Zu klären ist, wie die verschiedenen möglichen Beziehungen formell ausgedrückt werden.
- Drucksachen können zu beliebig vielen Tagesordnungspunkten in Beziehung stehen, um die **Beratungsfolge** einer Drucksache abzubilden. Hierbei kann die Beziehung jeweils mit einer Zuständigkeit versehen sein, die noch näher zu bestimmen ist (TODO).

5.9.3 Beispiel

```
{
    "id": "1234/2012",
    "date": "2013-01-04",
    "type": "Beantwortung einer Anfrage",
    "related_papers": [
        "0768/2012"
    ],
    "main_document": "3000.pdf",
    "attachments": [
        "3002.pdf",
        "3003.pdf"
    ],
    "locations": [
        {
```

```
            "description": "Theodor-Heuss-Ring 1",
            "lat": 7.148,
            "lon": 50.023
        }
    ],
    "committees": ["STA"],
    "creators": [
        {
            "typ": "Organisation",
            "id": "2000"
        },
        {
            "typ": "Person",
            "id": "1000"
        }
    ],
    "consultations": [
        {
            "meeting": "3271",
            "agendaitem": "3.1.2",
            "role": "Federführende Beratung"
        }
    ],
    "last_modified": "2013-01-08T12:05:27+01:00"
}
```


5.10 oparl:Document (Dokument)

Ein Dokument hält Metadaten einer Datei vor, beispielsweise einer PDF-Datei, eines RTF- oder ODF-Dokuments (oder auch einer Datei in einem proprietären Format).

Wird von einem Dokument in einem Nicht-PDF-Format (z.B. RTF oder ODF) eine PDF-Ableitung hinterlegt, ist diese Ableitung ebenfalls ein Dokument. Um zu zeigen, dass es sich um eine Ableitung handelt, verweist dieses auf das Original als "Master".

Im Unterschied zur Drucksache benötigt das Dokument keine nutzerfreundliche Kennung.

5.10.1 Eigenschaften

Schlüssel (`id`) Unveränderliche Kennung

Name (`name`) Dateiname, z.B. "12345.pdf"

Dateityp (`mime_type`) Mime-Typ des Inhalts, z.B. "application/pdf"

Veröffentlichungsdatum (`date`) Datum des Tages, an dem das Dokument ins System eingestellt wurde

Änderungsdatum und -uhrzeit (`last_modified`) Datum und Uhrzeit der letzten Änderung des Dokuments

Prüfsumme (`sha1_checksum`) SHA1-Prüfsumme des Dokumenteninhalts

URL (`url`) URL zum Abruf der Daten dieses Dokuments mittels HTTP GET-Aufruf

Nur-Text-Version (`text`) Reine Text-Wiedergabe des Dokumenteninhalts, sofern es sich nicht um eine reine Abbildung handelt.

5.10.2 Beziehungen

- Dokumente gehören zwingend zu einer **Drucksache**, optional auch zu mehreren. Ein Dokument kann entweder als Hauptdokument einer Drucksache oder als Anlage eingestuft sein.
- Ein Dokument kann auf ein anderes Dokument referenzieren, wenn es von dem anderen Dokument abstammt. So ist es möglich, von einem abgeleiteten Dokument zu seinem Dokumenten-Master zu gelangen (Beispiel: von einem PDF-Dokument zum OpenOffice-Original).


```
{
    "id": "3000",
    "name": "3000.pdf",
    "mime_type": "application/pdf",
    "date": "2013-01-04T07:54:13+01:00",
    "last_modified": "2013-01-04T07:54:13+01:00",
    "sha1_checksum": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
    "url": "http://ris.beispielstadt.de/api/documents/3000.pdf",
    "text": "Der Übersichtsplan zeigt alle Ebenen des ...",
    "master": "2099"
}
```


5.11 oparl:Consultation (Beratung)

5.12 oparl:Location (Ort)

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

OParl sieht die Angabe von Geodaten in Anlehnung an die GeoJSON-Spezifikation [13] vor. Die GeoJSON-Spezifikation erlaubt die Abbildung von vielen unterschiedlichen Geometrien wie Punkten, Pfaden und Polygonen. Während GeoJSON zu jedem Geodaten-Objekt auch die Speicherung zusätzlicher Metadaten ermöglicht, beschränkt sich OParl ledliglich auf das `geometry`-Attribut in GeoJSON. Sämtliche Geo-Koordinatenangaben werden in in OParl im WGS-84-System [11] erwartet.

5.12.1 Eigenschaften

Textanabe (`description`) *Optional.* Textliche Beschreibung eines Orts, z.B. in Form einer Adresse

Koordinaten (`geometry`) *Optional.* GeoJSON geometry Objekt

Zuletzt geändert (`last_modified`) Datum und Uhrzeit der letzten Änderung

5.12.2 Beziehungen

- Orte können mit Drucksachen in Verbindung stehen.


```
{
    "description": "Honschaftsstraße 312, 51061 Köln",
    "geometry": {
        "type": "Point",
        "coordinates": [7.03291, 50.98249]
    },
    "last_modified": "2013-02-14T14:05:27+01:00"
}
```

5.13 oparl:Contact (Kontakt)

6 Fußnoten

[11]: World Geodetic System 1984 (EPSG:4326), wird unter anderem auch vom Global Positioning System (GPS) verwendet.

[13]: GeoJSON www.geojson.org

[14]: Frankfurt Gestalten www.geojson.org

[15]: Offenes Köln offeneskoeln.de

[16]: OpenRuhr:RIS openruhr.de/openruhrris

7 Glossar

JSON-LD JSON for Linked Data

RIS Ratsinformationssystem

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