OParl Schnittstellen-Spezifikation (Entwurf für 1.0)

OParl Team - http://oparl.org/

Inhaltsverzeichnis

1 Einleitung **4**

1.1 Was ist OParl in Kürze? . . . 4
1.2 Zielsetzung von OParl . . . 4
1.3 Transparenz und Beteiligung durch Open Data . . . 5
1.4 Werdegang von OParl 1.0 . . . 6
1.5 Zukunft von OParl . . . 7
1.6 Nomenklatur der Spezifikation und Satzkonventionen . . . 9
1.6.1 Zwingende, empfohlene und optionale Anforderungen . . . 9
1.6.2 Geschlechterspezifische Begrifflichkeiten . . . 9
1.6.3 Codebeispiele . . . 10
1.6.4 Namespace-Präfixe für Objekt- und Datentypen . . . 10
1.7 Initiatoren . . . 11
1.8 Unterstützer . . . 11
1.9 Autoren . . . 11

2 Architektur **11**

2.1 Überblick . . . 12
2.2 Parlamentarisches Informationssystem . . . 12
2.3 Server . . . 12
2.4 API . . . 13
2.5 Client . . . 13
2.6 Cache . . . 13
2.7 Nutzerin oder Nutzer . . . 13
2.8 Objekt . . . 13

3 Nutzungsszenarien **13**

3.1 Szenario 1: Mobile Client-Anwendung . . . 14
3.2 Szenario 2: Integration in Web-Portal . . . 15
3.3 Szenario 3: Meta-Suche . . . 16
3.4 Szenario 4: Forschungsprojekt Themen- und Sprachanalyse . . . 16

4 Prinzipien und Funktionen des Schnittstelle **17**

4.1 Designprinzipien . . . 17
4.1.1 Aufbauen auf gängiger Praxis . . . 17
4.1.2 Verbesserung gegenüber dem Status Quo wo möglich . . . 17
4.1.3 RESTful . . . 18
4.1.4 Selbstbeschreibungsfähigkeit . . . 18
4.1.5 Erweiterbarkeit . . . 18
4.1.6 Browseability/Verlinkung . . . 18
4.1.7 Schritte in Richtung Linked Data . . . 19
4.2 Zukunftssicherheit . . . 19
4.3 HTTP und HTTPS . . . 20
4.4 URLs, IRIs und URIs . . . 21
4.4.1 URL-Kanonisierung . . . 22
4.4.2 Langlebigkeit . . . 23
4.5 Serialisierung mittels JSON und JSONP . . . 24
4.5.1 JSON . . . 24
4.5.2 JSONP . . . 24
4.6 Objektlisten . . . 25
4.6.1 Interne und externe Ausgabe von Listen . . . 26
4.6.2 Kompakte und vollständige Form (`listformat`) . . . 27
4.6.3 Paginierung . . . 28
4.6.4 Sortierung . . . 31
4.6.5 Filter . . . 31
4.7 Feeds . . . 32
4.7.1 Der Feed “Neue Objekte” . . . 32
4.7.2 Der Feed “Geänderte Objekte” . . . 33
4.7.3 Der Feed “Entfernte Objekte” . . . 34
4.8 Dateizugriff . . . 34
4.8.1 GET und HEAD Anfragen . . . 34
4.8.2 Allgemeiner Zugriff und expliziter Download . . . 35
4.8.3 Obligatorische und empfohlene Header . . . 35
4.8.4 Conditional GET . . . 35
4.8.5 Zustandsloser Dateizugriff . . . 36
4.8.6 Weiterleitungen . . . 36
4.8.7 Entfernte Dateien . . . 36
4.9 Content Negotiation . . . 36
4.10 HTTP-Kompression . . . 37
4.11 Ausnahmebehandlung . . . 37
4.12 Liste reservierter URL-Parameter . . . 38

5 Schema **38**

5.1 Übergreifende Aspekte . . . 39
5.1.1 Unicode-Zeichenketten als Standard . . . 39
5.1.2 `null`-Werte und “leere” Werte . . . 39
5.1.3 Kardinalität . . . 39
5.1.4 Datums- und Zeitangaben . . . 39
5.1.5 Vokabulare zur Klassifizierung . . . 40
5.1.6 Herstellerspezifische Erweiterungen . . . 41
5.1.7 URL-Pfade in den Beispielen . . . 41
5.2 Eigenschaften mit Verwendung in mehreren Objekttypen . . . 41
5.2.1 `id` . . . 41
5.2.2 `type` . . . 42
5.2.3 `name` und `shortName` . . . 42
5.2.4 `license` . . . 42
5.2.5 `created` . . . 43
5.2.6 `modified` . . . 43
5.2.7 `keyword` . . . 43
5.3 oparl:System (System) . . . 43
5.3.1 Eigenschaften . . . 44
5.4 oparl:Body (Körperschaft) . . . 44
5.4.1 Eigenschaften . . . 45
5.5 oparl:Organization (Gruppierung) . . . 46
5.5.1 Eigenschaften . . . 47
5.6 oparl:Person (Person) . . . 48
5.6.1 Eigenschaften . . . 49
5.7 oparl:Meeting (Sitzung) . . . 50
5.7.1 Eigenschaften . . . 50
5.8 oparl:AgendaItem (Tagesordnungspunkt) . . . 52
5.8.1 Eigenschaften . . . 52
5.9 oparl:Paper (Drucksache) . . . 53
5.9.1 Eigenschaften . . . 54
5.10 oparl:File (Datei) . . . 54
5.10.1 Eigenschaften . . . 55
5.11 oparl:Consultation (Beratung) . . . 56
5.11.1 Eigenschaften . . . 57
5.12 oparl:Location (Ort) . . . 57
5.12.1 Eigenschaften . . . 58
5.13 oparl:Membership . . . 58
5.13.1 Eigenschaften . . . 59

5.14 oparl:LegislativeTerm 59
5.14.1 Eigenschaften 60

1 Einleitung

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

1.1 Was ist OParl in Kürze?

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

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

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

1.2 Zielsetzung von OParl

OParl richtet sich an verschiedene Nutzergruppen und Stakeholder:

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

Die Gründe, warum Betreiber von parlamentarischen Informationssystemen den Zugriff darauf über eine standardisierte Schnittstelle ermöglichen sollten oder möchten, können vielfältig sein und je nach Nutzergruppe unterschiedlich.

Ein zentrales Argument für Verwaltung und politische Gremieni, sei es in Gebietskörperschaften oder auf Landes- oder Bundesebene, 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

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

möglich anzubieten, dass jegliche weitere Verwendung durch Dritte technisch möglich ist. Der seit einiger Zeit etablierte Begriff für dieses Prinzip heißt “Open Data”.

Open-Data-Initiativen können unter Rückgriff auf RIS mit OParl-Schnittstelle einfacher Dokumente und Daten aus unterschiedlichen Gebietskörperschaften in Open-Data-Katalogen verzeichnen und so einfacher auffindbar machen für die Weiterverwendung durch Dritte.

Bürgerinnen und Bürger, politische Parteien und zivilgesellschaftliche Organisationen können einfacher auf Inhalte parlamentarischer Informationssysteme zugreifen und diese entsprechend ihren Interessen aufbereiten. Dies können beispielsweise Visualisierungen von enthaltenen Daten, die Anreicherung von Informationsangeboten für spezielle Nutzergruppen oder die Schaffung von Benutzeroberflächen mit besonderen Funktionen für verschiedene Endgeräte sein.

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

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

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

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

Für Wissenschaftler, die z. B. an vergleichenden Untersuchungen zu Vorgängen in verschiedenen Gebietskörperschaften interessiert sind, ergeben sich ebenso vielfältige Möglichkeiten über mehrere RIS-Instanzen hinweg auf entsprechende Informationen zuzugreifen und diese so einfacher in ihre Analysen einzubeziehen.

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.

Anbieter von Server- und Softwareprodukten, die RIS-Lösungen anbieten, können mit der Implementation der OParl-Schnittstelle ihren Kunden eine entsprechende einheitliche Schnittstelle anbieten. Für Anbieter von Linked-Data-Plattformen ergeben sich u. a. Möglichkeiten zur vereinfachten Zusammenführung und Anreicherung von Inhalten parlamentarischer Informationssysteme.

Ausführlichere Beschreibungen einiger möglicher Anwendungsszenarien finden sich im Kapitel Nutzungsszenarien.

1.3 Transparenz und Beteiligung durch Open Data

Öffentliche Stellen verfügen über vielfältige Informationen und Daten. Seit einigen Jahren sind zivilgesellschaftliche Organisationen sowie Politik und Verwaltung unter dem Schlagwort

[2] Frankfurt Gestalten: http://www.frankfurt-gestalten.de/
[3] Offenes Köln: http://offeneskoeln.de/
[4] OpenRuhr:RIS: http://openruhr.de/openruhrris/

Open Data international und auch in Deutschland in unterschiedlichem Maße um eine stärkere Öffnung dieser Daten bemüht[5]. Bei dem Ansatz Open Data[6] geht es darum, diese Daten so bereitzustellen, dass Dritte diese einfacher finden und weiterverwenden können.

Die zehn Open-Data-Prinzipien der Sunlight-Foundation[7] beschreiben die Offenheit von Datensätzen. Wesentlich dabei sind vor allem die einfache rechtliche und die technische Offenheit. Bei ersterer geht es darum, dass Datensätze unter Nutzungsbestimmungen bereitgestellt werden, die kurz und verständlich formuliert sind und mindestens jegliche weitere Verwendung inklusive der kommerziellen erlauben, unter der Voraussetzung, dass bei der Weiterverwendung die Quelle benannt wird. Bei der technischen Offenheit steht die Bereitstellung von Datensätzen in möglichst maschinenlesbaren Formaten im Vordergrund. Dies bedeutet, stärker strukturierte Datensätze sind in der Bereitstellung zu bevorzugen. Liegen Daten innerhalb einer Organisation in einer Datenbank vor, so bietet es sich an, diese soweit möglich über eine Programmierschnittstelle (API) für Außenstehende bereitzustellen.

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

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

1.4 Werdegang von OParl 1.0

17. und 18. November 2012: Vorüberlegungen für einen offenen Standards für parlamentarische Informationssysteme beginnen auf der Veranstaltung "Stadt Land Code" der Open Knowledge Foundation (OKF) Deutschland. Marianne Wulff (VITAKO) und Marian Steinbach tauschen sich über Möglichkeiten, dies gemeinsam voran zu treiben, aus.

6. Dezember 2012: Im Rahmen einer Anhörung des Landtags von Nordrhein-Westfalen zum Thema Open Government und Open Data sind sowohl Jens Klessmann (Fraunhofer

[5] Eine weltweite Übersicht zu Open-Data-Projekten bietet z. B. der Open-Data-Showroom http://opendata-showroom.org/de/

[6] vgl. https://de.wikipedia.org/wiki/Open_data

[7] Ten Principles for Opening Up Open Government Information, https://sunlightfoundation.com/policy/documents/ten-open-data-principles

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

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

FOKUS) als auch Marian Steinbach als Sachverständige eingeladen. Am Rande der Veranstaltung beschließen sie, die Bemühungen um eine Standardisierung offener Ratsinformationen gemeinsam mit Unterstützung von VITAKO voranzutreiben. Im selben Monat beginnen Marianne Wulff, Jens Klessmann und Marian Steinbach mit der Planung eines initialen Workshops mit Vertreterinnen und Vertretern von Kommunen, kommunalen IT-Dienstleistern, RIS-Anbietern und Zivilgesellschaft. Ziel: Die Bereitschaft zur Zusammenarbeit an einem gemeinsamen Standard ermitteln. Die Arbeite an einem Entwurf für die vorliegene Spezifikation beginnt. Der Entwurf wird von Beginn an öffentlich auf GitHub.com bereit gestellt.

17. April 2013: Insgesamt 30 Teilnehmer versammeln sich zum ersten Workshop in Köln, um sich 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 Name für die Bemühungen der Gruppe stehen soll.

Mai 2013 bis Januar 2014: Die verteilte Arbeit am Standard-Entwurf läuft nach dem Workshop in Köln nur schleppend; der ursprünglich gesetzte Termin kann nicht gehalten werden. Für den 22. Januar 2014 laden die Initiatoren zu einem eintägigen OParl-Workshop in Bielefeld ein, um die Spezifikation in intensiver Zusammenarbeit vor Ort so weit wie möglich voran zu treiben und eine baldige Fertigstellung zu ermöglichen.

26. Januar 2014: In Düsseldorf findet ein weiterer technischer Workshop zur Arbeit an der Spezifikation statt.

27. März 2014: Die Initiatoren kommunizieren einen neuen Zeiplan für die Fertigstellung von Version 1.0.[9]. Dieser sieht eine Verabschiedung in der 23. KW (2. bis 6 Juni) vor. Zuvor soll in einer mehrwöchigen Review-Phase jedem die Möglichkeit offen stehen, den Stand der Spezifikation ausführlich zu kommentieren.

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

12. Mai 2014: An der Spezifikation wird weiterhin aktiv gearbeitet und ein kurzfristiger Redaktionsschluss nicht absehbar. Die Initiatoren geben einen neuen Zeitplan bekannt[10]. Hier wird als Termin für die Veröffentlichung der 23. Juni anvisiert.

Juni 2014: Aufgrund von Feedback aus der Review-Phase wird die Spezifikation deutlich überarbeitet.

TODO: Fortschreiben bis zur tatssächlichen Fertigstellung.

1.5 Zukunft von OParl

Die vorliegende Version 1.0 der OParl-Spezifikation erhebt keineswegs den Anspruch, ein aktuell und für die ferne Zukunft vollständige Lösung alle Problemstellungen rund um die Veröffentlichung parlamentarischer Informationen zu sein. Viele Funktionen, die denkbar und bestimmt sinnvoll wären, sind aus verschiedensten Gründen in dieser Version noch nicht berücksichtigt. Einige der Gründe, die dazu führten, ein Thema nicht auszuspezifizieren, waren:

- Zu wenig detaillierte Anforderungen aus der Praxis
- Zu großer Arbeitsaufwand für die Spezifikations-Entwicklung
- Hohe Aufwände bei den Server-Implementierern

Zu den Themen, die in zukünftigen Versionen adressiert werden können, zählen:

[9] http://oparl.org/news/2014/03/27/neuer-zeitplan-fuer-die-spezifikation/
[10] http://oparl.org/news/2014/05/12/verschiebung-beim-zeitplan/

- Loslösung von der kommunalen Ebene: Es ist möglich, dass OParl mit nur geringfügigen Änderungen oder Erweiterungen auch für die Ebene von Bundesländern (Landtage) oder des Bundes (Bundestag, Bundesrat) nutzbar wäre.

- Flexible Abfragemöglichkeiten für Objekte: Aufgrund der unklaren Anforderungslage sowie dem Bestreben möglichst wenige spezielle Lösungen nur für OParl zu schaffen sind in Version 1.0 nur sehr beschränkte Möglichkeiten vorgesehen, Listen von Drucksachen etc. nach bestimmten Kriterien einzuschränken. Zukünftig könnten hier weitere Möglichkeiten definiert werden, bis hin zur Suche nach Stichworten in Volltexten. Ein möglicher Ansatz hierfür wäre die Verwendung von Linked Data Fragements[11]. Diese ermöglichen Clients mächtige Abfragen, ohne dabei zu übermäßiger Last auf Serverseite zu führen.

- Detaillierte Wiedergabe von Abstimmungen: Das Thema ist vom Datenmodell/Schema der vorliegenden Version noch nicht abgedeckt, da es vielerorts nicht üblich ist, Abstimmungen über die Fraktionsebene genau zu erfassen. Zukünftig könnte es ein Ziel sein, das Abstimmungsverhalten einzelner Parlamentarier und Fraktionen genau zu dokumentieren.

- Strukturierte Protokolle: Während Protokolle in der Praxis in der Regel als unstrukturierte Fließtexte angelegt werden, könnte eine Strukturierung der Inhalte die Nachvollziehbarkeit des parlamentarischen Geschehens deutlich verbessern.

- Vokabular für Drucksachentypen: In der Praxis wird eine Vielzahl von Drucksachentypen genutzt. Um eine Vergleichbarkeit, beispielsweise zwischen Anträgen, innerhalb der Parlamente zu schaffen, könnte zukünftig eine Erweiterung des OParl-Vokabulars im Sinne von Linked Data angestrebt werden.[12]

- Weitere externe Standards, insbesondere zu Paginierung: Teile der Spezifikation beziehen sich auf technische Anforderungen die nicht nur für OParl relevant sind. Das betrifft insbesondere die Paginierung-Mechanismen. Idealerweise sollte OParl hierfür externe Standards verwenden. Deshalb werden entsprechende Standardisierungsvorhaben wie Linked Data Platform Paging[13] des W3C und das Hydra Core Vocabulary[14] beobachtet.

- Mehrsprachigkeit: Die Unterstützung von mehrsprachigen Inhalten.

- Schreibender Zugriff: Denkbar ist auch, dass OParl von der derzeitigen Ausrichtung auf den reinen lesenden Informationszugriff um die Möglichkeit, Inhalte anzulegen, zu verändern und zu entfernen sowie um das Konzept von authentifizierten Nutzern erweitert wird.

- Internationalisierung: Es gibt in sehr vielen Ländern Gebietskörperschaften mit politischen Gremien, deren Prozesse ähnlich strukturiert sind, wie diejenigen in Deutschland. Auch dort besteht Bedarf an standardisierten Vokabularen zur Veröffentlichung parlamentarischer Informationen. Deshalb sind – teilweise noch vor OParl – auch weitere entsprechende Initiativen entstanden.[15] Eine Zusammenarbeit mit derartigen Initiativen mit dem Ziel der Wiederverwendung von Arbeitsergebnissen ist vorstellbar.

- IT-Planungsrat: Dieser kann die Verbindlichkeit von Standards wie OParl für Deutschland beschliessen. Der "Vertrag über die Errichtung des IT-Planungsrats und über die Grundlagen der Zusammenarbeit beim Einsatz der Informationstechnologie in den Verwaltungen von Bund und Ländern – Vertrag zur Ausführung von Artikel 91c GG"[16] enthält in § 3 Absatz 2 diese Aussage: "Beschlüsse über Standards im Sinne des Absatz 1 werden vom IT-Planungsrat ... gefasst, soweit dies zum bund-länderübergreifenden Datenaustausch oder zur Vereinheitlichung des Datenaustauschs der öffentlichen Verwaltung mit Bürgern und Wirtschaft notwendig ist."

[11] http://linkeddatafragments.org/
[12] Das gesagte lässt sich auch auf viele andere Informationen, nicht nur auf Drucksachen, anwenden.
[13] https://dvcs.w3.org/hg/ldpwg/raw-file/default/ldp-paging.html
[14] http://hydra-cg.com/spec/latest/core/#collections
[15] Vgl. dazu beispielsweise http://popoloproject.com/, TODO: UK, KB Niederlande, Italienisches Parlament: http://data.camera.it/data/en/
[16] http://www.it-planungsrat.de/SharedDocs/Downloads/DE/ITPlanungsrat/Staatsvertrag/Staatsvertrag.html

Generell gilt auch für OParl: “Completion is a state that a good specification never reaches before it’s irrelevant.”[17] - Ian Hickson

1.6 Nomenklatur der Spezifikation und Satzkonventionen

1.6.1 Zwingende, empfohlene und optionale Anforderungen

Dieses Spezifikationsdokument nutzt die Modalverben *müssen*, *können* und *sollen* in einer Art und Weise, die bestimmte Anforderungen 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.[18]

Die Bedeutung im Einzelnen:

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

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

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

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

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

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

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

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

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

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

1.6.2 Geschlechterspezifische Begrifflichkeiten

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

[17] OGCWG - Lessons Learned: http://www.w3.org/community/oilgaschem/wiki/OGCWG_-_Lessons_Learned

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

1.6.3 Codebeispiele

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

Codebeispiele erheben insbesondere bei JSON-Code nicht den Anspruch auf hundertprozentige syntaktische Korrektheit. Insbesondere können in Codebeispielen Auslassungen vorkommen, die mit `...` gekennzeichnet werden. Darüber hinaus werden zugunsten der einfacheren Lesbarkeit Umlaute verwendet, obwohl OParl grundsätzlichlich die Verwendung von Unicode-Zeichneketten vorsieht.

1.6.4 Namespace-Präfixe für Objekt- und Datentypen

Bei der Erwähnung von Objekttypen, die in dieser Spezifikation beschrieben werden, wird in der Regel ein Präfix `oparl:` vor den Namen gesetzt, z. B. "oparl:Organization". Damit soll verdeutlicht werden, dass dieser Objekttyp innerhalb der OParl-Spezifikation beschrieben wird.

Das Präfix `oparl:` steht hierbei für die folgende Namespace-URL:

```
http://oparl.org/schema/1.0/
```

Dadurch kann eine Typenangabe wie `oparl:Organization` eindeutig in die folgende URL übersetzt werden:

```
http://oparl.org/schema/1.0/Organization
```

In einigen Fällen nutzt OParl Objekttypen oder Datentypen, die in anderen Spezifikationen beschrieben wurden, um so von bereits getaner Arbeit und etablierten Standards zu profitieren. Ein Beispiel dafür ist das Datumsformat, das in der XML-Schema-Spezifikation beschrieben wird. Der Namespace von XML-Schema hat die URL

```
http://www.w3.org/2001/XMLSchema
```

und wird im vorliegenden Dokument mit dem Präfix `xsd:` abgekürzt. Somit ist, wenn beispielsweise von der Eigenschaft `xsd:date` die Rede ist, tatsächlich diese Namespace-URL gemeint:

```
http://www.w3.org/2001/XMLSchema#date
```

Nachfolgend eine Aufstellung der weiteren in diesem Dokument verwendeten Namespace-Präfixe.

Präfix	Beschreibung	Namespace-URL
`xsd`	XML Schema	http://www.w3.org/2001/XMLSchema#
`foaf`	Friend of a Friend	http://xmlns.com/foaf/0.1/
`skos`	Simple Knowledge Organization System	http://www.w3.org/2004/02/skos/core#
`vcard`	vCard	http://www.w3.org/2006/vcard/ns#

1.7 Initiatoren

OParl wurde initiiert von Marian Steinbach[19], Jens Klessmann[20], Marianne Wulff und Christine Siegfried[21].

1.8 Unterstützer

Die folgenden Organisationen und Unternehmen zählen zu den Unterstützern von OParl:

Organisation/Firma	Kategorie
CC e-Gov GmbH	RIS-Hersteller
Citeq (Münster)	Kommunale Dienstleister
ITDZ Berlin	Kommunale Dienstleister
Kiru (Ulm)	Kommunale Dienstleister
KDVZ Rhein-Erft-Rur	Kommunale Dienstleister
KRZN	Kommunale Dienstleister
Open Knowledge Foundation e. V.	Initiativen
OpenRuhr	Initiativen
Parlamentwatch e. V.	Initiativen
Piratenpartei	Initiativen
PROVOX Systemplanung GmbH	RIS-Hersteller
QuinScape GmbH	RIS-Hersteller
regioIT (Aachen)	Kommunale Dienstleister
Somacos GmbH und Co. KG	RIS-Hersteller
Stadt Bonn	Kommune
Stadt Köln	Kommune
Stadt Moers	Kommune
Sternberg Software-Technik GmbH	RIS-Hersteller
Wikimedia Deutschland	Initiativen

1.9 Autoren

An diesem Dokument haben mitgewirkt:

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

2 Architektur

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

[19] http://www.sendung.de/
[20] http://www.fokus.fraunhofer.de/
[21] beide Vitako http://www.vitako.de/

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.[22]

2.1 Überblick

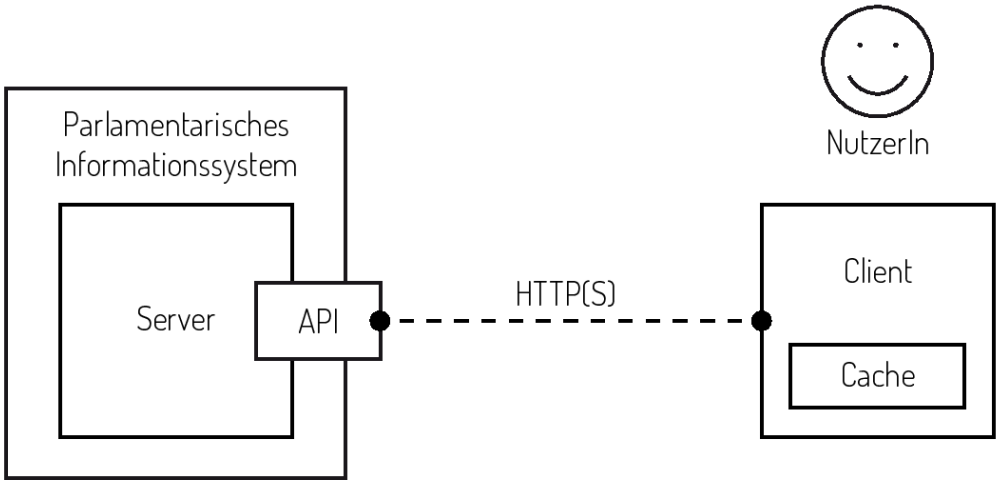

Abbildung 1: Architekturdiagramm

2.2 Parlamentarisches Informationssystem

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

Diese Systeme unterstützen in der Regel mehrere der folgenden Funktionen:

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

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

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

2.3 Server

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

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

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 die verschlüsselte Nutzung der API möglich, sofern Server dies unterstützen.

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 geschont 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 beschrieben 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.

Überblick der Szenarien:

1. Mobile Client-Anwendung
2. Integration in Web-Portal
3. Meta-Suche
4. Forschungsprojekt Themen- und Sprachanalyse

3.1 Szenario 1: Mobile Client-Anwendung

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

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

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

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

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

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

Für die Nutzer einer mobilen Client-Anwendung könnte es sich als besonders hilfreich erweisen, wenn Dokumente auf dem Server in verschiedenen Formaten zur Verfügung gestellt werden. Denn nicht jedes Endgerät mit kleinem Bildschirm 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 Volltextsuche 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 z. B. nach Distanz zum Objekt sortiert werden kann.

3.2 Szenario 2: Integration in Web-Portal

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

Ein Beispiel für die Realisierung eines solchen Integrations-Ansatzes wäre eine Kommune, die für ihre allgemeine Website eine Portallösung einsetzt und hier auch Inhalte aus dem kommunalen Ratsinformationssystem einspeisen und darstellen möchte. Die Inhalte könnten als Module mit anderen Inhalten, beispielsweise aus einem Web Content Management System (WCMS), gemeinsam auf einer Seite dargestellt werden.

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

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

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

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

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

3.3 Szenario 3: Meta-Suche

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

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

Dieses Modell ist grundsätzlich nicht auf einzelne OParl-Server oder einzelne Körperschaften beschränkt. Vielmehr könnte der Betreiber einer solchen Suchmaschine sich entschließen, die Informationen aus mehreren OParl-konformen Systemen zu indexieren. Nutzern könnte entweder angeboten werden, die Suche auf bestimmte Körperschaften, beispielsweise auf eine bestimmte Kommune, zu beschränken, oder ohne Beschränkung über alle angebotenen Körperschaften zu suchen.[23]

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

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

3.4 Szenario 4: Forschungsprojekt Themen- und Sprachanalyse

In einem Forschungsprojekt sollen Pro- und Contra-Argumentationen bei Ratsdiskussionen zum Ausbau von Stromtrassen identifiziert werden. Über die Analyse in mehreren Gebietskör-

[23] Daher der Begriff Meta-Suche

perschaften sollen die gefundenen Argumentationen zu wiederkehrenden Mustern verdichtet werden und festgestellt werden, wie diese Muster regional abweichen.

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

4 Prinzipien und Funktionen des Schnittstelle

In diesem Kapitel werden grundlegende Funktionsprinzipien einer OParl-Schnittstelle beschrieben.

4.1 Designprinzipien

4.1.1 Aufbauen auf gängiger Praxis

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

4.1.2 Verbesserung gegenüber dem Status Quo wo möglich

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

Ein Beispiel für eine derartige Funktion ist die Abbildung von Geodaten im Kontext von Drucksachen (`oparl:Paper`), um beispielsweise die Lage eines Bauvorhabens, das in einer Beschlussvorlage behandelt wird, zu beschreiben. Zwar ist den Autoren nur ein einziges parlamentarisches Informationssystem[24] in Deutschland bekannt, das Geoinformationen – und zwar in Form von Punktdaten, also einer Kombination aus Längen- und Breitengradangaben – mit Dokumenten verknüpft. Der Vorteil dieser Funktion ist jedoch anhand zahlreicher Anwendungsszenarien belegbar. Somit ist der vorliegenden OParl-Spezifikation die Möglichkeit beschrieben, Geodaten-Objekte einzubetten.

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

Auch die Ausgabe einer Nur-Text-Version im Kontext des Dokuments (`oparl:File`), 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.

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

4.1.3 RESTful

Die Bezeichnung "REST" (für "Representational State Transfer") wurde im Jahr 2000 von Roy Fielding eingeführt[25]. Die Definition von Fielding reicht sehr weit und berührt viele Details. In der Praxis wird der Begriff häufig genutzt, um eine Schnittstelle zu beschreiben,

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

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

4.1.4 Selbstbeschreibungsfähigkeit

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

Aber auch für die maschinelle Verarbeitung der Daten durch Clients kann die Selbstbeschreibung wichtig sein. Dies stellt allerdings erhöhte Anforderungen an die verwendeten Datenformate, da dafür formalisierte semantische Informationen enthalten sein müssen.

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 Abwärtskompatibilität, 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.

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 in Form von Graphen 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

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

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"[26] genannt.

4.1.7 Schritte in Richtung Linked Data

Der Begriff "Linked Data" steht für die Beschreibung von Daten in einer Form, die diese über ihren ursprünglichen Kontext hinaus verständlich macht.[27]

OParl unterstützt mit der vorliegenden Version 1.0 der Spezifikation die Anwendung einiger Linked-Data-Prinzipien. Damit soll die automatisierte Verarbeitung und Verknüpfung von Informationen aus parlamentarischen Informationnssystemen, auch über deren Grenzen hinweg, erleichtert werden.

Ein grundlegender Baustein der Linked-Data-Unterstützung in OParl ist die Tatsache, dass jedes Objekt durch eine URL identifiziert wird. So eignen sich OParl-Objekte für die Verknüpfung durch externe Anwendungen.

Ein weiteres wesentliches Linked-Data-Konzept in OParl ist die Möglichkeit, externe Vokabulare zur Klassifizierung von Objekten zu nutzen. So können beispielsweise Gruppierungen (d. h. Objekte des Typs `oparl:Organization`) als Ausschuss oder als Fraktion klassifiziert werden, wobei der Begriff "Ausschuss" oder "Fraktion" durch eine URL repräsentiert wird, die auf ein Konzept in einem externen Vokabular zeigt. Weitere Informationen hierzu sind in Vokabulare zur Klassifizierung beschrieben.

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.

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

- Die Kommune *Beispielstadt* betreibt ihren OParl-1.0-Server unter der URL `https://oparl.example.org/1.0/`.
- Verschiedene Clients, die für OParl Version 1.0 entwickelt wurden, kommen bei Nutzerinnen und Nutzern, die sich für den Stadtrat in Beispielstadt intressieren, zum Einsatz. Jeder Client-Nutzer hat dazu lediglich die URL `https://oparl.example.org/1.0/` des OParl-Servers in der Client-Konfiguration hinterlegt.

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

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

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

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

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

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

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

4.3 HTTP und HTTPS

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

Hierbei SOLL eine verschlüsselte Variante des Protokolls, HTTPS, zum Einsatz kommen. Dabei werden auch die URLs verschlüsselt, deren Kenntnis möglicherweise Rückschlüsse auf Interessen von Nutzer ermöglicht. 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 (vgl. 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 gilt für Anfragen an die entsprechende unverschlüsselte URL ZWINGEND:

- unter der URL ist kein Webserver erreichbar, oder
- der Server unter der URL beantwortet die Anfrage mit einer Weiterleitung an die HTTPS-URL (HTTP Status-Code 301)

Gleiches gilt umgekehrt: hat sich der Betreiber generell für den Betrieb des OParl-Servers über unverschlüsseltes HTTP entscheiden, dann MUSS für die entsprechenden HTTPS-URLs eine der beiden folgenden Möglichkeiten gelten:

- Entweder ist unter den entsprechenden HTTPS-URLs kein Webserver erreichbar
- oder Anfragen an die HTTPS-URLs werden mit Redirects auf die entsprechenden HTTP-URLs beantwortet (FRAGE: ist das ein sinnvolles Szenario?).

4.4 URLs, IRIs und URIs

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

Im Rahmen dieses Dokuments wird aus Gründen der Verständlichkeit generell der allgemein gebräuchliche Begriff *URL* werwendet, auch wenn damit tatsächlich die internationalisierte Variante nach RFC 3987[28], die korrekterweise *IRI* bzw. *Internationalized Resource Identifier* genannt werden müsste, gemeint ist. Einige Quellen wiederum nutzen den Begriff *URI* bzw. *Uniform Resource Identifier*. Das vorliegende Dokumente fasst alle drei Konzepte mit dem Begriff *URL* zusammen und ignoriert damit die Unterschiede der einzelnen Begriffe, da diese im Rahmen dieser Spezifikation nicht von Bedeutung sind.

Die grundsätzliche Funktionsweise von URLs ist in RFC 3986 beschrieben[29]. Darauf aufbauend sind hier die Bestandteile einer beispielhaften URL mit den Bezeichnungen beschriftet, mit denen sie in diesem Dokument benannt werden:

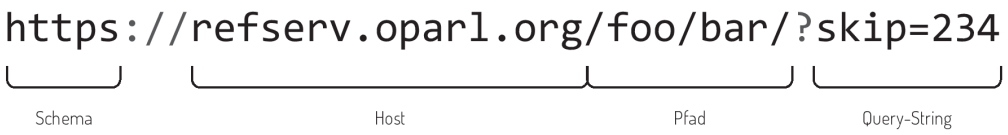

Abbildung 2: Benennung von URL-Bestandteilen

Der optionale *Query-String* besteht dabei aus beliebig vielen *Query-Parametern*, die jeweils einen Namen (links des Gleichheitszeichens) und einen Wert haben können.

[28] RFC 3987: http://tools.ietf.org/html/rfc3987
[29] RFC 3986: http://tools.ietf.org/html/rfc3986

4.4.1 URL-Kanonisierung

Absicht ist, dass jedes Objekt, das ein Server über eine OParl-API anbietet, über genau eine URL identifizierbar und abrufbar ist. Diese Vereinheitlichung der URL wird nachfolgend *Kanonisierung* genannt.

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.stadtrat.stadt-koeln.de/
```

kommuniziert beispielsweise direkt die Zugehörigkeit zur Stadt Köln als Betreiber des Systems. Die Bezeichnung “stadtrat” in der Subdomain zeigt den Zweck des Systems allgemein verständlich an. Der Host-Name “oparl.stadtrat.stadt-koeln.de” deutet an, dass diese URL zu einer OParl-Schnittstelle zu diesem System gehört. Eine technische Notwendigkeit zur Verwendung einer eigenen Domain für OParl besteht jedoch nicht, da JSON-Dokumente und HTML-Seiten mittels Content Negotiation über eine gemeinsame Domain ausgeliefert werden können.

Um die Kanonisierung zu gewährleisten, SOLLEN Betreiber alle Möglichkeiten ausschließen, die dazu führen können, dass eine Ressource neben der kanonischen URL noch über andere URLs abrufbar ist. Diese Faktoren können sein:

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

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

- https://83.123.89.102/
- https://oparl.stadtrat.stadtkoeln.de/

- https://risserv.stadt-koeln.de/

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

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

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

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

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

```
https://oparl.example.org/members?body=1&committee=2
https://oparl.example.org/members?committee=2&body=1
```

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

Von Clients wird erwartet, dass sie die URLs, die ihnen von Servern angeboten werden, unverändert verwenden. Clients SOLLEN NICHT versuchen, Schreibweisen von URLs zu ändern, Query-String-Parameter umzusortieren oder Ähnliches.

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[30] sowie die Europäische Kommission[31].

4.5 Serialisierung mittels JSON und JSONP

Eine OParl-konforme API gibt Objekte in Form von JSON aus. 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[32] 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 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 für JSONP[33]. Durch JSONP wird die JSON-Notation so erweitert, dass der ausgegebene Code ausführbarer JavaScript-Code wird. Damit wird erreicht,

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

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

[32] RFC4627: https://tools.ietf.org/html/rfc4627

[33] JSONP steht für "JSON with padding". Eine formelle Spezifikation existiert nicht. Der Wikipedia-Artikel http://en.wikipedia.org/wiki/JSONP fasst viele Quellen zusammen.

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-Unterstützung 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. (FRAGE: Sind Umlaute erlaubt?)

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 400 (*Bad Request*) Antwort bedienen.

4.6 Objektlisten

Generell kommt es beim Aufruf eines einzelnen Objekts in vielen Fällen vor, dass eine Reihe von Objekten referenziert wird, die mit dem aktuellen Objekt in Beziehung stehen. Für einzelne Eigenschaften ist es nur erlaubt, genau ein verbundenes Objekt zu referenziert (unter "Schema" gekennzeichnet mit einer "Kardinalität" von höchstens 1). Andere Eigenschaften erlauben die Verknüpfung einer beliebigen Anzahl von anderen Objekten. Ein Beispiel dafür liefert der Objekttyp `oparl:System`, der über die Eigenschaft `body` auf sämtliche Objekte vom Typ `oparl:Body` (Körperschaften) des Systems zeigt.

Dieses Kapitel beschreibt, wie solche Listen von verknüpften Objekten ausgegeben werden und welche Möglichkeiten dabei Server und Clients haben, um diese Ausgabe zu beeinflussen. Dabei werden die folgenden Konzepte behandelt:

- Interne und externe Ausgabe von Listen
- Kompakte und vollständige Form
- Paginierung
- Sortierung
- Filter

4.6.1 Interne und externe Ausgabe von Listen

Das folgende Beispiel zeigt eine Möglichkeit, wie die Eigenschaft `body` ausgegeben werden kann. Dabei handelt es sich um eine *interne*, also die Ausgabe der Listenelemente direkt im eigentlich abgerufenen Objekt.

```
{
    "id": "https://oparl.example.org/",
    "type": "http://oparl.org/schema/1.0/System",
    "body": [
        "https://oparl.example.org/bodies/1",
        "https://oparl.example.org/bodies/2",
        "https://oparl.example.org/bodies/3"
    ],
    ...
}
```

Wie oben zu sehen ist, ist der Wert der Eigenschaft `body` in diesem Fall ein Array. Die Einträge des Arrays sind URLs. Es handelt sich dabei um die URLs aller Objekte vom Typ `oparl:Body`, die mit dem gezeigten `oparl:System` Objekt in Beziehung stehen.

Eine alternative Möglichkeit für die Ausgabe der selben Information ist die *externe* Listenausgabe. Mit dieser Form der Ausgabe sieht das oben gezeigte Objekt nun so aus:

```
{
    "id": "https://oparl.example.org/",
    "type": "http://oparl.org/schema/1.0/System",
    "body": "https://oparl.example.org/bodies/",
    ...
}
```

In diesem Fall ist der Wert der Eigenschaft `body` kein Array, sondern eine URL. Diese URL kann vom Client genutzt werden, um die entsprechende Liste mit Objekten aufzurufen. Wie die entsprechende Ausgabe des Servers aussieht, wird weiter unten unter beschrieben.

Diese beiden Mechanismen sind grundsätzlich immer anwendbar, wenn eine Mehrzahl von Objekten mit einem Objekt verknüpft ist, unabhängig von der Art des verknüpfenden oder des verknüpften Objekts.

Die Entscheidung, ob eine Liste intern, also im Kontext eines einzelnen Objekts, oder extern, also über eine eigene URL ausgegeben wird, obliegt allein dem Server. Bei der Abwägung durch den Server sollte dieser berücksichtigen:

1. Die interne Listenausgabe eignet sich für kleine Listen mit wenigen Elementen.
2. Die externe Listenausgabe eignet sich für längere Listen, da hier auch Paginierung und Filterung möglich sind.
3. Die externe Listenausgabe ermöglicht über die Ausgabe einer URL hinaus auch die Ausgabe vollständiger Objekte.

Mehr zu 2. und 3. ist den folgenden Abschnitten zu entnehmen.

Die externe Listenausgabe wird explizit EMPFOHLEN in den folgenden Fällen:

- Eine Liste wächst mit der Zeit, wie z.B. die Liste aller Drucksachen einer Körperschaft.
- Es handelt sich um Listen von Objekten des Typs `oparl:Paper` (Drucksache) oder `oparl:Meeting`

- Es handelt sich bei der Liste um einen Feed.

Server DÜRFEN in den URLs für die externe Ausgabe von Listen NICHT den reservierten URL-Parameter `listformat` verwenden. Server MÜSSEN in den URLs für den Listenaufruf stets die URL zum Abruf der *kompakten Form* ausgeben. Die Unterscheidung zwischen *kompakter* und *vollständiger* Form wird nachfolgend beschrieben.

4.6.2 Kompakte und vollständige Form (`listformat`)

Wie im vorangehenden Abschnitt beschrieben, gibt es die Möglichkeit, Listen von Objekten über eine eigene URL zugänglich zu machen (*externe Listenausgabe*). Bei dieser externen Ausgabe gibt es zwei verschiedene Ausgabeformate, die sich durch den Umfang der Informationen unterscheiden, die je Objekt ausgegeben werden:

- **Kompakte Form**: Hier wird je Eintrag nur die URL des Objekts ausgegeben.
- **Vollständige Form**: Hier wird jedes Objekt in der Liste vollständig ausgegeben. Was genau "vollständig" bedeutet, wird nachstehend näher beschrieben.

Die Entscheidung, ob die kompakte oder die vollständige Form ausgegeben wird, obliegt dem Client. Dieser aktiviert die vollständige Ausgabe über den URL-Parameter `listformat`. Ist dieser Parameter nicht gesetzt, MUSS der Server die kompakte Form ausgeben. Ist der Parameter auf den Wert `complete` gesetzt, MUSS der Server die vollständige Form ausgeben.

Hat beispielsweise der Server zum externen Aufruf der Liste die URL

`https://oparl.example.org/bodies/1/papers/`

ausgegeben, ist unter dieser grundsätzlich die kompakte Form zu erwarten. Der Client kann diese URL so erweitern, um die vollständige Form anzufordern:

`https://oparl.example.org/bodies/1/papers/?listformat=complete`

Das folgende Beispiel zeigt, wie die Ausgabe der kompakten Form in einem einfachen Fall aussehen kann:

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

Die grundlegende Syntax ist für die externe Listenausgabe identisch, unabhängig davon, ob die kompakte oder vollständige Form ausgegeben wird: Der Server gibt ein JSON-Objekt aus, das eine Eigenschaft `items` enthält. Diese Eigenschaft hat den Typ `Array`.

Die vollständige Form ist so definiert, dass darin jedes Objekt mit allen von OParl für diesen Typ definierten Eigenschaften ausgegeben werden MUSS, die auch beim individuellen Aufruf des jeweiligen Objekts ausgegeben werden.

Das nachfolgende Beispiel zeigt dies verkürzt, analog zur oben gezeigten Liste:

```
{
    "items": [
        {
            "id": "https://oparl.example.org/bodies/0/papers/2",
            "type": "http://oparl.org/schema/1.0/Paper",
            "body": "https://oparl.example.org/bodies/0",
            "name": "Antwort auf Anfrage 1200/2014",
            "publishedDate": "2014-04-04T16:42:02+02:00",
            "paperType": "https://oparl.example.org/vocab/answer",
            "mainDocument": "https://oparl.example.org/documents/925",
            "originator": [
                "https://oparl.example.org/organization/2000"
            ]
        },
        {
            "id": "https://oparl.example.org/bodies/0/papers/5",
            "type": "http://oparl.org/schema/1.0/Paper",
            "body": "https://oparl.example.org/bodies/0",
            "name": "Mitteilung der Verwaltung",
            "publishedDate": "2014-06-01T12:24:18+02:00",
            "paperType": "https://oparl.example.org/vocab/message",
            "mainDocument": "https://oparl.example.org/documents/2758",
            "originator": [
                "https://oparl.example.org/people/1000"
            ]
        },
        ...
    ]
}
```

Wie zu sehen ist, hat die Eigenschaft `items` als Wert nun ein Array mit JSON-Objekten.

Die Anforderung der vollständigen Form wirkt sich *nicht rekursiv* aus. Die einzelnen JSON-Objekte können ihrerseits wieder Eigenschaften haben, die auf mehrere Objekte verweisen. Diese Eigenschaften sind von der Anforderung der vollständigen Listenausgabe durch den Client nicht betroffen. Hier obliegt es wieder dem Server, zwischen der internen und der externen Listenausgabe (siehe oben) zu wählen. Bei der internen Listenausgabe ist ohnehin nur die kompakte Form (Ausgabe von URLs), wie oben im Beispiel gezeigt, erlaubt.

Die Sortierreihenfolge der ausgegebenen Einträge SOLL unabhängig von der Ausgabe der kompakten oder volläständigen Form identisch sein.

Die vollständige Listenausgabe SOLL nur für Listen verwendet werden, die bis zu 100 Einträge umfassen. Links zu solch kurzen Listen wir jedoch in OParl-Objekten in der Regel nicht enthalten.

4.6.3 Paginierung

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

Die Entscheidung, ob eine Seite teilweise und daher mit Paginierung ausgegeben wird, liegt allein beim Server. Bei Listen mit mehr als 100 Einträgen ist dies EMPFOHLEN.

Der Server gibt für eine Liste, bei der die Paginierung aktiv ist, d. h. nicht alle Listenelemente ausgegeben wurde, zusätzliche Eigenschaften aus. Das nachfolgende Beispiel zeigt dies für den Anfang einer paginierten Liste:

```
{
    "items": [
        "https://oparl.example.org/bodies/0/papers/2",
        "https://oparl.example.org/bodies/0/papers/5",
        "https://oparl.example.org/bodies/0/papers/7",
        ...
    ],
    "itemsPerPage": 100,
    "nextPage": "https://oparl.example.org/bodies/0/papers/?skip_id=495"
}
```

Über die ZWINGEND bei Paginierung ausgegebene Eigenschaft `itemsPerPage` kommuniziert der Server, wie viele Einträge pro Listenseite ausgegeben werden. Die Zahl der Einträge, die der Server dabei je Listenseite ausliefert, SOLL dabei mindestens 10 und maximal 100 betragen. Der Wert von `itemsPerPage` MUSS auf allen Listenseiten der selben Liste einheitlich sein. Nur bei Ausgabe der letzten Listenseite DARF der Server weniger Einträge ausgeben, als von `itemsPerPage` angegeben.

Weiterhin wird bei Paginierung über eine Eigenschaft `nextPage` eine URL zum Abruf der jeweils nächsten Listenseite ausgegeben. Die Beschaffenheit der URL bestimmt der Server frei, das obige Beispiel ist in keiner Form bindend.

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

1. Der Server stellt eine URL für eine Liste zur Verfügung.
2. Der Client ruft diese URL der Liste auf.
3. Der Server antwortet mit einer verkürzten Listenausgabe und gibt mittels `nextPage`-Eigenschaft die URL für den Abruf der nächsten Listenseite an.
4. Der Client ruft die URL für die nächste Listenseite auf.

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

5. Der Server liefert die letzte Listenseite ohne `nextPage`-Eigenschaft aus.

Zusätzlich zur dem für die Paginierung ZWINGENDEN Eigenschaft `nextPage`, die lediglich auf der letzten Listenseite entfällt, können Server OPTIONAL weitere URLs zum Abruf bestimmter Listenseiten anbieten:

Erste Listenseite (Eigenschaft `firstPage`): Sofern die aktuell abgerufene Listenseite nicht den Anfang der Liste wiedergibt, KANN der Server diese Eigenschaft ausgeben, deren Wert die URL zum Abruf der *ersten* Listenseite ist.

Letzte Listenseite (Eigenschaft `lastPage`): Sofern die aktuell abgerufene Listenseite nicht das Ende der Liste wiedergibt, KANN der Server diese Eigenschaft ausgeben, deren Wert die URL zum Abruf der *letzten* Listenseite ist.

Vorherige Listenseite (Eigenschaft `prevPage`): Sofern die aktuell abgerufene Listenseite nicht den Anfang der Liste wiedergibt, KANN der Server diese Eigenschaft ausgeben, deren Wert die URL zum Abruf der *vorigen* Listenseite ist.

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

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

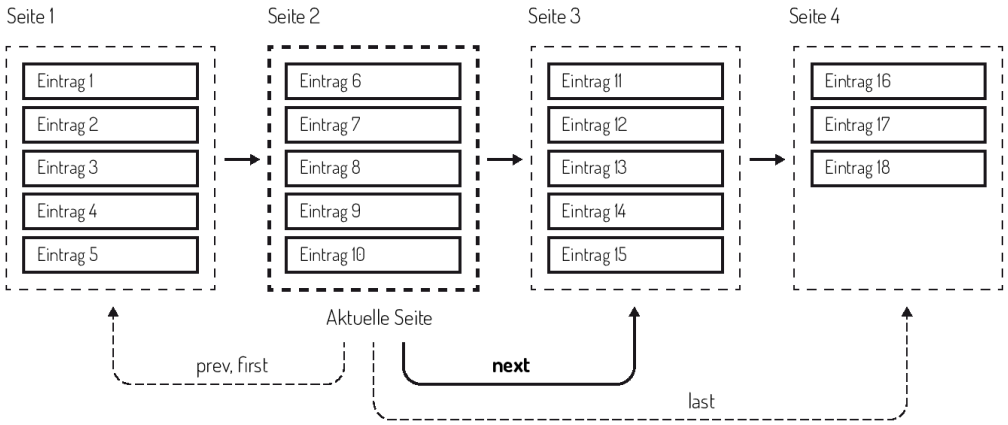

Abbildung 3: Paginierung: Schematische Darstellung

- Es ist davon auszugehen, dass Clients für den gesamten Abruf aller Seiten einer Liste längere Zeit benötigen. In der Zwischenzeit kann sich der Inhalt der Liste bereits ändern, etwa durch das Hinzukommen neuer Einträge. Die Paginierung ist idealerweise so zu implementieren, dass sich das Hinzukommen oder Entfernen von Einträgen möglichst nicht auf einen Client auswirkt, der aktuell die Liste paginiert, um alle Einträge abzurufen. Wir bezeichnen dies als **stabile Paginierung**.
- Eine wesentliche Anforderung an Listen mit Paginierung ist, dass alle Einträge der Liste in einer konsistenten Reihenfolge sortiert ausgegeben werden MÜSSEN. Das bedeutet, dass die Sortierung beim Server im Idealfall anhand einer eindeutigen und unveränderlichen Objekteigenschaft vorgenommen wird. Hierfür eignen sich die Objekt-URLs, da sie genau diese beiden Anforderungen erfüllen sollten.

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

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

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

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

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

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

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

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

4.6.4 Sortierung

OParl definiert keine Möglichkeit für Clients, auf die Reihenfolge von Listeneinträgen Einfluss zu nehmen. Von Servern wird die Einhaltung einiger grundlegender Anforderungen erwartet, die teilweise bereits erwähnt wurden.

Server SOLLEN generell für eine **stabile Sortierung** von Listeneinrägen sorgen. Das heißt, die Sortierung von Einträgen folgt einem konstanten Prinzip und ändert sich nicht von Abfrage zu Abfrage. Eine Einfache Möglichkeit, dies Umzusetzen, wäre in vielen Fällen die Sortierung von Objekten nach ihrer eindeutigen und unveränderlichen ID.

4.6.5 Filter

Bei der *externen Listenausgabe* (siehe weiter oben) werden in Abhängigkeit vom ausgegebenen Objekttyp bestimmte Möglichkeiten geboten, die Ausgabe von Listen auf eine Teilmenge einzuschränken.

Hierfür sind die URL-Paramter `startdate` und `enddate` vorgesehen. Beide können vom Client unabhängig von einander gesetzt werden. Sie schränken die Objektmenge anhand auf einen Zeitraum ein, der entweder einseitig oder beidseitig begrenzt ist. Die Einschränkung bezieht sich auf ein Bezugsdatum, das vom Objekttyp abhängt.

Der Filter mittels `startdate` und `enddate` ist nur auf Listen mit den folgenden Objekttypen anwendbar:

- `oparl:File`: Bezugsdatum ist hier die Eigenschaft `date` (Erstellungsdatum)
- `oparl:Meeting`: Bezugsdatum ist hier die Eigenschaft `start` (Startzeitpunkt der Sitzung)
- `oparl:Paper`: Bezugsdatum ist hier die Eigenschaft `publishedDate` (Veröffentlichungsdatum)

Für die genannten Objekttypen MUSS der Server bei externer Litenausgabe die beschriebenen Filter untertützen.

Die Filter werden vom Client aktiviert, indem der oder die gewünschte(n) URL-Parameter der vom Server angegebenen URL für die Listenausgabe hinzugefügt werden. Lautet diese URL für eine Liste von Drucksachen so,

```
https://oparl.example.org/papers/
```

dann kann der Client die folgende URL bilden, um die Ausgabe der Liste auf Drucksachen einzuschränken, die nach dem 1.1.2014 veröffentlicht wurden:

```
https://oparl.example.org/papers/?startdate=2014-01-01T00%3A00%3A00%2B01%3A00
```

Der Server interpretiert die Angabe eines `startdate` so, dass das Bezugsdatum aller ausgegebenen Objekte gleich oder größer dem im Parameter angegebenen Datum sein muss. Der Parameter `enddate` ist entsprechend so zu interpretieren, dass alle Bezugsdaten der ausgelieferten Objekte kleiner oder gleich dem angegebenen Wert sein müssen.

Sind beide Parameter gesetzt, handelt es sich um eine Boolsche UND-Verknüpfung. Der Server liefert entsprechend nur Objekte aus, deren Bezugsdatum größer/gleich dem Wert von `startdate` und kleiner/gleich dem Wert von `enddate` ist.

Clients MÜSSEN die Werte von `startdate` und `enddate` im Format xsd:dateTime, also mit Uhrzeit und Zeitzone, angeben (siehe dazu auch Datums- und Zeitangaben) und für eine entsprechende URL-Kodierung sorgen.

4.7 Feeds

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

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

Der Begriff "Feed" ist eine Anlehnung an die weit verbreiteten RSS-[34] oder Atom-Feeds[35], deren Publikationslogik im Wesentlichen auf der chronologischen Sortierung beruht. Im Unterschied zu Atom oder RSS ist hier jedoch keine XML-Ausgabe beabsichtigt.

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

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

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

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

4.7.1 Der Feed "Neue Objekte"

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

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

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

Das nachstehende Beispiel zeigt die mögliche Ausgabe des Feeds:

```
{
    "items": [
        {
            "id": "https://oparl.example.org/files/3",
            "type": "http://oparl.org/schema/1.0/File",
            "created": "2014-01-07T12:59:01+01:00"
        },
        {
            "id": "https://oparl.example.org/papers/21",
            "type": "http://oparl.org/schema/1.0/Paper",
            "created": "2014-01-05T18:29:37+01:00"
        },
        {
            "id": "https://oparl.example.org/files/5",
            "type": "http://oparl.org/schema/1.0/File",
            "created": "2014-01-04T11:26:48+01:00"
```

[34] RSS 2.0 Specification: http://cyber.law.harvard.edu/rss/rss.html
[35] Atom ist in RFC4287 spezifiziert: http://www.ietf.org/rfc/rfc4287.txt

```
        },
        ...
    ],
    "itemsPerPage": ...,
    "nextPage": ...
}
```

Die Funktionsweise entspricht grundsätzlich der von gewöhnlichen Listen mit Paginierung, wie im Kapitel Objektlisten beschrieben.

Davon abweichend gibt der Feed zu jedem neuen Objekt in der Liste unter `items` ein JSON-Objekt mit drei Eigenschaften aus:

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

Der jeweils in der Eigenschaft `created` ausgegebene Zeitpunkt SOLL vom Server als Sortierkriterium des Feeds genutzt werden.

4.7.2 Der Feed "Geänderte Objekte"

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

Die Definition einer "Änderung" kann sich zwischen den Objekttypen unterscheiden. Tendenziell soll die Definition eher weiter ausgelegt werden, als enger. Als Änderung einer Gruppierung (oparl:Organization) könnte es beispielsweise verstanden werden, wenn eine neue Mitgliedschaft zur Organisation hinzukommt. Das Erstellen eines Objekts (im Sinne des Feeds "Neue Objekte") sollte hingegen nicht als Änderung gewertet werden, um das redundante Erscheinen eines neuen Objekts sowohl im Feed "Neue Objekte" als auch im Feed "Geänderte Objekte" zu vermeiden.

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

```
{
    "items": [
        {
            "id": "https://oparl.example.org/files/2",
            "type": "http://oparl.org/schema/1.0/File",
            "modified": "2014-01-08T14:28:31+01:00"
        },
        {
            "id": "https://oparl.example.org/papers/0",
            "type": "http://oparl.org/schema/1.0/Paper",
            "modified": "2014-01-08T12:14:27+01:00"
        },
        {
            "id": "https://oparl.example.org/files/1",
            "type": "http://oparl.org/schema/1.0/File",
            "modified": "2014-01-06T17:01:00+01:00"
        },
    ],
    "itemsPerPage": ...,
    "nextPage": ...
}
```

Das Ausgabeformat entspricht weitgehend dem des Feeds “Neue Objekte”, jedoch heißt hier die Eigenschaft für den Zeitpunkt der letzten Änderung `modified`. Entsprechend gilt, dass der als `modified` ausgegebene Zeitpunkt als Sortierkriterium der Liste gelten SOLL.

4.7.3 Der Feed “Entfernte Objekte”

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

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

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

```
{
    "items": [
        {
            "id": "https://oparl.example.org/people/22",
            "removed": "2013-11-11T11:11:00+01:00"
        },
        ...
    ],
    "itemsPerPage": ...,
    "nextPage": ...
]
```

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

Im Unterschied zu den beiden zuvor beschriebenen Feeds wird im Feed “Gelöschte Objekte” keine Eigenschaft `type` am jeweiligen Objekt ausgegeben.

Clients SOLLEN vermeiden, die URLs der jeweiligen Einträge erneut aufzurufen.

4.8 Dateizugriff

Mit dem Begriff “Datei” sind im Sinne dieser Spezifikation alle Ressourcen gemeint, die von einem OParl-Server zur Verfügung gestellt werden und deren Metadaten über die JSON-API als `oparl:File` abgerufen werden können. Es handelt sich dabei beispielsweise um Textdokumente im PDF-Format, Abbildungen im JPEG- oder PNG-Format etc., die wesentliche Inhalte der parlamentarischen Informationen im OParl-System ausmachen.

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

Detail zu sämtlichen angesprochenen Mechanismen sind in den verschiedenen Teilen der HTTP-1.1-Spezifikation[36] zu finden.

4.8.1 GET und HEAD Anfragen

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

[36] vgl. http://tools.ietf.org/html/rfc7230, http://tools.ietf.org/html/rfc7231, http://tools.ietf.org/html/rfc7232

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

4.8.2 Allgemeiner Zugriff und expliziter Download

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

Im Unterschied dazu KANN der Server dem Client in der Eigenschaft `downloadUrl` eine weitere URL anbieten, hier *Download-URL* genannt. Diese dient im Gegensatz zur Zugriffs-URL speziell zum Herunterladen und Speichern der Datei in einem Dateisystem des Endnutzers. Bei Zugriff auf die Download-URL MUSS der Server in der HTTP-Antwort einen `Content-Disposition` Header senden.[37] Dieser Header MUSS als ersten Parameter den Typ `attachment` sowie den `filename`-Parameter mit dem Namen der Datei enthalten.

Beispiel:

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

FRAGE: Sind in Dateinamen sinnvoll?

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

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

4.8.3 Obligatorische und empfohlene Header

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

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

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

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

4.8.4 Conditional GET

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

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

[37] vgl. RFC2138 http://www.ietf.org/rfc/rfc2183

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

4.8.5 Zustandsloser Dateizugriff

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

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

4.8.6 Weiterleitungen

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

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

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

4.8.7 Entfernte Dateien

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

4.9 Content Negotiation

Das Prinzip *Content Negotiation* wurde bereits in RFC2295[38] beschrieben und bedeutet, dass WWW-Server eine Ressource in verschiedenen Formaten bereithalten können und Clients die Möglichkeit haben, eine Vorliebe für ein bestimmtes Format zu übermitteln. Auch die HTTP-1.1-Spezifikation[39] schließt Content Negotiation ein.

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

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

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

[38] RFC2295: http://tools.ietf.org/html/rfc2295
[39] RFC7231: http://tools.ietf.org/html/rfc7231#section-3.4

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

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

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

4.10 HTTP-Kompression

Die zwischen Servern und Clients übertragenen Datenvolumen SOLLEN mit Hilfe von Kompressionsverfahren reduziert werden, wenn sowohl der Client als auch der Server dies unterstützen. Dabei kommt das in HTTP 1.1[40] beschriebene Verfahren zum Einsatz.

HTTP 1.1 stellt drei komprimierte Kodierungen vor, wobei die Liste durch Registrierung neuer Verfahren bei der IANA erweitert werden kann. Diese sind:

- gzip
- compress
- deflate

Server-Implementierer SOLLEN mindestens eines dieser drei Verfahren unterstützen, wenn Clients dies mittels Accept-Encoding-Header anfragen.

Die Verwendung von HTTP-Kompression ist grundsätzlich sowohl bei JSON-Daten als auch bei Dateizugriffen möglich. Bei Dateizugriffen sind die zu erwartenden Einsparungen beim Datenvolumen stark abhängig vom jeweiligen Dateifomat. Bei bereits komprimierten Dateien wie beispielsweise OpenOffice oder PDF lassen sich oft nur geringe oder gar keine weiteren Ersparnisse erzielen. Daher DARF grundsätzlich der Server in solchen Fällen eine unkomprimierte HTTP-Antwort senden, auch wenn der Client ein unterstütztes Kompressionsverfahren angefragt hat.

4.11 Ausnahmebehandlung

Nicht immer kann ein Server die Anfrage eines Clients erfolgreich, also im Sinne der Anfrage, behandeln und eine entsprechende Antwort liefern. Beispiele für solche Ausnahmefälle könnten sein (ohne Anspruch auf Vollständigkeit):

- Eine angefragte Ressource existiert nicht
- Eine angefragte Ressource existiert nicht mehr
- Der Client nutzt eine nicht unterstützte HTTP-Methode, z. B. `POST`
- Der Client nutzt einen nicht interpretierbaren URL-Parameter

Die HTTP-1.1-Spezifikation sieht für derartige und weitere Ausnahmefälle zahlreiche spezifische Status-Codes vor, die hier nicht wiederholt werden sollen. Verallgemeinernd lautet die Anforderung an jeden OParl-Server, dass diese Ausnahmen mit den entsprechenden HTTP-Status-Codes beantworten SOLLEN. Damit wird Client-Entwicklern die Möglichkeit gegeben, diese Fälle entsprechend zu behandeln.

Clients DÜRFEN darüber hinaus nicht davon ausgehen, dass mit der HTTP-Antwort im Fall einer Ausnahme noch weitere verwertbare Informationen wie z. B. eine Fehlermeldung gesendet werden.

[40] RFC7231 Section 5.3.4: http://tools.ietf.org/html/rfc7231#section-5.3.4

4.12 Liste reservierter URL-Parameter

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

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

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

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

`listformat`: Parameter zur Steuerung der Listenausgabe. Siehe dazu: Objektlisten: Kompakte und vollständige Form

`subject`, `predicate`, `object`: Reserviert für eine eventuelle zukünftige Verwendung von Linked Data Fragments[41] in OParl.

5 Schema

Dieses Kapitel beschreibt das Schema von OParl. Das Schema bildet das Datenmodell 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.

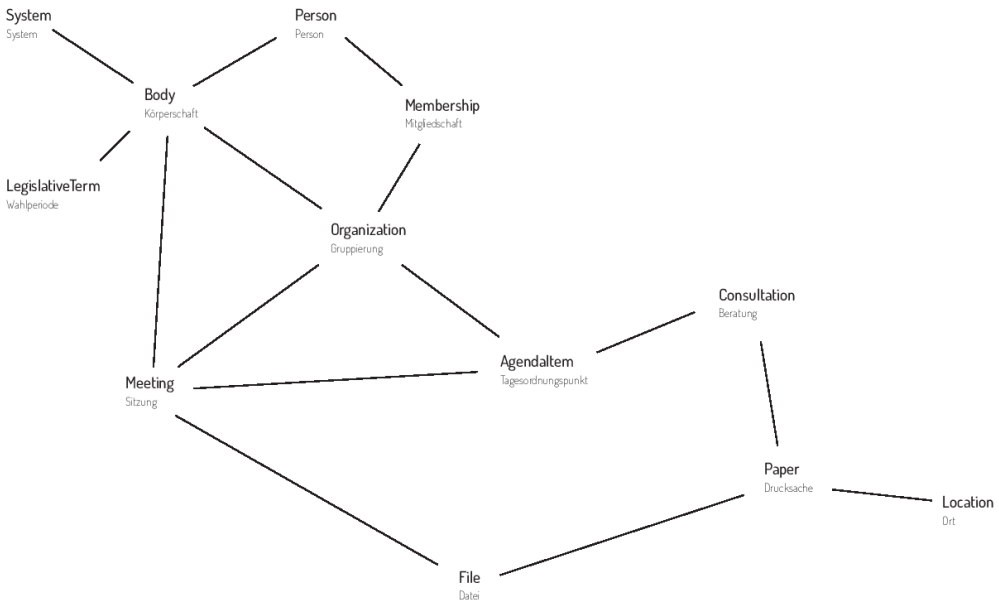

Abbildung 4: OParl Objekttypen: Ein Überblick

[41] Linked Data Fragments: http://linkeddatafragments.org/

5.1 Übergreifende Aspekte

5.1.1 Unicode-Zeichenketten als Standard

Die Schema-Beschreibung gibt zu jeder Eigenschaft eines Objekttypen an, welchen Typ der Wert dieser Eigenschaft haben muss.

Sofern keine Typ-Angabe zu einer Eigenschaft vorhanden ist, oder die Typ-Angabe `String` oder `xsd:string` lautet, werden Unicode-Zeichenketten als Datentyp erwartet.

5.1.2 `null`-Werte und "leere" Werte

JSON erlaubt es grundsätzlich, Eigenschaften mit dem Wert `null` zu versehen.

Clients MÜSSEN eine Eigenschaft mit dem Wert `null` so behandeln, als wäre die Eigenschaft nicht im Objekt vorhanden. OParl-Server SOLLEN die Ausgabe von Eigenschaften mit dem Wert `null` grundsätzlich vermeiden.

Analog dazu SOLLEN Server vermeiden, leere JSON-Arrays und -Objekte (`[]` und `{}`) auszugeben. Auch hier sind Clients dazu angehalten, diese wie nicht existierende Eigenschaften zu behandeln.

Ausnahmen bilden hier Eigenschaften, die ihrerseits als Pflichteigenschaften ("ZWINGEND") deklariert sind und die Kardinalität "1 bis *"besitzen, also eine Liste als Wert haben können. Diese Eigenschaften DÜRFEN auch dann gesetzt sein, wenn ihr Wert eine leere Liste ist.

5.1.3 Kardinalität

Zur expliziten Unterscheidung, ob eine Eigenschaft einen einzelnen Wert (z. B. eine Zeichenkette, eine URL, eine Zahl) oder alternativ eine Liste mit mehreren Elementen als Wert haben darf, ist in der Schema-Beschreibung zu jeder Eigenschaft die *Kardinalität* angegeben. Dabei sind verschiedene Angaben zur Eigenschaft möglich:

- 0 bis 1: OPTIONAL und MUSS NICHT gesetzt sein. Wenn sie gesetzt ist, DARF sie genau einen Wert haben.
- 1: MUSS gesetzt sein und genau einen Wert haben.
- 0 bis *: OPTIONAL und MUSS NICHT gesetzt sein. Wenn sie gesetzt ist, DARF sie beliebig viele Werte haben.
- 1 bis *: MUSS vorhanden sein, es MUSS mindestens ein Wert gesetzt sein. Es DÜRFEN auch mehrere Werte vorhanden sein.

Zur Ausgabe von Listen innerhalb eines Objekts sowie über eigene URLs finden sich ausführlichere Erläuterungen im Abschnitt Objektlisten.

5.1.4 Datums- und Zeitangaben

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

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

Diese Spezifikationen stützen sich auf RFC 3339[42]) und RFC 3339 wiederum auf ISO 8601.

[42]RFC3339: http://www.ietf.org/rfc/rfc3339.txt

In der vorliegenden Spezifikation verwenden wir den Präfix `xsd`, um Eigenschaften aus der XMLSchema-Spezifikation zu referenzieren.[43]. Datums- und Zeittyp werden entsprechend in diesem Dokument als `xsd:date` bzw. `xsd:dateTime` bezeichnet.

5.1.5 Vokabulare zur Klassifizierung

Einige Objekttypen besitzen Eigenschaften zum Zweck der Klassifizierung von Objekten. Im Einzelnen sind dies:

- `classification` des Objekttyps `oparl:Organization`
- `documentRole` des Objekttyps `oparl:File`
- `formOfAddress` des Objekttyps `oparl:Person`
- `keyword` in mehreren Objekttypen
- `paperType` des Objekttyps `oparl:Paper`
- `post` des Objekttyps `oparl:Organization`
- `result` des Objekttyps `oparl:AgendaItem`
- `role` des Objekttyps `oparl:Consultation`
- `role` des Objekttyps `oparl:Membership`
- `role` des Objekttyps `oparl:Person`
- `status` des Objekttyps `oparl:Person`
- `title` des Objekttyps `oparl:Person`

Diese Eigenschaften können als Wert wahlweise einfache Zeichenketten (Strings) haben, z. B. `"Beantwortung einer Anfrage"` oder aber URLs. Wenn eine URL verwendet wird, MUSS diese auf ein JSON-LD-Objekt[44] vom Typ `skos:Concept` zeigen. Dieses Objekt MUSS eine Eigenschaft `prefLabel` besitzen, in dem die benutzerfreundliche Benennung des Konzepts wiedergegeben wird.[45]

Ein Beispiel für ein `skos:Concept` Objekt, wie es für die Eigenschaft `status` eines Objekts vom Typ `oparl:Person` genutzt werden kann:

```
{
    "@context": {
        "prefLabel": {
            "@id": "http://www.w3.org/2004/02/skos/core#prefLabel"
        }
    },
    "@type": "http://www.w3.org/2004/02/skos/core#Concept",
    "prefLabel": "Ratsherr | Ratsfrau"
}
```

Das Objekt darf unter einer beliebigen URL abgelegt werden. Diese kann, muss aber nicht Teil des jeweiligen OParl-Systems sein.

Sinnvoll wird die Verwendung von URLs zur Klasifizierung, wenn mehrere Systeme auf die selben URLs verweisen, damit also ein gemeinsames Vokabular zur Klassifizierung nutzen. Die Verwendung eines übergreifenden Vokabulars soll dazu beitragen, dass die automatisierte Auswertung von parlamentarischen Informationen über die Grenzen einzelner Systeme hinweg deutlich erleichtert wird. So könnte beispielsweise eine bestimmte Art von Drucksache über Systemgrenzen hinweg als solche erkannt werden, wenn die Systeme auf das selbe `skos:Concept` Objekt verweisen.

[43] Der Präfix "xsd" steht somit für die URL `http://www.w3.org/2001/XMLSchema#`
[44] JSON-LD 1.0: `http://www.w3.org/TR/json-ld/`
[45] Diese Konstrukte entstammen dem *Simple Knowledge Organization System* (SKOS): `http://www.w3.org/2009/08/skos-reference/skos.html`

Für die Zukunft ist geplant, dass OParl hierzu Vokabulare mit entsprechenden SKOS-Objekten zur Verfügung stellt, die dann von Datenanbietern per URL referenziert werden können.

Da die `skos:Concept` Objekte, die über eine URL verlinkt werden, praktisch keinen Änderungen unterliegen, SOLLEN Clients diese Ressourcen nur selten abrufen und das Ergebnis der Anfragen in ihrem eigenen Cache speichern. Server SOLLEN das Caching unterstützen, indem Sie die üblichen Mechanismen von HTTP-Headern wie `Expires` und `Max-age` nutzen.

5.1.6 Herstellerspezifische Erweiterungen

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

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

5.1.7 URL-Pfade in den Beispielen

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

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

```
https://ratsmitglied-max-mustermann.example.org/mein-oparl-datensatz
```

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

Wenn also in einer Beispiel-URL so etwas wie

```
bodies/0/peoples/
```

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

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

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

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

```
personen/?__=LfyIfvCWq8SpBQj0MiyHaxDZwGJ
```

oder auch z. B. dies ¸~~~ LfyIfvCWq8SpBQj0MiyHaxDZwGJ ¸~~~

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

5.2 Eigenschaften mit Verwendung in mehreren Objekttypen

5.2.1 `id`

Die Eigenschaft `id` ist für jeden OParl-Objekttypen vorgegeben und enthält den eindeutigen Bezeichner des Objekts, nämlich seine URL. Dies ist ein ZWINGENDES Merkmal für jedes Objekt.

TODO: Zu klären ist noch, ob das auch für oparl:Location gilt.

5.2.2 `type`

Objekttypenangabe des Objekts, ZWINGEND für jedes Objekt. Der Wert ist eine Namespace-URL. Für die OParl-Objekttypen sind die folgenden URLs definiert:

Typ (kurz)	Namespace-URL
`oparl:AgendaItem`	http://oparl.org/schema/1.0/AgendaItem
`oparl:Body`	http://oparl.org/schema/1.0/Body
`oparl:Consultation`	http://oparl.org/schema/1.0/Consultation
`oparl:File`	http://oparl.org/schema/1.0/File
`oparl:LegislativeTerm`	http://oparl.org/schema/1.0/LegislativeTerm
`oparl:Location`	http://oparl.org/schema/1.0/Location
`oparl:Meeting`	http://oparl.org/schema/1.0/Meeting
`oparl:Membership`	http://oparl.org/schema/1.0/Membership
`oparl:Organization`	http://oparl.org/schema/1.0/Organization
`oparl:Paper`	http://oparl.org/schema/1.0/Paper
`oparl:Person`	http://oparl.org/schema/1.0/Person
`oparl:System`	http://oparl.org/schema/1.0/System

5.2.3 `name` und `shortName`

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

Die Werte von `name` und `shortName` des selben Objekts SOLLEN nicht identisch sein.

5.2.4 `license`

Die Eigenschaft `license` erlaubt es, am jeweiligen Objekt die URL einer Lizenz anzugeben. Damit wird gekennzeichnet, welche Lizenz der Veröffentlicher der Daten für das jeweilige Objekt vergibt.[46]

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

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

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

Lesenswert zum Thema Lizensierung von Linked Data ist auch er Abschnitt “Licenses, Waivers and Norms for Data” im online zugänglichen Linked Data Book.[47]

5.2.5 `created`

Datum und Uhrzeit der Erstellung des jeweiligen Objekts.

Datentyp: `xsd:dateTime`.

5.2.6 `modified`

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

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

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

Datentyp: `xsd:dateTime`.

5.2.7 `keyword`

Die Eigenschaft `keyword` dient der Kategorisierung von Objekten und ist in einer Vielzahl von Objekttypen zu diesem Zweck einsetzbar.

Mehr zur Funktionsweise dieser Eigenschaft wird im Abschnitt Vokabulare zur Klassifizierung beschrieben.

5.3 oparl:System (System)

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

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 System zu bekommen und die URLs zum Zugriff auf andere Informationen in Erfahrung zu bringen.

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

Beispiel

```
{
    "id": "https://oparl.example.org/",
    "type": "http://oparl.org/schema/1.0/System",
    "oparlVersion": "http://oparl.org/specs/1.0/",
```

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

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

5.3.1 Eigenschaften

`oparlVersion` Die URL der OParl-Spezifikation, die von diesem Server unterstützt wird. Der Wert MUSS die URL `http://oparl.org/specs/1.0/` sein. Typ: URL. Kardinalität: 1. ZWINGEND.

`body` Liste der URLs der `oparl:Body`-Objekte, also der Körperschaften, die auf dem System vorliegen. Alternativ kann statt einer Liste eine einzelne URL zum Abruf der Liste angeboten werden. Typ: URL des `oparl:Body` Objekts Kardinalität: 1. ZWINGEND.

`name` Nutzerfreundlicher Name für das System, mit dessen Hilfe Nutzer das System erkennen und von anderen unterscheiden können. Typ: String. Kardinalität: 0 bis 1. EMPFOHLEN.

`contactEmail` E-Mail-Adresse für Anfragen zur OParl-API. Die Angabe einer E-Mail-Adresse dient sowohl Nutzerinnen wie auch Entwicklerinnen von Clients zur Kontaktaufnahme mit dem Betreiber. Typ: String im Format `foaf:mbox` Kardinalität: 0 bis 1. EMPFOHLEN.

`contactName` Name des Ansprechpartners oder der Abteilung, die über die `contactEmail` erreicht werden kann. Typ: String. Kardinalität: 0 bis 1. EMPFOHLEN.

`newObjects` URL des Feeds "Neue Objekte". Typ: URL. Kardinalität: 0 bis 1. EMPFOHLEN.

`updatedObjects` URL des Feeds "Geänderte Objekte". Typ: URL. Kardinalität: 0 bis 1. EMPFOHLEN.

`removedObjects` URL des Feeds "Entfernte Objekte". Typ: URL. Kardinalität: 0 bis 1. EMPFOHLEN.

`website` URL zur WWW-Oberfläche des parlamentarischen Informationssystems. Typ: URL. Kardinalität: 0 bis 1. OPTIONAL.

`vendor` Software-Anbieter, von dem die OParl-Server-Software stammt. Typ: URL. Kardinalität: 0 bis 1. OPTIONAL.

`product` Informationen zu der auf dem System genutzten OParl-Server-Software. Typ: URL. Kardinalität: 0 bis 1. OPTIONAL.

5.4 oparl:Body (Körperschaft)

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

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

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

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

Beispiel


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


5.4.1 Eigenschaften

`system` System, zu dem dieses Objekt gehört. Typ: URL des `oparl:System` Objekts. Kardinalität: 1. ZWINGEND.

`shortName` Kurzer Name der Körperschaft. Typ: Datentyp `xsd:string`. Kardinalität: 0 bis 1. EMPFOHLEN.

`name` Der offizielle lange Name der Körperschaft. Typ: Datentyp `xsd:string`. Kardinalität: 1. ZWINGEND.

`website` Allgemeine Website der Körperschaft. Typ: URL. Kardinalität: 0 bis 1. EMPFOHLEN.

`license` Lizenz, die für die Daten, die über diese API abgerufen werden können, gilt, sofern nicht am einzelnen Objekt anders angegeben. Siehe dazu auch die übergreifende Beschreibung zur Eigenschaft `license`. Typ: URL. Kardinalität: 0 bis 1. EMPFOHLEN.

`licenseValidSince` Zeitpunkt, seit dem die unter `license` angegebene Lizenz gilt. Vorsicht bei Änderungen der Lizenz die zu restriktiveren Bedingungen führen. Typ: `xsd:Date`. Kardinalität: 0 bis 1. EMPFOHLEN.

`rgs` Regionalschlüssel der Körperschaft als zwölfstellige Zeichenkette[48]. Typ: String. Kardinalität: 0 bis 1. EMPFOHLEN.

`equivalentBody` Dient der Angabe beliebig vieler zusätzlicher URLs, die die selbe Körperschaft repräsentieren. Hier können beispielsweise, sofern vorhanden, der entsprechende Eintrag der Gemeinsamen Normdatei der Deutschen Nationalbibliothek[49], der DBPedia[50] oder der Wikipedia[51] angegeben werden. Typ: Array mit URLs. Kardinalität: 0 bis *. EMPFOHLEN.

`contactEmail` Dient der Angabe einer Kontakt-E-Mail-Adresse mit "mailto:"-Schema. Die Adresse soll die Kontaktaufnahme zu einer für die Körperschaft und idealerweise das parlamentarische Informationssystem zuständigen Stelle ermöglichen. Typ: String im Format `foaf:mbox`. Kardinalität: 0 bis 1. EMPFOHLEN.

`contactName` Name oder Bezeichnung der mit `contactEmail` erreichbaren Stelle. Typ: String. Kardinalität: 0 bis 1. OPTIONAL.

`paper` Drucksacheen unter dieser Körperschaft. Vgl. Objektlisten. Typ: Liste von `oparl:Paper` Objekten. Kardinalität: 0 bis *. ZWINGEND.

`member` Personen in dieser Körperschaft. Vgl. Objektlisten. Typ: Liste von `oparl:Person` Objekten. Kardinalität: 0 bis *. ZWINGEND.

`meeting` Sitzungen dieser Körperschaft. Vgl. Objektlisten. Typ: Liste von `oparl:Meeting` Objekten. Kardinalität: 0 bis *. ZWINGEND.

`organization` Gruppierungen in dieser Körperschaft. Vgl. Objektlisten. Typ: Liste von `oparl:Organization` Objekten. Kardinalität: 0 bis *. ZWINGEND.

`legislativeTerm` Wahlperioden in dieser Körperschaft. Vgl. Objektlisten. Typ: Liste von `oparl:LegislativeTerm` Objekten. Kardinalität: 0 bis *. EMPFOHLEN.

`keyword` Schlagwort(e). Vgl. Vokabulare zur Klassifizierung. Typ: Array mit Strings oder URLs zu `skos:Concept` Objekten. Kardinalität: 0 bis *. OPTIONAL.

`created` Datum/Uhrzeit der Erzeugung des Objekts. Typ: `xsd:dateTime`. Kardinalität: 0 bis 1. EMPFOHLEN.

`modified` Datum/Uhrzeit der letzten Bearbeitung des Objekts. Typ: `xsd:dateTime`. Kardinalität: 0 bis 1. EMPFOHLEN.

5.5 oparl:Organization (Gruppierung)

Dieser Objekttyp dient dazu, Gruppierungen von Personen abzubilden, die in der parlamentarischen Arbeit eine Rolle spielen. Dazu zählen in der Praxis insbesondere Fraktionen und Gremien.[52]

Beispiel 1

```
{
    "id": "https://oparl.example.org/organization/34",
    "type": "http://oparl.org/schema/1.0/Organization",
    "body": "https://oparl.example.org/bodies/1",
    "name": "Ausschuss für Haushalt und Finanzen",
```

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

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

[50] DBPedia http://www.dbpedia.org/

[51] Wikipedia http://de.wikipedia.org/

[52] Ein Teil der Eigenschaften ist der "Organization" Ontologie (kurz: `org:Organization`) des W3C entnommen. Quelle: The Organization Ontology, W3C Recommendation 16 January 2014, http://www.w3.org/TR/vocab-org/. Deren Bezeichnungen wurden deshalb beibehalten. Das betrifft z.B. die Verwendung von `classification`.

```
    "shortName": "Finanzausschuss",
    "post": [
        "https://oparl.example.org/post/chairperson",
        "https://oparl.example.org/post/deputychairperson"
    ],
    "meeting": "https://oparl.example.org/meetings_for_org/34",
    "membership": [
        "https://oparl.example.org/memberships/27",
        "https://oparl.example.org/memberships/48",
        "https://oparl.example.org/memberships/57"
    ],
    "classification": "https://oparl.example.org/vocab/finance",
    "keyword": [
        "finanzen",
        "haushalt"
    ],
    "created": "2012-07-16T16:01:44+02:00",
    "startDate": "2012-07-17T00:00:00+02:00",
    "modified": "2012-08-16T14:05:27+02:00"
}
```

Beispiel 2

```
{
    "id": "https://oparl.example.org/organization/34",
    "type": "http://oparl.org/schema/1.0/Organization",
    "body": "https://oparl.example.org/bodies/1",
    "name": "Ausschuss für Haushalt und Finanzen",
    "meeting": "https://oparl.example.org/meetings_for_org/34",
    "membership": "https://oparl.example.org/meetings_for_org/34",
    "modified": "2012-08-16T14:05:27+02:00"
}
```

5.5.1 Eigenschaften

`body` Körperschaft, zu der diese Gruppierung gehört. Typ: URL eines `oparl:Body` Objekts. Kardinalität: 1. ZWINGEND.

`name` Offizielle (lange) Form des Namens der Gruppierung. Typ: Datentyp `xsd:string`. Kardinalität: 1. ZWINGEND.

`membership` Mitgliedschaften dieser Gruppierung. Typ: Liste von `oparl:Membership` Objekten. Kardinalität: 0 bis *. ZWINGEND.

`meeting` Sitzungen dieser Gruppierung. Invers zur Eigenschaft `organization` der Klasse `oparl:Meeting`. Da die Anzahl der Sitzungen stetig wachsen kann, wird EMPFOHLEN, die Liste über eine eigene URL verfügbar zu machen und damit Paginierung sowie die Filterung mittels startDate und endDate Parametern zu ermöglichen. Siehe dazu auch Objektlisten. Typ: Liste mit URLs von Objekten des Typs `oparl:Meeting`. Kardinalität: 0 bis *. ZWINGEND.

`shortName` Der Name der Gruppierung als Kurzform. Typ: Datentyp `xsd:string`. Kardinalität: 0 bis 1. OPTIONAL.

`post` Positionen, die für diese Gruppierung vorgesehen sind. Die Werte dieser Eigenschaft funktioniert wie in Vokabulare zur Klassifizierung beschrieben entweder als URL zu einem `skos:Concept` oder als String. Die Strings bzw. `prefLabel`-Eigenschaften der Objekte SOLLEN sowohl die männliche als auch die weibliche Form enthalten, und zwar in dem Muster “männliche Form | weibliche Form” (genau in der Reihenfolge mit

einem Leerzeichen vor und nach dem "|"). Wenn sich beide Formen nicht unterscheiden, dann DARF die Form nur einmal verwendet werden: "Mitglied" und nicht "Mitglied | Mitglied". Weitere Beispiele: "Vorsitzender | Vorsitzende", "1. Stellvertreter | 1. Stellvertreterin", "2. Stellvertreter | 2. Stellvertreterin", "Schriftführer | Schriftführerin", "Stellvertretender Schriftführer | Stellvertretende Schriftführerin", "Ordentliches Mitglied", "Stellvertretendes Mitglied". TODO: "Ordentliches Mitglied", "Stellvertretendes Mitglied" müssen anders behandelt werden! Typ: Liste von Strings oder URLs zu `skos:Concept` Objekten. Kardinalität: 0 bis *. OPTIONAL.

`subOrganizationOf` Ggf. URL der übergeordneten Gruppierung. Typ: `oparl:Organization`. Kardinalität: 0 bis 1. OPTIONAL.

`classification` Die Art der Gruppierung. In Frage kommen z.B. "Rat", "Hauptausschuss", "Ausschuss", "Beirat", "Projektbeirat", "Kommission", "AG", "Verwaltungsrat". Die Angabe soll möglichst präzise erfolgen. So ist die Angabe "Hauptausschuss" präziser als "Ausschuss". Im Vokabular SOLL dann dieses Verhältnis zwischen "Ausschuss" und "Hauptausschuss" kodiert sein ("https://oparl.example.org/hauptausschuss skos:broader https://oparl.example.org/ausschuss"). Vgl. Vokabulare zur Klassifizierung. Typ: `skos:Concept`. Kardinalität: 0 bis 1. EMPFOHLEN.

`keyword` Schlagworte. Vgl. Vokabulare zur Klassifizierung. Typ: `skos:Concept`. Kardinalität: 0 bis *. OPTIONAL.

`startDate` Gründungsdatum der Gruppierung. Kann z. B. das Datum der konstituierenden Sitzung sein. Typ: `xsd:date` oder `xsd:dateTime`. Kardinalität: 0 bis 1. EMPFOHLEN.

`endDate` Datum des letzten Tages der Existenz der Gruppierung. Typ: `xsd:date` oder `xsd:dateTime`. Kardinalität: 0 bis 1. OPTIONAL.

`created` Datum/Uhrzeit der Erzeugung des Objekts. Typ: `xsd:dateTime`. Kardinalität: 0 bis 1. EMPFOHLEN.

`modified` Datum/Uhrzeit der letzten Bearbeitung des Objekts. Typ: `xsd:dateTime`. Kardinalität: 0 bis 1. EMPFOHLEN.

5.6 oparl:Person (Person)

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

Beispiel


```
{
    "id": "https://oparl.example.org/person/29",
    "type": "http://oparl.org/schema/1.0/Person",
    "name": "Prof. Dr. Max Mustermann",
    "familyName": "Mustermann",
    "givenName": "Max",
    "title": [
        "https://oparl.example.org/vocab/person/title/prof",
        "https://oparl.example.org/vocab/person/title/dr"
    ],
    "formOfAddress": "https://oparl.example.org/vocab/foa/ratsmitglied",
    "gender": "https://oparl.example.org/vocab/person/gender/male",
    "email": "mailto:max@mustermann.de",
    "phone": "tel:+493012345678",
    "streetAddress": "Musterstraße 5",
    "postalCode": "11111",
    "locality": "Musterort",
```

```
    "status": "https://oparl.example.org/status/buergermeister",
    "hasMembership": [
        "https://oparl.example.org/membership/11",
        "https://oparl.example.org/membership/34"
    ],
    "created": "2011-11-11T11:11:00+01:00",
    "modified": "2012-08-16T14:05:27+02:00"
}
```

5.6.1 Eigenschaften

`name` Der vollständige Name der Person mit akademischem Grad und dem gebräuchlichen Vornamen, wie er zur Anzeige durch den Client genutzt werden kann. Typ: String. Kardinalität: 1. ZWINGEND.

`familyName` Familienname bzw. Nachname. Typ: String. Kardinalität: 0 bis 1. OPTIONAL.

`givenName` Vorname bzw. Taufname. Typ: String. Kardinalität: 0 bis 1. OPTIONAL.

`formOfAddress` Anrede. Diese Eigenschaft funktioniert wie in Vokabulare zur Klassifizierung beschrieben entweder als URL zu einem `skos:Concept` oder als String. Der String bzw. `prefLabel` SOLL sowohl die männliche als auch die weibliche Bezeichnung enthalten. Beispiele: "Herr | Frau", "Ratsherr | Ratsfrau". Typ: URL eines `skos:Concept` Objekts oder String. Kardinalität: 0 bis 1. OPTIONAL.

`title` Akademische(r) Titel. Vgl. Vokabulare zur Klassifizierung. Typ: URL eines `skos:Concept` Objekts oder String. Kardinalität: 0 bis *. OPTIONAL.

`gender` Geschlecht. Zulässige Werte sind `vcard:Female`, `vcard:Male`, `vcard:None` und `vcard:Other`. Für den Fall, dass das geschlecht der Person unbekannt ist, SOLL die Eigenschaft nicht ausgegeben werden. Typ: String im Format `vcard:Gender`. Kardinalität: 0 bis 1. OPTIONAL.

`phone` Telefonnummer der Person mit `tel:` Schema, ohne Leerzeichen. Typ: String Kardinalität: 0 bis 1. OPTIONAL.

`email` E-Mail-Adresse mit `mailto:` Schema. Typ: String im Format `foaf:mbox`. Kardinalität: 0 bis 1. OPTIONAL.

`streetAddress` Straße und Hausnummer der Kontakt-Anschrift der Person. Typ: String. Kardinalität: 0 bis 1. OPTIONAL.

`postalCode` Postleitzahl der Kontakt-Anschrift der Person. Typ: String. Kardinalität: 0 bis 1. OPTIONAL.

`locality` Ortsangabe der Kontakt-Anschrift der Person. Typ: `vcard:locality` Kardinalität: 0 bis 1. OPTIONAL.

`status` Status. Diese Eigenschaft funktioniert wie in Vokabulare zur Klassifizierung beschrieben entweder als URL zu einem `skos:Concept` oder als String. Die Strings bzw. `prefLabel` SOLLEN sowohl die männliche als auch die weibliche Form enthalten, und zwar in dem Muster "männliche Form | weibliche Form" (genau in der Reihenfolge mit einem Leerzeichen vor und nach dem "|"). Wenn sich beide Formen nicht unterscheiden, dann DARF die Form nur einmal verwendet werden: "Ratsmitglied" und nicht "Ratsmitglied | Ratsmitglied". Dadurch kann auch solche Software einen sinnvollen Text anzeigen, die keine Fall-Unterscheidung nach Geschlecht der Personen vornimmt. Weitere Beispiele: "Bürgermeister | Bürgermeisterin", "Bezirksbürgermeister | Bezirksbürgermeisterin", "Stadtverordneter | Stadtverordnete", "Bezirksverordneter | Bezirksverordnete", "Sachkundiger Bürger | Sachkundige Bürgerin", "Einzelstadtverordneter | Einzelstadtverordnete" (Mitglieder des Rates die keiner Fraktion/Organisation angehören). Vgl. Vokabulare zur Klassifizierung. Typ: URL eines `skos:Concept` Objekts oder String. Kardinalität: 0 bis *. OPTIONAL.

`hasMembership` Mitgliedschaften der Person in Gruppierungen, z. B. Gremien und Fraktionen. Typ: Liste von `org:Membership` Objekten. Kardinalität: 0 bis *. OPTIONAL.

`keyword` Diese Eigenschaft funktioniert wie in Vokabulare zur Klassifizierung beschrieben entweder als URL zu einem `skos:Concept` oder als String. Typ: URL eines `skos:Concept` Objekts oder String. Kardinalität: 0 bis *. OPTIONAL.

`created` Datum/Uhrzeit der Erzeugung des Objekts. Typ: `xsd:dateTime` Kardinalität: 0 bis 1. EMPFOHLEN.

`modified` Datum/Uhrzeit der letzten Bearbeitung des Objekts. Typ: `xsd:dateTime`. Kardinalität: 0 bis 1. EMPFOHLEN.

5.7 oparl:Meeting (Sitzung)

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

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

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

Beispiel


```
{
    "id": "https://oparl.example.org/meeting/281",
    "type": "http://oparl.org/schema/1.0/Meeting",
    "name": "4. Sitzung des Finanzausschusses",
    "start": "2013-01-04T08:00:00+01:00",
    "end": "2013-01-04T12:00:00+01:00",
    "streetAddress": "Musterstraße 5, Raum 136",
    "postalCode": "11111",
    "locality": "Musterort",
    "organization": "https://oparl.example.org/organization/34",
    "invitation": [
        "https://oparl.example.org/document/586"
    ],
    "resultsProtocol": "https://oparl.example.org/document/628",
    "verbatimProtocol": "https://oparl.example.org/document/691",
    "auxiliaryDocument": [
        "https://oparl.example.org/document/588",
        "https://oparl.example.org/document/589"
    ],
    "agendaItem": [
        "https://oparl.example.org/agendaitem/1045",
        "https://oparl.example.org/agendaitem/1046",
        "https://oparl.example.org/agendaitem/1047",
        "https://oparl.example.org/agendaitem/1048"
    ],
    "created": "2012-01-06T12:01:00+01:00",
    "modified": "2012-01-08T14:05:27+01:00"
}
```


5.7.1 Eigenschaften

`start` Datum und Uhrzeit des Anfangszeitpunkts der Sitzung. Bei einer zukünftigen Sitzung ist dies der geplante Zeitpunkt, bei einer stattgefundenen KANN es der tatsächliche Startzeitpunkt sein. Typ: Datentyp `xsd:dateTime`. Kardinalität: 1. ZWINGEND.

`end` Endzeitpunkt der Sitzung als Datum/Uhrzeit. Bei einer zukünftigen Sitzung ist dies der geplante Zeitpunkt, bei einer stattgefundenen KANN es der tatsächliche Endzeitpunkt sein. Typ: Datentyp `xsd:dateTime`. Kardinalität: 0 bis 1. EMPFOHLEN.

`streetAddress` Straße und Hausnummer der Kontakt-Anschrift der Person. Typ: String. Kardinalität: 0 bis 1. OPTIONAL.

`postalCode` Postleitzahl der Kontakt-Anschrift der Person. Typ: String. Kardinalität: 0 bis 1. OPTIONAL.

`locality` Ortsangabe der Kontakt-Anschrift der Person. Typ: `vcard:locality` Kardinalität: 0 bis 1. OPTIONAL.

`location` Sitzungsort in Form von Geodaten. Typ: URL eines `oparl:Location` Objekts. Kardinalität: 0 bis 1. OPTIONAL.

`organization` Gruppierung der die Sitzung zugeordnet ist. Wenn eine Liste angegeben wird, dann ist diese geordnet. Das erste Element ist dann das federführende Gremium. TODO: Eigenschaft für federführendes Gremium ergänzen und dann Ordnung entfernen. invers zur Eigenschaft `meeting` der Klasse `oparl:Organization`. Typ: `oparl:Organization`. Kardinalität: 1 bis *. ZWINGEND.

`chairPerson` Vorsitz der Sitzung Typ: `oparl:Person`. FRAGE: Was ist bei Wechsel des Vorsitzes während der Sitzung? Kardinalität: 0 bis 1. EMPFOHLEN.

`scribe` Schriftführer, Protokollant. Typ: `oparl:Person`. FRAGE: Können mehrere Personen vorkommen? Was ist bei Wechsel während der Sitzung? Kardinalität: 0 bis 1. EMPFOHLEN.

`participant` Teilnehmer der Sitzung. Bei einer Sitzung in der Zukunft sind dies die geladenen Teilnehmer, bei einer stattgefundenen Sitzung SOLL die Liste nur diejenigen Teilnehmer umfassen, die tatsächlich an der Sitzung teilgenommen haben. FRAGE: besser zwei separate Eigenschaften `attendant` und `invited` ? Typ: Liste von Objekten des Typs `oparl:Person`. Vgl. Objektlisten. Kardinalität: 0 bis *. DEPRECATED.

`invitation` Einladungsdokument zur Sitzung. FRAGE: Kann es mehr als ein solches Dokument geben? Typ: Liste von Objekten des Typs `oparl:File`. Vgl. Objektlisten. Kardinalität: 0 bis *. EMPFOHLEN.

`resultsProtocol` Ergebnisprotokoll zur Sitzung. Diese Eigenschaft kann selbstverständlich erst nach dem Stattfinden der Sitzung vorkommen. Typ: URL eines Objekts vom Typ `oparl:File`. Kardinalität: 0 bis 1. EMPFOHLEN.

`verbatimProtocol` Wortprotokoll zur Sitzung. Diese Eigenschaft kann selbstverständlich erst nach dem Stattfinden der Sitzung vorkommen. Typ: URL eines Objekts vom Typ `oparl:File`. Kardinalität: 0 bis 1. EMPFOHLEN.

`auxiliaryDocument` Dokumentenanhang zur Sitzung. Hiermit sind Dokumente gemeint, die üblicherweise mit der Einladung zu einer Sitzung verteilt werden, und die nicht bereits über einzelne Tagesordnungspunkte referenziert sind. Typ: Liste von Objekten des Typs `oparl:File`. Vgl. Objektlisten. Kardinalität: 0 bis *. OPTIONAL.

`agendaItem` Tagesordnungspunkte der Sitzung. Die Reihenfolge ist relevant. Es kann Sitzungen ohne TOPs geben. Typ: Liste von Objekten des Typs `oparl:AgendaItem`. Vgl. Objektlisten. Kardinalität: 0 bis *. OPTIONAL.

`keyword` Schlagworte. Diese Eigenschaft funktioniert wie in Vokabulare zur Klassifizierung beschrieben entweder als URL zu einem `skos:Concept` oder als String. Typ: Liste von Strings oder URLs zu `skos:Concept` Objekten. Kardinalität: 0 bis *. OPTIONAL.

`created` Datum und Uhrzeit der Erzeugung des Objekts. Typ: Datentyp `xsd:dateTime`. Kardinalität: 0 bis 1. EMPFOHLEN.

`modified` Datum und Uhrzeit der letzten Änderung des Objekts. Typ: Datentyp `xsd:dateTime`. Kardinalität: 0 bis 1. EMPFOHLEN.

5.8 oparl:AgendaItem (Tagesordnungspunkt)

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

Die Beziehung zwischen einem Tagesordnungspunkt und einer Drucksache wird über ein Objekt vom Typ `oparl:Consultation` hergestellt, das über die Eigenschaft `consultation` referenziert werden kann.

Beispiel

```
{
    "id": "https://oparl.example.org/agendaitem/3271",
    "type": "http://oparl.org/schema/1.0/AgendaItem",
    "meeting": "https://oparl.example.org/meeting/281",
    "number": "10.1",
    "name": "Satzungsänderung für Ausschreibungen",
    "public": true,
    "consultation": "https://oparl.example.org/consultation/1034",
    "result": "https://oparl.example.org/vocab/decided_modified",
    "resolution": "Der Beschluss weicht wie folgt vom Antrag ab: ...",
    "modified": "2012-08-16T14:05:27+02:00"
}
```

5.8.1 Eigenschaften

`meeting` Sitzung, der der Tagesordnungspunkt zugeordnet ist. Typ: URL eines Objekts vom Typ `oparl:Meeting`. Kardinalität: 1. ZWINGEND.

`number` Gliederungs-"Nummer" des Tagesordnungspunktes. Eine beliebige Zeichenkette, wie z. B. "10.", "10.1", "C", "c)" o. ä. Die Reihenfolge wird nicht dadurch, sondern durch die Reihenfolge der TOPs im `agendaItem`-Attribut von `oparl:Meeting` festgelegt. Typ: String. Kardinalität: 0 bis 1. OPTIONAL.

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

`public` Kennzeichnet, ob der Tagesordnungspunkt zur Behandlung in öffentlicher Sitzung vorgesehen ist/war. Es wird ein Wahrheitswert (`true` oder `false`) erwartet. Typ: Boolean. Kardinalität: 0 bis 1. EMPFOHLEN.

`consultation` Beratung, die diesem Tagesordnungspunkt zugewiesen ist. Typ: URL eines Objekts vom Typ `oparl:Consultation`. Kardinalität: 0 bis 1. FRAGE: Wirklich immer nur maximal 1 ? EMPFOHLEN.

`result` Kategorische Information darüber, welches Ergebnis die Beratung des Tagesordnungspunktes erbracht hat, in der Bedeutung etwa "Unverändert beschlossen" oder "Geändert beschlossen". Diese Eigenschaft funktioniert wie in Vokabulare zur Klassifizierung beschrieben entweder als URL zu einem `skos:Concept` oder als String. Typ: String oder URL eines `skos:Concept` Objekts. Kardinalität: 0 bis 1. EMPFOHLEN.

`resolution` Falls in diesem Tagesordnungspunkt ein Beschluss gefasst wurde, kann hier ein Text oder Dokument angegeben werden. Das ist besonders dann in der Praxis relevant, wenn der gefasste Beschluss (z. B. durch Änderungsantrag) von der Beschlussvorlage abweicht. Typ: String oder URL eines Objekts vom Typ `oparl:File`. Kardinalität: 0 bis 1. OPTIONAL.

`auxiliaryDocument` Dateianhänge zum Tagesordnungspunkt. Typ: Liste von Objekten des Typs `oparl:File`. Vgl. Objektlisten. Kardinalität: 0 bis *. OPTIONAL.

`keyword` Schlagwort. Diese Eigenschaft funktioniert wie in Vokabulare zur Klassifizierung beschrieben entweder als URL zu einem `skos:Concept` oder als String. Typ: Liste von Strings oder URLs zu `skos:Concept` Objekten. Kardinalität: 0 bis *. OPTIONAL.

`created` Erzeugungsdatum und -zeit des Objekts. Typ: `xsd:dateTime`. Kardinalität: 0 bis 1. EMPFOHLEN.

`modified` Datum und Uhrzeit der letzten Änderung. Typ: `xsd:dateTime`. Kardinalität: 0 bis 1. EMPFOHLEN.

5.9 oparl:Paper (Drucksache)

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

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

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

Beispiel


```
{
    "id": "https://oparl.example.org/paper/749",
    "type": "http://oparl.org/schema/1.0/Paper",
    "body": "https://oparl.example.org/bodies/1",
    "name": "Antwort auf Anfrage 1200/2014",
    "reference": "1234/2014",
    "publishedDate": "2014-04-04T16:42:02+02:00",
    "paperType": "https://oparl.example.org/vocab/answer",
    "relatedPaper": [
        "https://oparl.example.org/paper/699"
    ],
    "mainDocument": "https://oparl.example.org/document/925",
    "auxiliaryDocument": [
        "https://oparl.example.org/document/926"
    ],
    "location": [
        "https://oparl.example.org/locations/4472"
    ],
    "originator": [
        "https://oparl.example.org/organization/2000",
        "https://oparl.example.org/people/1000"
    ],
    "consultation": [
        "https://oparl.example.org/consultation/5676",
        "https://oparl.example.org/consultation/5689"
    ],
    "underDirectionOf": [
        "https://oparl.example.org/organization/2000"
    ],
    "modified": "2013-01-08T12:05:27+01:00"
}
```

5.9.1 Eigenschaften

`body` Körperschaft, zu der die Drucksache gehört. Typ: `oparl:Body`. Kardinalität: 1. ZWINGEND.

`name` Titel der Drucksache. Typ: String. Kardinalität: 1. ZWINGEND.

`reference` Kennung bzw. Aktenzeichen der Drucksache, mit der sie in der parlamentarischen Arbeit eindeutig referenziert werden kann. Typ: String. Kardinalität: 0 bis 1. OPTIONAL.

`publishedDate` Veröffentlichungsdatum der Drucksache. Typ: `xsd:dateTime` oder `xsd:date`. Kardinalität: 0 bis 1. ZWINGEND.

`paperType` Art der Drucksache, z. B. "Beantwortung einer Anfrage". Diese Eigenschaft funktioniert wie in Vokabulare zur Klassifizierung beschrieben entweder als URL zu einem `skos:Concept` oder als String. Kardinalität: 0 bis 1. Typ: String oder URL eines `skos:Concept` Objekts. EMPFOHLEN.

`relatedPaper` Inhaltlich verwandte Drucksachen. Typ: Liste von Objekten des Typs `oparl:Paper`. Vgl. Objektlisten. Kardinalität: 0 bis *. OPTIONAL.

`mainDocument` Das Hauptdokument zu dieser Drucksache. Beispiel: Die Drucksache repräsentiert eine Beschlussvorlage und das Hauptdokument enthält den Text der Beschlussvorlage. Typ: URL eines Objekts vom Typ `oparl:File`. Kardinalität: 1. ZWINGEND.

`auxiliaryDocument` Anhänge zur Drucksache. Diese sind, in Abgrenzung zum Hauptdokument (`mainDocument`), untergeordnet und es kann beliebig viele davon geben. Typ: Liste von Objekten des Typs `oparl:File`. Vgl. Objektlisten. Kardinalität: 0 bis *. OPTIONAL.

`location` Sofern die Drucksache einen inhaltlichen Ortsbezug hat, beschreibt diese Eigenschaft den Ort in Textform und/oder in Form von Geodaten. Typ: Liste von Objekten des Typs `oparl:Location`. Vgl. Objektlisten. Kardinalität: 0 bis *. OPTIONAL.

`originator` Urheber der Drucksache, kann eine oder mehrere Person(en) bzw. Gruppierung(en) sein. Typ: `oparl:Person` | `oparl:Organization`. Kardinalität: 0 bis *. EMPFOHLEN.

`consultation` Beratungen der Drucksache. Typ: `oparl:Consultation`. Kardinalität: 0 bis *. OPTIONAL.

`modified` Letzter Änderungszeitpunkt des Objekts. Typ: Datentyp `xsd:dateTime`. Kardinalität: 1. EMPFOHLEN.

`keyword` Schlagworte. Diese Eigenschaft funktioniert wie in Vokabulare zur Klassifizierung beschrieben entweder als URL zu einem `skos:Concept` oder als String. Typ: Liste von Strings oder URLs zu `skos:Concept` Objekten. Kardinalität: 0 bis *. OPTIONAL.

`underDirectionOf` Federführung. Amt oder Abteilung, für die Inhalte oder Beantwortung der Drucksache verantwortlich. Typ: Liste von Objekten des Typs `oparl:Organization`. Kardinalität: 0 bis *. OPTIONAL.

5.10 oparl:File (Datei)

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

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

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

Beispiel


```
{
    "id": "https://oparl.example.org/document/57739",
    "type": "http://oparl.org/schema/1.0/File",
    "name": "Anlage 1 zur Anfrage",
    "fileName": "57739.pdf",
    "paper": [
        "https://oparl.example.org/paper/2396"
    ],
    "mimeType": "application/pdf",
    "date": "2013-01-04T07:54:13+01:00",
    "modified": "2013-01-04T07:54:13+01:00",
    "sha1Checksum": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
    "size": 82930,
    "accessUrl": "https://oparl.example.org/document/57739.pdf",
    "downloadUrl": "https://oparl.example.org/document/download/57739.pdf",
    "text": "Der Übersichtsplan zeigt alle Ebenen des ...",
    "masterDocument": "https://oparl.example.org/document/57738",
    "license": "http://www.opendefinition.org/licenses/cc-by",
    "documentRole": "https://oparl.example.org/vocab/file/role/evidence"
}
```


5.10.1 Eigenschaften

`fileName` Dateiname, unter dem die Datei in einem Dateisystem gespeichert werden kann. Beispiel: "einedatei.pdf" Typ: ASCII-Zeichenkette, aber als Unicode-String Kardinalität: 1. ZWINGEND.

`name` Ein zur Anzeige für Endnutzer bestimmter Name für dieses Objekt. Leerzeichen DÜRFEN enthalten sein, Datei-Endungen wie ".pdf" SOLLEN NICHT enthalten sein. Der Wert SOLL NICHT mit dem Wert der Eigenschaft `fileName` identisch sein. Typ: String. Kardinalität: 0 bis 1. EMPFOHLEN.

`mimeType` MIME-Type des Inhalts[53]. Sollte das System einer Datei keinen spezifischen Typ zuweisen können, wird EMPFOHLEN, hier `application/octet-stream` zu verwenden. Typ: String. Kardinalität: 1. EMPFOHLEN.

`date` Datum und Zeit der Erstellung der Datei. Wahlweise, falls dies nicht vom System kommuniziert werden kann oder soll, KANN alternativ der Zeitpunkt der Veröffentlichung ausgegeben werden. Typ: xsd:dateTime. Kardinalität: 1. ZWINGEND.

`modified` Datum und Zeit der letzten Änderung der Datei bzw. der Metadaten. Als Änderung der Datei gilt alles, was den Inhalt der Datei verändert und beispielsweise zu einer Veränderung der Prüfsumme führen würde, nicht aber die Änderung des Dateinamens (siehe Eigenschaft `name`). Als Änderung der Metadaten hingegen würde beispielsweise die Änderung des Dateinamens gelten. Hier soll immer das größere der beiden Daten ausgegeben werden, also der am wenigsten lang zurückliegende Änderungszeitpunkt. Typ: xsd:dateTime. Kardinalität: 1. ZWINGEND.

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

`size` Größe der Datei in Bytes. Typ: ganze Zahl. Kardinalität: 1. ZWINGEND.

`sha1Checksum` SHA1-Prüfsumme des Dokumenteninhalts in Hexadezimal-Schreibweise. Typ: String. Kardinalität: 0 bis 1. OPTIONAL.

`text` Reine Text-Wiedergabe des Dateiinhalts, sofern dieser in Textform wiedergegeben werden kann. Typ: String. Kardinalität: 0 bis 1. EMPFOHLEN.

`accessUrl` URL zum allgemeinen Zugriff auf die Datei. Näheres unter Dateizugriff. Typ: URL. Kardinalität: 1. ZWINGEND.

`downloadUrl` URL zum Download der Datei. Näheres unter Dateizugriff. Typ: URL. Kardinalität: 0 bis 1. EMPFOHLEN.

`paper` Falls die Datei zu einer oder mehreren Drucksaches (`oparl:Paper`) gehört, MÜSSEN diese Drucksachen über diese Eigenschaft angegeben werden. Typ: Liste von `oparl:Paper` Objekten. Kardinalität: 0 bis *. EMPFOHLEN.

`meeting` Falls die Datei zu einer oder mehreren Sitzungen (`oparl:Meeting`) gehört, MÜSSEN diese Sitzungen über diese Eigenschaft angegeben werden. Typ: Liste von `oparl:Meeting` Objekten. Kardinalität: 0 bis *. EMPFOHLEN.

`masterDocument` Datei, von der das aktuelle Objekt abgeleitet wurde. Details dazu in der allgemeinen Beschreibung weiter oben. Typ: URl eines Objekts vom Typ `oparl:File`. Kardinalität: 0 bis 1. OPTIONAL.

`derivativeDocument` Datei, die von dem aktuellen Objekt abgeleitet wurde. Details dazu in der allgemeinen Beschreibung weiter oben. TODO: invers zu `masterDocument`. Von der Verwendung wird deshalb in der `prov`-Spezifikation abgeraten[54]. Typ: Liste von `oparl:File` Objekten. Kardinalität: 0 bis *. OPTIONAL.

`license` Lizenz, unter der die Datei angeboten wird. Wenn diese Eigenschaft verwendet wird, dann ist sie anstelle einer globalen Angabe im übergeordneten `oparl:Body`- bzw. `oparl:System`-Objekt maßgeblich.[55] Typ: URL. Kardinalität: 0 bis 1. OPTIONAL.

`documentRole` Rolle, Funktion, Sorte des Dokuments in Bezug auf eine Sitzung. Die Eigenschaft SOLL entsprechend nur in Verbindung mit der Eigenschaft `meeting` gesetzt sein. Siehe dazu Vokabulare zur Klassifizierung. Typ: String oder URL eines `skos:Concept` Objekts. Kardinalität: 0 bis 1. OPTIONAL.

`keyword` Schlagworte. Diese Eigenschaft funktioniert wie in Vokabulare zur Klassifizierung beschrieben entweder als URL zu einem `skos:Concept` oder als String. Typ: Liste von Strings oder URLs zu `skos:Concept` Objekten. Kardinalität: 0 bis *. OPTIONAL.

5.11 oparl:Consultation (Beratung)

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

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

Beispiel

```
{
    "id": "https://oparl.example.org/consultation/47594",
    "type": "http://oparl.org/schema/1.0/Consultation",
    "paper": "https://oparl.example.org/paper/2396",
    "agendaItem": "https://oparl.example.org/agendaitem/15569",
```

[54] http://www.w3.org/TR/prov-o/#inverse-names
[55] vgl. license

```
    "organization": "https://oparl.example.org/organization/96",
    "authoritative": false,
    "role": "https://oparl.example.org/role/decision"
}
```

5.11.1 Eigenschaften

paper Drucksache, die beraten wird. Typ: URL eines Objekts vom Typ `oparl:Paper`. Kardinalität: 1. ZWINGEND.

agendaItem Tagesordnungspunkt, unter dem die Drucksache beraten wird. Typ: URL eines Objekts vom Typ `oparl:AgendaItem`. Kardinalität: 0 bis 1. EMPFOHLEN.

organization Gremium, dem die Sitzung zugewiesen ist, zu welcher der zuvor genannte Tagesordnungspunkt gehört. Hier kann auch eine mit Liste von Gremien angegeben werden (die verschiedenen `oparl:Body` und `oparl:System` angehören können). Die Liste ist dann geordnet. Das erste Gremium der Liste ist federführend. Typ: `oparl:Organization`. Kardinalität: 1 bis *. ZWINGEND.

authoritative Drückt aus, ob bei dieser Beratung ein Beschluss zu der Drucksache gefasst wird (`true`) wird oder nicht (`false`). Typ: Boolean. Kardinalität: 0 bis 1. OPTIONAL.

role Rolle oder Funktion der Beratung. Zum Beispiel Anhörung, Entscheidung, Kenntnisnahme, Vorberatung usw. Diese Eigenschaft funktioniert wie in Vokabulare zur Klassifizierung beschrieben entweder als URL zu einem `skos:Concept` oder als String. Typ: String oder URL eines `skos:Concept` Objekts. Kardinalität: 0 bis 1. OPTIONAL.

keyword Schlagworte. Diese Eigenschaft funktioniert wie in Vokabulare zur Klassifizierung beschrieben entweder als URL zu einem `skos:Concept` oder als String. Typ: Liste von Strings oder URLs zu `skos:Concept` Objekten. Kardinalität: 0 bis *. OPTIONAL.

5.12 oparl:Location (Ort)

Dieser Objekttyp dient dazu, den Ortsbezug einer Drucksache formal abzubilden. Ortsangaben können sowohl aus Textinformationen bestehen (beispielsweise dem Namen einer Straße/eines Platzes oder eine genaue Adresse) als auch aus Geodaten. Ortsangaben sind auch nicht auf einzelne Positionen beschränkt, sondern können eine Vielzahl von Positionen, Flächen, Strecken etc. abdecken.

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

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

OParl sieht bei Angabe von Geodaten ZWINGEND die Verwendung des Well-Known-Text-Formats (WKT) der Simple Feature Access Spezifikation[56] vor. WKT erlaubt die Beschreibung von unterschiedlichen Geometrien wie Punkten (`Point`), Pfaden (`LineString`), Polygonen (`Polygon`) und viele andere mehr.

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

Für die Ausgabe über eine OParl-API MÜSSEN sämtliche Koordinatenangaben solcher Geodaten im System WGS84[57] angegeben werden, und zwar in Form von Zahlenwerten (Fließkommazahlen) für Längen- und Breitengrad.

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

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

Beispiele

Ein einfaches Objekt mit Punktkordinate:

```
{
    "id": "https://oparl.example.org/locations/29856",
    "type": "http://oparl.org/schema/1.0/Location",
    "description": "Honschaftsstraße 312, Köln",
    "geometry": "POINT (7.03291 50.98249)"
}
```

Ortsangabe mit Polygon-Objekt:

```
{
    "id": "https://oparl.example.org/locations/29856",
    "type": "http://oparl.org/schema/1.0/Location",
    "description": "Rechtes Rheinufer zwischen Deutzer
        Brücke und Hohenzollernbrücke",
    "geometry": "POLYGON ((
                6.9681106 50.9412137,
                6.9690940 50.9412137,
                6.9692169 50.9368270,
                6.9681218 50.9368270,
                6.9681106 50.9412137))"
}
```

5.12.1 Eigenschaften

`description` Textliche Beschreibung eines Orts, z. B. in Form einer Adresse. Typ: String. Kardinalität: 0 bis 1. EMPFOHLEN.

`geometry` Geodaten-Repräsentation des Orts. Ist diese Eigenschaft gesetzt, MUSS ihr Wert der Spezifikation von Well-Known Text (WKT) entsprechen. Typ: String. Kardinalität: 0 bis 1. EMPFOHLEN.

`keyword` Schlagworte mit `skos:prefLabel`. Vgl. dazu Vokabulare zur Klassifizierung. Typ: Array mit Strings oder URLs zu `skos:Concept` Objekten. Kardinalität: 0 bis *. OPTIONAL.

5.13 oparl:Membership

Über Objekte diesen Typs wird die Mitgliedschaft von Personen in Gruppierungen dargestellt. Diese Mitgliedschaften können zeitlich begrenzt sein. Zudem kann abgebildet werden, dass eine Person eine bestimmte Rolle bzw. Position innerhalb der Gruppierung inne hat, beispielsweise den Vorsitz einer Fraktion.

Beispiel 1

```
{
    "id": "https://oparl.example.org/memberships/385",
    "type": "http://oparl.org/schema/1.0/Membership",
    "person": "https://oparl.example.org/people/862",
    "organization": "https://oparl.example.org/organizations/5",
    "role": "https://oparl.example.org/vocab/membership/role/chair",
    "votingRight": true,
    "startDate": "2013-12-03T16:30:00+01:00"
}
```

Beispiel 2


```
{
    "id": "https://oparl.example.org/memberships/693",
    "person": "https://oparl.example.org/people/284",
    "organization": "https://oparl.example.org/organizations/9",
    "role": "Sachkundiger Bürger | Sachkundige Bürgerin",
    "votingRight": false,
    "startDate": "2013-12-03T16:30:00+01:00",
    "endDate": "2014-07-28T00:00:00+02:00"
}
```


5.13.1 Eigenschaften

`person` Die betreffende Person, die Mitglied einer Gruppierung ist oder war. Typ: URL eines `oparl:Person` Objekts. Kardinalität: 1. ZWINGEND.

`organization` Die Gruppierung, in der die Person Mitglied ist oder war. Typ: URL eines `oparl:Organization` Objekts. Kardinalität: 1. ZWINGEND.

`role` Rolle der Person für die Gruppierung. Kann genutzt werden, um verschiedene Arten von Mitgliedschaften zum Beispiel in Gremien zu unterscheiden. Diese Eigenschaft funktioniert wie in Vokabulare zur Klassifizierung beschrieben entweder als URL zu einem skos:Concept oder als String. Der String (oder entsprechend das prefLabel des verlinkten Objekts) SOLL in dieser Form sowohl die männliche als auch die weibliche Rollenbezeichnung enthalten: `"Vorsitzender | Vorsitzende"`. Typ: URL eines `skos:Concept` Objekts oder String. Kardinalität: 0 bis 1. OPTIONAL.

`post` The post held by the person in the organization. Typ: `org:Post`. TODO: Prüfen, ob das ohne JSON-LD Sinn macht, oder ob hier zusätzliche Erkäuterungen notwendig sind. OPTIONAL.

`onBehalfOf` Entsendende Gruppierung, Fraktion, fraktionsloses oder externes Gremium. Es kann auch Mitglieder geben, die von keiner anderen Gruppierung entsendet wurden (z. B. fraktionslose Abgeordnete). Da eine solche Person sich gewissermaßen selbst "entsendet" hat, SOLL in dem Fall hier der selbe Wert angegeben werden wie bei der Eigenschaft `person`.[58] Typ: URL eines `oparl:Organization` oder `oparl:Person` Objekts. Kardinalität: 0 bis 1. OPTIONAL.

`votingRight` Gibt an, ob die Person in der Gruppierung stimmberechtigtes Mitglied ist. Typ: `boolean`. Kardinalität: 0 bis 1. OPTIONAL.

`startDate` Anfangszeitpunkt der Mitgliedschaft.[59] Typ: `xsd:dateTime`. Kardinalität: 0 bis 1. OPTIONAL.

`endDate` Der Endzeitpunkt der Mitgliedschaft.[60] Typ: `xsd:dateTime`. Kardinalität: 0 bis 1. OPTIONAL.

5.14 oparl:LegislativeTerm

Dieser Objekttyp dient der Beschreibung einer Wahlperiode.

Beispiel

[58] Dies entspricht `opengov:onBehalfOf` in Popolo. http://popoloproject.com/specs/membership.html
[59] Abgeleitet von: `schema:validFrom` in Popolo. http://popoloproject.com/specs/membership.html
[60] Abgeleitet von: `schema:validThrough` in Popolo. http://popoloproject.com/specs/membership.html

```
{
    "id": "https://oparl.example.org/term/21",
    "type": "http://oparl.org/schema/1.0/LegislativeTerm",
    "name": "21. Wahlperiode",
    "startDate": "2010-12-03",
    "endDate":  "2013-12-03"
}
```

5.14.1 Eigenschaften

`name` Nutzerfreundliche Bezeichnung der Wahlperiode. Typ: `xsd:string`. Kardinalität: 1. ZWINGEND.

`startDate` Der erste Tag der Wahlperiode. Typ: `xsd:date`. Kardinalität: 0 bis 1. EMPFOHLEN.

`endDate` Der letzte Tag der Wahlperiode. Typ: `xsd:date`. Kardinalität: 0 bis 1. EMPFOHLEN.