



Release Notes i-views 6.1

Inhalt

1. i-views 6.1	1
1.1. Neue Funktionen und Erweiterungen	1
1.1.1. Sprachen-Komponente	1
1.1.2. Knowledge-Builder als Web-Anwendung	1
1.1.3. Aufgaben-Komponente	1
1.1.4. Benutzerdefinierte Komponenten	1
1.1.5. View Configuration und View Config Mapper	2
1.1.6. RDF-Import	4
1.1.7. GQL-Abfragesprache	4
1.1.8. Sonstige Verbesserungen	5
1.1.9. Inkompatibilitäten	7
1.1.10. Entfernte Komponenten	7
1.1.11. Abgekündigte Komponenten	7
2. i-views 6.1.2	8
2.1. Neue Funktionen und Erweiterungen	8
2.1.1. Model-Context-Protocol-Komponente	8
2.1.2. Variantenmodell-Komponente	8
2.1.3. iiRDS-Komponente	9
2.1.4. Benutzerdefinierte Komponenten	9
2.1.5. Sonstige Verbesserungen	9
2.2. Inkompatibilitäten	10
2.2.1. JavaScript-API: Iteratoren als Funktionsargumente	10

1. i-views 6.1

1.1. Neue Funktionen und Erweiterungen

1.1.1. Sprachen-Komponente

Die Sprachen-Komponente enthält Schema und Funktionalität, um Sprachen als Objekte des Knowledge-Graph konfigurieren und nutzen zu können. Diese Komponente ist ab 6.1 fester Bestandteil von i-views und ermöglicht die Auswahl von Sprachen in der Anzeige sowohl im Knowledge-Builder als auch im View Config Mapper. Die bisherige Konfiguration über Attribute wird beim Upgrade auf 6.1 automatisch umgewandelt und funktioniert dann wie zuvor.

1.1.2. Knowledge-Builder als Web-Anwendung

Ein Knowledge-Builder lässt sich jetzt auch als View Configuration Mapper-Webanwendung realisieren. Dabei können zum großen Teil die gewohnten View-Konfigurationselemente und Techniken verwendet werden, die in jeder View Configuration Mapper-Webanwendung zum Einsatz kommen.

Ein fertig konfigurierter Knowledge-Builder als Web-Anwendung (kurz: "Web-KB") ist als vorkonfigurierte benutzerdefinierte Komponente erhältlich. Dieser Web-KB kann in einen beliebigen Knowledge-Graph importiert werden und ist dann sofort nutzbar.

1.1.3. Aufgaben-Komponente

Die Aufgaben-Komponente enthält Schema und Funktionalität, um Aufgaben, die ein Jobclient ausführt, als Objekte im Knowledge-Graph zur Verfügung zu stellen. Auf diese Weise sind die Erstellung neuer Aufgaben und die Übersicht über die Abarbeitung von Aufgaben im konfigurierten Web-Frontend möglich.

1.1.4. Benutzerdefinierte Komponenten

1.1.4.1. Zuweisung

- Registraturobjekte, welche nur als Attributwert existieren und keinen Registrierungsschlüssel haben, werden nun implizit derselben Komponente zugewiesen wie das Element mit dem Attribut.
- Neue Option um Relationen zu markieren, welche nicht zur selben Komponente gezählt werden wie ihre Quelle und somit auch nicht mit dieser exportiert werden.

1.1.4.2. Werkzeuge

- Neues Werkzeug zur Bewertung des aktuellen Zustands der Zuweisungen im Graph und zur Analyse von Problemen wie zyklischen Abhängigkeiten, mehrfach Zuweisung, Schichtenverletzungen und mehr.

- In der Komponentenansicht des Graph-Editors werden
 - unzugewiesene Elemente anders dargestellt als Elemente von Softwarekomponenten.
 - Schichtenverletzungen hervorgehoben und in einem Tooltip erklärt.
- Wenn der Zuweisen-Knopf des Zuweisungswerkzeugs deaktiviert ist, hat er einen Tooltip mit einer Erläuterung.
- Das Zuweisungswerkzeug hat nun eine Option, um alle Unterelemente bis zum ersten Zugewiesenen zu selektieren.
- Das Werkzeug zur Definition von Relationen als Abhängigkeit zeigt nun auch Relationen an, welche von Systemkomponenten als Abhängigkeit definiert sind.

1.1.4.3. Transfer

- Beim Export kann die Version der Komponente angepasst werden.
- Ordner exportieren nun auch die Information, in welchen anderen Ordnern sie sich befinden.
- Relationen, welche zu einer Voraussetzung führen, werden beim Export immer als Referenz exportiert und beim Import immer überschrieben. Relationen, die zu einem abhängigen Element führen, werden weder exportiert noch überschrieben.

1.1.4.4. Sonstiges

- Option an Komponenten um alle neuen Objekte von zugewiesenen Typen im selben Cluster zu speichern.
- Das Attribut 'Elemente anhand Präfix/URI/Relation auswählen' wurde durch das Attribut 'Elemente ausschließlich anhand der Auswahlobjekte auswählen' ersetzt.

1.1.5. View Configuration und View Config Mapper

1.1.5.1. Erscheinungsbild & Theming

- Statt numerischer Werte werden Breiten- und Höhenangaben jetzt als Zeichenketten konfiguriert, sodass neben Pixelangaben auch Einheiten wie **em**, **vw** oder Schlüsselwörter wie **auto** und **fit-content** verwendet werden können.
- Das Standard-Theme (**vcmlight**) kann über die **themes**-Options-Konfiguration überschrieben werden. Durch Angabe eines Themes mit dem Namen **vcmlight** und einer eigenen Farb- oder Variablendefinition lässt sich das Erscheinungsbild der gesamten Anwendung anpassen, ohne Custom CSS zu benötigen.
- Einzelne Panels können jetzt ein eigenes Vuetify-Theme erhalten, das über die Backend-Konfiguration definiert wird. Dadurch können verschiedene Panels in einer Anwendung optisch voneinander abgegrenzt werden.
- Menüs (Toolbars, Menüleisten, Listen-Menüs) und Aktionen unterstützen jetzt eine **appearance**-Konfiguration im **Style.extra**-Feld. Über **density**, **size**, **variant** und **color** kann das Erscheinungsbild der Aktionen einheitlich gesteuert werden.

- Zwei neue Templates für Dialoge erleichtern das Erstellen von Standarddialogen mit Kopf- und Fußzeilen.
- Der Abstand zwischen verschachtelten Panels wurde entfernt. Nur Content-Panels erhalten weiterhin ein Innen-Padding, was es besser ermöglicht, Panels auf die volle Browser-Höhe zu stylen und mit eigenen Scrollbars auszustatten.

1.1.5.2. Eigenschaften & Formularfelder

- Im Lese- und Bearbeitungsmodus kann konfiguriert werden, welche Sprachen angezeigt werden sollen und ob die Übersetzungsliste aus- oder eingeklappt dargestellt wird. In diesem Zusammenhang wurde auch die Darstellung im Frontend verbessert.
- Bei Eingabefeldern mit mehreren Teilkomponenten (z.B. FlexTime, Intervalle) wird das automatische Speichern (Autosave) jetzt zuverlässiger ausgelöst.
- Alle Boolean-Eingabe- und Anzeigekomponenten (Checkbox, Switch, Formular) erhalten jetzt eine CSS-Klasse, die den aktuellen Zustand (`no-value`, `true`, `false`) widerspiegelt. Das erleichtert gezieltes Styling über Custom CSS.
- Die Anzeige von Relationszielen in Bearbeitungsfeldern wurde überarbeitet und ist nun besser lesbarer und weniger visuell aufdringlich.
- Es gibt jetzt ein neues Formularfeld für den Upload von Binärdateien (Blobs). Damit können Nutzer über Formulare direkt Dateien hochladen.

1.1.5.3. Tabelle

- Für Tabellen wurden drei neue CSS-Hilfsklassen eingeführt, mit denen der Inhalt von Tabellenzellen ausgerichtet werden kann: `vcm-table-cell-position-start`, `vcm-table-cell-position-center` und `vcm-table-cell-position-end`. Die Klassen können auf die gesamte Tabelle, einzelne Zeilen, einzelne Spalten oder eine beliebige Kombination davon gesetzt werden.

1.1.5.4. Net-Navigator

- Der Dialog zum Auswählen von Knoten beim Erweitern eines Graphknotens wird nun als modaler Dialog statt als Dropdown-Menü angezeigt. Knoten werden nach Relationen gruppiert, alphabetisch sortiert und können über ein Suchfeld gefiltert werden.

1.1.5.5. Integration

- Die Anwendung wird jetzt als internes npm-Paket `@i-views/vcm-vue` veröffentlicht und kann damit als Bibliothek in Anpassungsprojekte eingebunden werden.
- Stores, Facades und TypeScript-Typen sind jetzt über den Standard-Import der Bibliothek zugänglich, was die Integration in externe Anwendungen vereinfacht.
- Mit `npm init @i-views/vcm-vue@latest` steht ein interaktives CLI-Tool bereit, das ein vollständig vorkonfiguriertes View-Config-Mapper-Anpassungsprojekt erzeugt.

1.1.6. RDF-Import

Der RDF-Import unterstützt nun auch die Turtle-Syntax inklusive Quoted Triples (Teil von RDF-star).

1.1.7. GQL-Abfragesprache

Mit i-views 6.1 steht eine neue Abfragesprache für den Knowledge-Graph zur Verfügung: die Graph Query Language (GQL). GQL ist ein internationaler ISO-Standard zur Abfrage von Graphdatenbanken und an die weit verbreitete Sprache Cypher angelehnt.

Im Unterschied zur grafisch aufgebauten Strukturabfrage werden GQL-Abfragen als Text formuliert. Das Ergebnis einer GQL-Abfrage ist stets eine Tabelle — vergleichbar mit einer Excel-Tabelle oder dem Ergebnis einer SQL-Abfrage. Damit eignet sich GQL besonders gut für Auswertungen, Berichte und den Datenexport. Als Textformat ist GQL darüber hinaus gut geeignet, um den Knowledge-Graph aus externen Systemen heraus abzufragen — darunter auch KI-Agenten, die Abfragen eigenständig formulieren und auswerten können.

1.1.7.1. GQL-Features und Alleinstellungsmerkmale

GQL-Abfragen beschreiben Muster im Graphen, liefern aber im Gegensatz zur Strukturabfrage stets eine strukturierte Ergebnistabelle statt einer unsortierten Menge von Objekten. Jede Spalte der Tabelle entspricht einem explizit benannten Rückgabefeld — zum Beispiel direkt ein Attributwert, ein berechneter Ausdruck oder ein referenziertes Objekt. Das eröffnet Möglichkeiten, die mit Strukturabfragen nicht oder nur umständlich realisierbar sind:

- Aggregation und Gruppierung — Funktionen wie **COUNT**, **SUM**, **AVG**, **MIN** und **MAX** fassen Werte über mehrere Zeilen zusammen; mit **GROUP BY** erfolgt die Auswertung je Gruppe, zum Beispiel die Anzahl der Produkte je Hersteller.
- Boolesche Ausdrücke — Bedingungen können mit **AND**, **OR** und **NOT** frei kombiniert werden.
- Arithmetik und Wertverarbeitung — in Rückgabe und Bedingungen stehen mathematische Operatoren sowie String-, Datums- und Listenfunktionen zur Verfügung, zum Beispiel zur Berechnung eines Rabattpreises direkt in der Abfrage.
- Sortierung — Ergebnisse können nach beliebigen Feldern auf- oder absteigend sortiert werden, auch nach Feldern, die nicht Teil der Ausgabe sind.
- Seitenweise Ausgabe — mit **LIMIT** und **OFFSET** lassen sich Ergebnismengen aufteilen, was für Listenansichten mit Blättern-Funktion genutzt werden kann.
- Pfade variabler Länge — Kanten können mit Mengenangaben versehen werden, um Pfade über eine variable Anzahl von Hops zu suchen, zum Beispiel "alle Personen, die über zwei bis vier Bekanntschaftsstufen erreichbar sind".
- Mengenoperationen — mehrere Teilabfragen lassen sich per **UNION**, **INTERSECT** oder **EXCEPT** kombinieren.
- Verschachtelte Abfragen — andere gespeicherte Suchen (GQL, Strukturabfragen, Volltextsuchen) können als Unterprozeduren aufgerufen und deren Ergebnisse weiterverarbeitet werden.

1.1.7.2. i-views-spezifische Erweiterungen

Die GQL-Implementierung in i-views berücksichtigt die Besonderheiten des Knowledge-Graph-Datenmodells:

- Mehrsprachige Attributwerte werden automatisch in der passenden Anwendungssprache ausgegeben. Mit Sprachqualifikatoren lassen sich auch gezielt bestimmte Übersetzungen abfragen.
- Mehrwertige Attribute werden analog zu mehrwertigen Relationen in separate Ergebniszeilen aufgeteilt.
- Alle Knowledge-Graph-Elemente — also auch Relationen und Attribute — können als Knoten im Abfragemuster auftreten, sodass Metaeigenschaften auf natürliche Weise zugänglich sind.
- Neben den GQL-Standardoperatoren steht die Funktion **IV_COMPARE** für erweiterte Vergleiche zur Verfügung, darunter Platzhaltersuchen, reguläre Ausdrücke und geodätische Distanzvergleiche.
- Systemrelationen sind über vordefinierte Labels (Präfix **iv_**) direkt in Abfragen nutzbar.

1.1.7.3. Ausführung und Integration

GQL-Abfragen können auf verschiedenen Wegen ausgeführt werden:

- GQL Workbench — interaktive Oberfläche im Knowledge Builder unter *Werkzeuge > GQL Workbench* zum Entwickeln und Testen von Abfragen
- Registrierte Suchen — GQL-Abfragen können wie andere Sucharten als Ordnerelement oder registrierte Suche gespeichert werden
- REST-API — über eine konfigurierbare Built-In-Resource lassen sich Abfragen per HTTP aufrufen; das Ergebnis kann als JSON, CSV oder Excel ausgegeben werden
- JavaScript-API — über **\$k.GQLQuery** können GQL-Abfragen programmatisch ausgeführt und die Ergebnistabelle zeilenweise ausgewertet werden

1.1.8. Sonstige Verbesserungen

- Die neue Systemrelation "hat Ziel" ermöglicht die Traversierung von Relation zu Relationsziel.

1.1.8.1. JavaScript-API

- In der **JavaScript API** wurden mit **experimental** gekennzeichnete Funktionen überarbeitet
 - Viele Funktionen werden jetzt offiziell unterstützt
 - Funktionen ohne bekannte Nutzung in Projekten wurden entfernt
 - Problematische Funktionen wurden mit **deprecated** markiert
- **Type.topType()** : Returns the top type of the user-definable part of the knowledge graph

1.1.8.2. Verbesserungen im Knowledge Builder

- Die Banner der Detailansichten von Graph-Elementen wurden vereinheitlicht.



- Links folgt auf das Icon des Elements seine Art (zeigt bei Objekten den Typ und Ergänzungen an), das Menü zur Interaktion mit dem Element und der Name des Elements.
- Auf der rechten Seite werden zusätzliche Informationen wie zum Beispiel die aktuelle Komponentenzuweisung angezeigt.
- Der Kontrast für einige Farben wurde verbessert.
- Tooltips werden so lange angezeigt, wie der Cursor über dem dazugehörigen Element bleibt.
- Strg + Klick auf einen Pfeil in einer Baumansicht klappt den gesamten Ast aus.
- Knöpfe mit Tastatur-Fokus sind hervorgehoben.
- Die Hervorhebung selektierter Reiter (Tabs) ist jetzt deutlich kontrastreicher und besser erkennbar.
- Auf der Einstellungsseite für Hotkeys ist nun auch eine Liste mit statischen Hotkeys.
- Im Graph Editor können nun die IDs von Relationen kopiert werden.
- Im Graph Editor können die Menge der selektierten Knoten und die Menge der unselektierten Knoten vertauscht werden.
- Neues Standard-Icon für Ergänzungstypen.
- Der Verwendungen-Dialog zeigt nun auch Verwendungen als Domäne und Obertyp an.
- Die Entscheider in der Rechte-Konfiguration werden beim Hinzufügen eingefärbt, um sie besser voneinander unterscheiden zu können.
- Die Einträge verschiedener Kontext-Menüs wurden überarbeitet, um dem Anwender eine konsistentere Erfahrung zu bieten.
- Bei der Abbildung von Datenquellen werden Warnungen angezeigt, wenn Elemente der Datenquelle nicht abgebildet wurden. Um den Mapping-Baum nicht zu überfrachten, wird eine separate Liste aller Warnungen angeboten.
- Relationsziele ohne Namen werden als <Objekt vom Typ xy> dargestellt statt <ohne Name>.
- Der Text von Dialogen mit Fehlermeldungen ist kopierbar, um Fehlermeldungen einfacher weiterleiten zu können.
- Neue Abfragen oder Skripte können ad-hoc aus dem JavaScript-Editor erstellt werden, wenn man eine Referenz auf eine bisher nicht vorhandene Abfrage bzw. Skript einfügt.
- Tabelleninhalte werden automatisch aktualisiert, wenn man eine Suche auslöst. Zuvor war dazu ein zusätzlicher Knopfdruck "Aktualisieren" erforderlich.
- Die Umrechnung in alle Einheiten derselben Größe wird in der Ansicht „Einstellung der Einheiten“ angezeigt.

- "Verwendungen anzeigen" bei Rechte-Operatoren
- Einstellungen für den Code-Editor (Zeilennummern anzeigen, Hell-/Dunkelmodus)

1.1.8.3. Verbesserungen der Strukturabfrage

- Zahlreiche Verbesserungen zur Stabilität und Performance
- Index-Warnungen zur Strukturabfrage enthalten nun Indexierungshinweise und basieren auf den tiefgreifenden Analysen des Index-Reports. Die Berechnung der Hinweise dauert zwar länger, ist aber präziser als die bisherigen Warnungen. Um die Anzeige einer Strukturabfrage nicht unnötig zu verzögern, können die Indexierungshinweise so konfiguriert werden, dass sie erst auf explizite manuelle Anforderung berechnet werden.

1.1.9. Inkompatibilitäten

Beim Upgrade werden Konfigurationen soweit wie möglich automatisch angepasst. In einigen Fällen ist das nicht möglich:

1.1.9.1. Relationsziel oder -quelle ändern

Operationen, die Quelle bzw. Ziel einer Relation ändern, wurden vor 6.1 gesondert behandelt. Die unveränderte Seite der Relation wurde nicht von Zugriffsrechten und Triggern erfasst.

Ab 6.1 wird die vollständige Relation als gelöscht und neu erzeugt behandelt. Auf diese Weise lassen sich vor allem Trigger konsistent und ohne Sonderbehandlung formulieren.

Das bisherige Verhalten kann jedoch manuell wiederhergestellt werden, indem man mittels eines Skript-Filters auf das "relocate"-Flag im access-Objekt prüft. Filtert man accesses mit "relocate"-Flag heraus, dann ergibt sich das bisherige Verhalten.

```
function accessMatch(access) {  
  return (access.relocate != true)  
}
```

1.1.9.2. Timeouts

HTTP- und SMTP-Verbindungen haben keine inhärenten Timeouts mehr. Werden Timeouts gewünscht, dann müssen diese konfiguriert werden (bei REST-Services über die entsprechende Ressource). Vor allem beim Betrieb eines Web-Frontends über den View Config Mapper ist dies zu beachten.

1.1.10. Entfernte Komponenten

1.1.11. Abgekündigte Komponenten

Folgende Bestandteile der Software sind abgekündigt und werden ggf. in zukünftigen Releases

entfernt.

- VCM-material: Frontends, die auf material basieren und entweder individuelle Styling Anpassungen oder Views verwenden müssen mit dem Umstieg auf Vue angepasst werden. Wird in Version 6.2 entfernt.
- Lucene Volltextindex: Die Anbindung von Lucene soll durch die vorhandene Elastic Anbindung ersetzt werden. Diese ist einfacher zu skalieren und benötigt nicht mehr die technisch komplexe Verbindung mit der Java VM.
- ViewConfig Rendering in JavaScript: die Funktionen zum Erzeugen von Pseudo-Tabellen als JavaScript Objekte benutzt eine alte interne Schnittstelle, die in der Kommunikation des Backends mit dem Frontend eingesetzt wurde. Daher muss das Ergebnis oft umständlich noch nachbearbeitet werden im JavaScript Code.
- KScript: Diese Skript-Variante kann schon länger nicht mehr angelegt werden, wird aber bei der Migration von alten Versionen noch unterstützt. Prinzipiell ist JavaScript der designierte Nachfolger. Da es bisher aber keinen automatischen Übersetzer von KScript nach JavaScript gibt, muss in Projekte der Umstieg auf JavaScript manuell erfolgen.
- KPath: als Teilsprache im KScript Umfeld entstanden, kann es heute nur noch in Such-Pipelines verwendet werden. Prinzipiell ist ein Ersatz durch JavaScript möglich, aber noch nicht umgesetzt.

2. i-views 6.1.2

2.1. Neue Funktionen und Erweiterungen

2.1.1. Model-Context-Protocol-Komponente

Das [Model Context Protocol](#) (MCP) ist ein offener Standard, der KI-Agenten und Werkzeugen für große Sprachmodelle (LLMs) die Interaktion mit externen Systemen über eine klar definierte Schnittstelle ermöglicht. Die neue Model-Context-Protocol-Komponente fungiert als MCP-Server und stellt Daten und Operationen des Knowledge-Graphen als Tools bereit, die KI-Agenten entdecken und aufrufen können.

Tools sind nicht vordefiniert — stattdessen konfigurieren Sie graphspezifische Tools im Knowledge Builder unter **TECHNIK > Model Context Protocol**. Jedes Tool kapselt eine bestimmte Funktionalität des Knowledge-Graphen — eine Abfrage, einen Export, einen Import, ein Skript oder eine View-Konfiguration — und macht sie für jeden MCP-kompatiblen KI-Client verfügbar. Über einen konfigurierbaren MCP-Service werden die Tools bereitgestellt und können von gängigen KI-Agenten-Frameworks und Entwicklungswerkzeugen direkt eingebunden werden.

2.1.2. Variantenmodell-Komponente

Die neue Variantenmodell-Komponente enthält Schema und Funktionalität, um Objekte als Varianten anderer Objekte zu modellieren. Eine Variante eines Objekts "erbt" alle Eigenschaften

dieses Objekts, die sie nicht selbst überschreibt. Die geerbten Eigenschaften werden über spezielle Systemrelationen sowie Skriptfunktionen zur Verfügung gestellt.

Der ergänzende Typ *Tabulation* ermöglicht zusätzlich die Erstellung tabellarischer Übersichten über Variantenausprägungen. Eine Tabulation listet alle möglichen konkreten Varianten eines Variantenbaums als Zeilen einer Tabelle, wobei jede Spalte eine konfigurierte Eigenschaft darstellt.

2.1.3. iiRDS-Komponente

Die neue iiRDS-Komponente stellt den Rahmen bereit, um iiRDS-Pakete in den Knowledge-Graphen zu importieren.

2.1.4. Benutzerdefinierte Komponenten

- Der Schreibschutz von Komponenten ist stärker und erstreckt sich über mehr Bereiche im Knowledge Builder.
- Import und Export zeigen nun ihre aktuelle Geschwindigkeit und bereits vergangene Zeit an.
- Neue Relation für *Auswahl von semantischen Elementen* Objekte, um Typen zu spezifizieren, deren Elemente dann zur Komponente zugewiesen sind, ohne dass der Typ selbst derselben Komponente angehören muss.
- View-Konfigurationsbäume im Knowledge Builder können nach Komponentenzuweisung ihrer Elemente eingefärbt werden.
- Die Baumansicht von View-Konfigurationselementen im KB hat nun auch direkt ein Untermenü für benutzerdefinierte Komponenten, welches zusätzliche Optionen zum Zuweisen ganzer Abschnitte der View-Konfiguration enthält.

2.1.5. Sonstige Verbesserungen

- Die neue Systemrelation *Identität* verbindet jedes Knowledge-Graph-Element mit sich selbst. Sie ist überall dort nützlich, wo ein Werkzeug eine konfigurierte Relation zur Strukturierung von Ergebnissen erwartet — beispielsweise kann ein JSON-Export diese Relation nutzen, um verschachtelte JSON-Strukturen zu erzeugen, die auf einem einzelnen Graph-Element basieren.
- In der GQL-Abfragesprache steht nun die Erweiterungsfunktion `IV_NAME` zur Verfügung, mit der der Name eines Knotens direkt im Abfrageausdruck abgefragt werden kann.
- In der JavaScript-API steht nun die Methode `$k.GQLQueryResult.prototype.toArray()` zur Verfügung, um GQL-Ergebnisse bequem als Array weiterzuverarbeiten.
- Als neue Konverter-Option für LaTeX-Dokumente unterstützt die Dokumenterstellungskomponente nun auch die Anbindung an einen dedizierten TeX-Live-Konvertierungsdienst.
- Konfigurierbare REST-Services unterstützen nun OpenID Connect als Authentifizierungsverfahren.
- Ergänzungstypen können nun verkettet werden: Das Schema, das ein Objekt durch einen Ergänzungstyp erhält, kann seinerseits Ergänzungsrelationen zu weiteren Ergänzungstypen enthalten.

2.2. Inkompatibilitäten

2.2.1. JavaScript-API: Iteratoren als Funktionsargumente

Die meisten JavaScript-API-Funktionen, die bisher ein Array als Argument erwartet haben, akzeptieren nun auch Iteratoren und iterierbare Objekte. Im Gegenzug führt die Übergabe von Objekten, die weder Array noch Iterator sind und bisher stillschweigend als leeres Array behandelt wurden, nun zu einem Fehler (`'... is not iterable'`). Skripte, die solche Argumente übergeben, müssen angepasst werden.