

Dr. Felix Bach, Daniel Fähle, Sandra Göller, Timo Holste, Jan Schweikert, Sarah Rebecca Ondraszek

Blue Paper: Technische Spezifikationen für den NFDI4Memory Data Space

Version 1.0, Stand: 04.08.2025

Autor:innenliste:

Dr. Felix Bach | [Project administration](#), [Supervision](#), [Writing – review & editing](#) | ORCID: [0000-0002-5035-7978](#)

Daniel Fähle | [Writing – review & editing](#) | ORCID: [0000-0002-2532-7873](#)

Sandra Göller | [Project administration](#), [Conceptualization](#), [Writing – original draft](#), [Writing – review & editing](#) | ORCID: [0000-0003-4553-3671](#)

Timo Holste | [Project administration](#), [Writing – review & editing](#) | ORCID: [0009-0007-4737-4973](#)

Jan Schweikert | [Conceptualization](#), [Writing – original draft](#) | ORCID: [0000-0003-4774-2717](#)

Sarah Rebecca Ondraszek | [Writing – review & editing](#) | ORCID: [0009-0003-7945-6704](#)

im Namen des 4Memory-Konsortiums.

Zusammenfassung

In diesem Blue Paper werden das Konzept und das technische Grundgerüst des NFDI4Memory Data Space beschrieben. Der NFDI4Memory Data Space wird als digitales Ökosystem aus Datenquellen und Diensten die zentrale Infrastruktur des Konsortiums bilden und es Wissenschaftler:innen ermöglichen, über Fach- und Repositoriengrenzen hinweg Quellen und zugehörige Daten zu finden und mit ihnen sowohl auf herkömmliche Weise als auch mit Methoden der „Digital Humanities“ zu arbeiten. Technisch handelt es sich um ein Framework aus Application Programming Interfaces (APIs) und einem Knowledge Graph auf Basis der NFDI4Memory Ontology MemO. Der Fokus des Blue Papers liegt auf der technischen Spezifikation der verschiedenen APIs – etwa für Suche, Download und Konvertierung bzw. Transformation von Daten. Die Zielgruppe dieses Blue Papers sind Betreibende von Datenquellen und Key Services der NFDI4Memory-Community, welche sich an den NFDI4Memory Data Space anbinden und ihre Daten und Services dort auffindbar und (nach-)nutzbar machen möchten.

Danksagung

Diese Arbeit ist im Rahmen des NFDI-Konsortiums 4Memory entstanden (www.4memory.de). Wir danken der Deutschen Forschungsgemeinschaft (DFG) für die finanzielle Unterstützung – Projektnummer 50 1609550.

Inhaltsverzeichnis

Einleitung	1
Problemstellung	2
Ziele	2
Datenstruktur	4
Der NFDI4Memory Data Space	7
NFDI4Memory Ontology (MemO) und Knowledge Graph	9
Schnittstellen	10
Core-Schnittstellen	11
Query API & SPARQL	11
Update API	11
Access API	13
Service-Schnittstellen	14
OCR API	14
Document Conversion API	15
Schema Conversion API	15
Spezifikationen	16
Query API (v1.0)	16
Update API (v1.0)	18
Access API (v1.0)	20
Schema Converter API (v1.0)	21
OCR API (v1.0)	23
Document Converter API (v1.0)	25
Anmerkungen	27
Zu den Nutzenden des NFDI4Memory Data Space	27
Zu Sicherheitsaspekten des NFDI4Memory Data Space	27
Anhang	28
Beispiel Identify-Antwort in der OAI-PMH Update API	28
API Advertisement Schema (in XSD)	29

Einleitung

Der NFDI4Memory Data Space, welcher von Task Area 3 „Data Services“¹ des Konsortiums NFDI4Memory entwickelt und implementiert wird², bündelt heterogene Datenquellen und sog. Key Services³ von Forschungs- und Gedächtnisorganisationen sowie Informationsinfrastruktureinrichtungen aus den Reihen der NFDI4Memory-Co-Applicants und -Participants. Ziel ist eine aggregierte und übergreifende Suche und Nachnutzung von Daten⁴, welche für die historisch arbeitenden Geisteswissenschaften relevant sind. Im Vorfeld der Antragstellung fand ein intensiver Austausch mit der Community statt, im Zuge dessen ein Set aus zentralen Ressourcen zusammengestellt wurde, das den Kern des Data-Space-Portfolios bildet. Zu den zu integrierenden Datenquellen zählen neben Aggregatoren⁵ die Datenbestände aus Fachdatenbanken bzw. Forschungsdatenrepositorien⁶ und Informationsdienste für die historische Forschung⁷. Darüber hinaus werden Key Services über den NFDI4Memory Data Space auffindbar und nachnutzbar gemacht: Dazu zählen Dienste für die zuverlässige Speicherung und Langzeitarchivierung von Forschungsdaten⁸, Normdatenserver mit entsprechenden Vokabularen und Normdateien⁹, Dienste zur dauerhaften Identifizierung von Ressourcen¹⁰ sowie Werkzeuge und Technologien zur Volltexterkennung¹¹ und zur Georeferenzierung¹². Der NFDI4Memory Data Space ermöglicht Forschenden mithilfe von einheitlich definierten Schnittstellen und einer Wissensbasis in Form eines Knowledge Graphs (KG), über verschiedene Datenquellen hinweg zu suchen und individuelle Korpora aus unterschiedlichen Datenquellen zusammenzustellen. Im Folgenden werden wir¹³ das Konzept und das technische Grundgerüst des NFDI4Memory Data Space, einem Framework aus

1 <https://4memory.de/ueber-4memory/task-areas/data-services/> (aufgerufen am 30.07.2025).

2 Zur Einbettung des NFDI4Memory Data Space in das Konsortiums siehe: Holste, Timo und Razum, Matthias. "Der NFDI4Memory Data Space als digitales Ökosystem für die historisch arbeitenden Wissenschaften, Gedächtnisinstitutionen und Informationsinfrastrukturen" ABI Technik 44, no. 4 (2024): 259-267. <https://doi.org/10.1515/abitech-2024-0039> (aufgerufen am 30.07.2025).

3 Siehe die Listen im Abschnitt zur Task Area „Data Services“ im Projektantrag: Paulmann, Johannes, John Wood, Klaus Ceynowa u. a. „NFDI4Memory. Consortium for the historically oriented humanities. Proposal for the National Research Data Infrastructure (NFDI).“ zenodo (2022) , 77-82, <https://doi.org/10.5281/zenodo.7428489> (aufgerufen am 30.07.2025).

4 Paulmann u. a. 2022: 37-39.

5 Deutsche Digitale Bibliothek (DDB) und Archivportal-D (APD)

6 Factgrid, Handschriftenportal, Kalliope, Laudatio Oral-History.Digital, eAQUA-Portal, K10plus, EHRI Portal, CrossAsia.

7 H-Soz-Kult, Clio-online, Historicum.net.

8 DIMAG, RADAR oder ViDa.

9 DANTE, Factgrid, GND, oder xTree.

10 URN-Dienst oder DOI.

11 OCR-D, Transkribus.

12 Virtuelles Kartenforum oder Kartenspeicher.

13 Wir möchten den Kolleg:innen aus den Participant- und Co-Applicant-Einrichtungen für Ihre wertvollen Rückmeldungen im Rahmen des Reviewprozesses danken, die bereits in diese Version des Blue Papers eingeflossen bzw. für zukünftige Iterationen vorgemerkt sind.

Application Programming Interfaces (APIs) und einem Knowledge Graph¹⁴ (KG) auf Basis der NFDI4Memory Ontology MemO¹⁵, beschreiben. Der Fokus liegt hierbei auf der technischen Spezifikation der verschiedenen APIs – etwa für Suche, Download und Konvertierung bzw. Transformation von Daten. Die Zielgruppe dieses Blue Papers sind Betreibende von Datenquellen und Key Services der NFDI4Memory-Community¹⁶, welche sich an den NFDI4Memory Data Space anbinden und ihre Daten und Services dort auffindbar und (nach-)nutzbar machen möchten. Neue Versionen des Blue Papers sind im Rahmen der iterativen Entwicklung des NFDI4Memory Data Space geplant. Hierbei wird das technische Konzept der enthaltenen Schnittstellen möglichst stabil bleiben, Aktualisierungen werden in den jeweiligen neuen Versionen beschrieben und verfügbar gemacht.

Problemstellung

Bei der Suche nach Daten müssen Historiker:innen meist in zahlreichen Datenquellen recherchieren und sich die für ihr Projekt notwendigen Datensammlungen mühsam aus diversen Ressourcen zusammenstellen. Unter Einbindung umfassender digitaler Portale, wie z.B. der Deutschen Digitalen Bibliothek (DDB) oder dem Handschriftenportal, ermöglicht der NFDI4Memory Data Space mit einem zentralen Sucheinstieg eine aggregierte thematische Suche über Fach- und Repositoriengrenzen hinweg. Der NFDI4Memory Knowledge Graph dient dabei als zentrale Wissensbasis. Über einheitlich definierte Schnittstellen bietet der NFDI4Memory Data Space einen direkten Zugriff zu den Ressourcen und ihren sammlungsspezifischen Metadaten sowie zu anderen Werkzeugen und Diensten für die Bearbeitung und Nachnutzung von Daten wie z.B. Text-Extraktion mittels Optical Character Recognition-Technologien (OCR) sowie Konvertierung von Schemata oder Formaten. Dabei will der NFDI4Memory Data Space die beteiligten Einrichtungen bzw. Key Services und Datenquellen nicht ersetzen oder in den Hintergrund drängen. Vielmehr will er Sichtbarkeit schaffen und den Zugang der Wissenschaft zu Daten und Diensten im Sinne eines Vermittlers verbessern.

Ziele

Der NFDI4Memory Data Space verfolgt drei Ziele:

1. Einheitliche, niedrighschwellige und übergreifende Suche in für die historisch arbeitenden Geisteswissenschaften relevanten Ressourcen.¹⁷

¹⁴ <https://nfdi.fiz-karlsruhe.de/4memory/> (aufgerufen am 30.07.2025).

¹⁵ Ondraszek, Sarah, Oleksandra Bruns, Tabea Tietz, Etienne Posthumus, Jörg Waitelonis, Harald Sack. „MemO v.1.0.0, Module Release: 2025-01-15“, <https://nfdi.fiz-karlsruhe.de/4memory/ontology/> (aufgerufen am 30.07.2025).

¹⁶ Paulmann u. a. 2022: 82.

¹⁷ Ausgehend von den im Antrag genannten Key Services und Datenquellen, Paulmann u. a. 2022: 8-12.53-54. 77-79.

2. Einheitlicher und einfacher Zugriff auf Metadaten und digitale Ressourcen (im Antrag „Datenquellen“ genannt).
3. Einheitlicher und unkomplizierter Zugriff auf häufig benötigte Dienste (im Antrag „Key Services“ genannt).

Der NFDI4Memory Data Space ist als übergeordnete, föderierte Struktur zu verstehen, die über den Knowledge Graph eine aggregierte Suche nach Daten sowie den einheitlichen Zugriff auf diese anbietet. Das API-Framework legt eine weitere Abstraktionsebene über die nach wie vor heterogenen und verteilt vorliegenden Datenbestände und ermöglicht damit auch automatisierte und vereinheitlichte Zugriffe. Dabei sollen nicht nur Daten und Datenquellen, sondern auch relevante Dienste gefunden werden, die für die Bearbeitung der Daten genutzt werden können. Essentiell sind dafür die Beteiligung der NFDI4Memory Participants bzw. der Dienste und Datenquellen am NFDI4Memory Data Space sowie die Bereitschaft, Schnittstellen zu implementieren und Daten für den NFDI4Memory Data Space bereitzustellen.

Zielgruppe des NFDI4Memory Data Space sind Forscher:innen der historisch arbeitenden Geisteswissenschaften, welche Ressourcen nicht nur finden, sondern mithilfe von Diensten für eigene Forschung nutzen, „weiterverarbeiten“ und dabei entstandene Ergebnisse ggf. wieder als FAIRe Forschungsdaten bereitstellen wollen.¹⁸

Um mit dem NFDI4Memory Data Space einen innovativen Mehrwert für die historische Forschung zu schaffen, müssen zunächst die Key Services und Datenquellen der NFDI4Memory Participants sowie deren Daten in den Data Space integriert werden, d.h. es müssen entsprechende deskriptive Metadaten in den NFDI4Memory Knowledge Graph aufgenommen und dort auffindbar gemacht werden. Nach der Integration dieser im Antrag genannten Key Services und Datenquellen wird es entscheidend sein, dass auch weitere Forschende und Einrichtungen mit Datenbeständen bzw. Diensten außerhalb des Participants-Kreises es für sinnvoll erachten, diese im NFDI4Memory Data Space einzubringen. Um dies zu ermöglichen, wird die Task Area „Data Services“ in der zweiten Hälfte der Projektlaufzeit beginnen, einen technischen Beratungsdienst einzurichten, der bei der Integration zusätzlicher Ressourcen in den NFDI4Memory Data Space unterstützt.

¹⁸ Für Überlegungen dazu, wie ein Werkzeugkasten für Analyse, Anreicherung und Auswertungen von primären Forschungsdaten aussehen könnte, siehe Fähle, Daniel, Harald Sack. „Ein „digitaler Werkzeugkasten“ für historische Forschung mit Archivgut. Status quo und Perspektiven“. In Deuten und streiten, suchen und finden. Neue Möglichkeiten der Kooperation zwischen Archiven und Geschichtswissenschaft beim Aufbau digitaler Infrastrukturen. Hrsg. v. Rainer Hering, Gerald Maier. 29–45. Stuttgart: Jan Thorbecke Verlag 2023.

Datenstruktur

Voraussetzung für die Aufnahme von Metadaten zu Daten und Datenquellen in den NFDI4Memory Knowledge Graph ist die Einhaltung minimaler Anforderungen zur Struktur der Daten, die eine effektive Suche über den NFDI4Memory Data Space sicherstellen.

Daten, welche zum Datenbestand einer Datenquelle gehören, müssen im einfachsten Fall aus Metadaten und einem eindeutigen Identifier (ID) bestehen. Der Identifier muss innerhalb einer Datenquelle persistent und eindeutig sein.¹⁹ Diese Form der Datenstruktur bezeichnen wir hier als Informationsobjekt (Abb. 1).

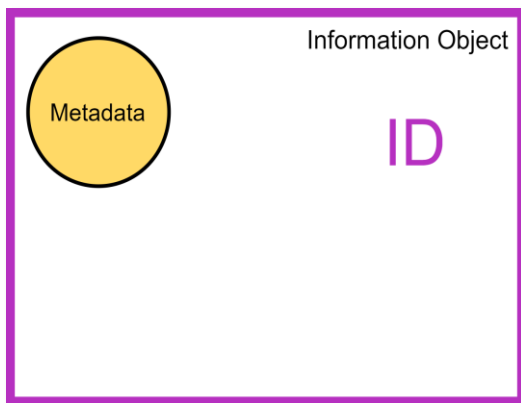


Abb. 1: Informationsobjekt ohne Ressource mit einem Metadatenobjekt.

Meist sind Informationsobjekte in Datenbeständen schon mit einem Identifier ausgezeichnet. Die Metadaten – oder hier Metadatenobjekte – beschreiben eine Ressource, welche ein physisches Objekt (Abb. 2) oder ein digital abrufbares Objekt (Abb. 3) sein kann.

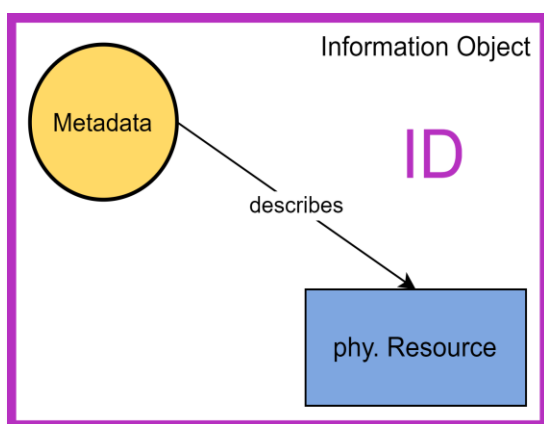


Abb. 2: Informationsobjekt mit physischer, nicht digital abrufbarer Ressource und beschreibendem Metadatenobjekt.

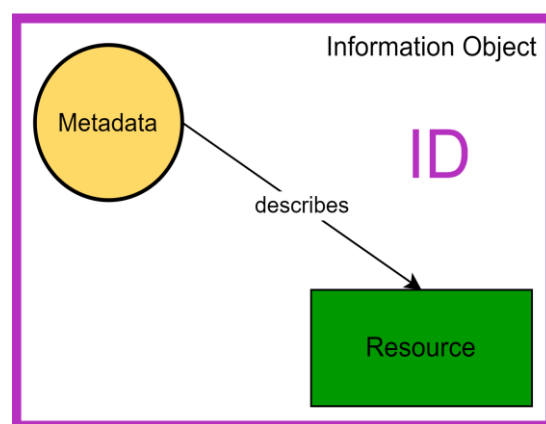


Abb. 3: Informationsobjekt mit digitaler, abrufbarer Ressource und beschreibendem Metadatenobjekt.

¹⁹ Der Identifier eines Informationsobjektes muss keine globale Gültigkeit besitzen. Wichtig ist, dass er sich innerhalb der Datenquelle nicht ändert, da die Informationsobjekte in den NFDI4Memory Knowledge Graph integriert werden. Wird das Informationsobjekt aktualisiert, muss der Identifier gleich bleiben, sodass eine korrekte Zuordnung im NFDI4Memory Knowledge Graph möglich ist.

Beispiele für physische Objekte sind analoge Dokumente wie Urkunden in einem Archiv oder Sammlungsobjekte wie Exponate in einem Museum. Beispiele für digital abrufbare Objekte sind Datensätze zu oder Retrodigitalisate von physischen Objekten wie z.B. einer gescannten Urkunde. Ein Informationsobjekt kann auch mehrere Metadatenobjekte haben. Z.B. können deskriptive und technische Metadaten vorliegen, die eine Ressource beschreiben und bspw. den Vorgang und die verwendeten Parameter der Digitalisierung der Ressource beschreiben. Eine andere Möglichkeit wäre, dass inhaltlich gleiche Metadaten in verschiedenen Formaten vorliegen (Abb. 4).

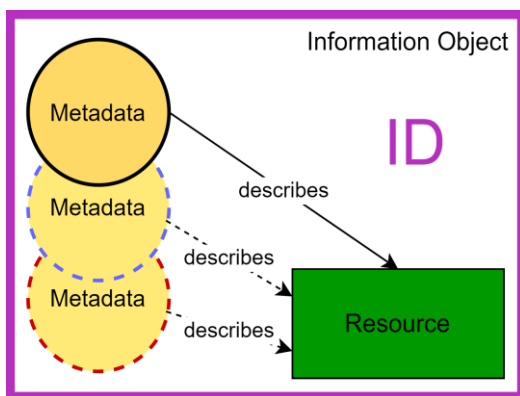


Abb. 4: Informationsobjekt mit digitaler, abrufbarer Ressource und mehreren Metadatenobjekten.

Wenn in einem Datensatz mehrere Ressourcen zu einer Einheit mit einem einzigen Metadatenobjekt zusammengefasst werden (Abb. 5), handelt es sich hingegen nicht um ein Informationsobjekt, da eine Teil-Ressource nicht direkt adressierbar ist. Das führt zu Problemen beim Auffinden von und Zugriff auf Teil-Ressourcen. Auch das Metadatenobjekt kann nur eine generalisierte Beschreibung der Teil-Ressourcen sein, da bei einer Spezialisierung des Metadatenobjekts die Gültigkeit zu einer oder mehreren Teil-Ressourcen verloren geht.

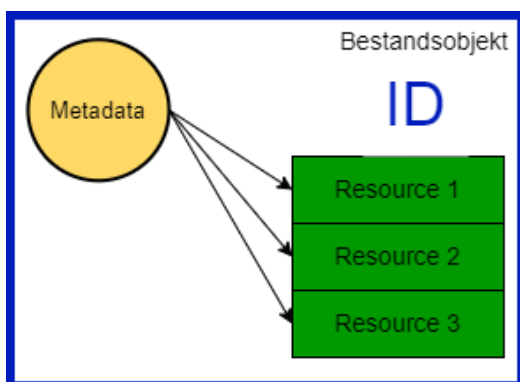


Abb. 5: Bestandsobjekt mit mehreren Ressourcen und Metadatenobjekt.

Für die Modellierung zu einem Informationsobjekt brauchen die einzelnen Teil-Ressourcen jeweils ein eigenes Metadatenobjekt und eine eigene ID. Um die ursprüngliche

Zusammengehörigkeit der Ressourcen beizubehalten kann ein ressourcenloses Informationsobjekt erstellt werden, dessen Metadatenobjekt Referenzen auf die herausgelösten Informationsobjekte enthält (Abb. 6).

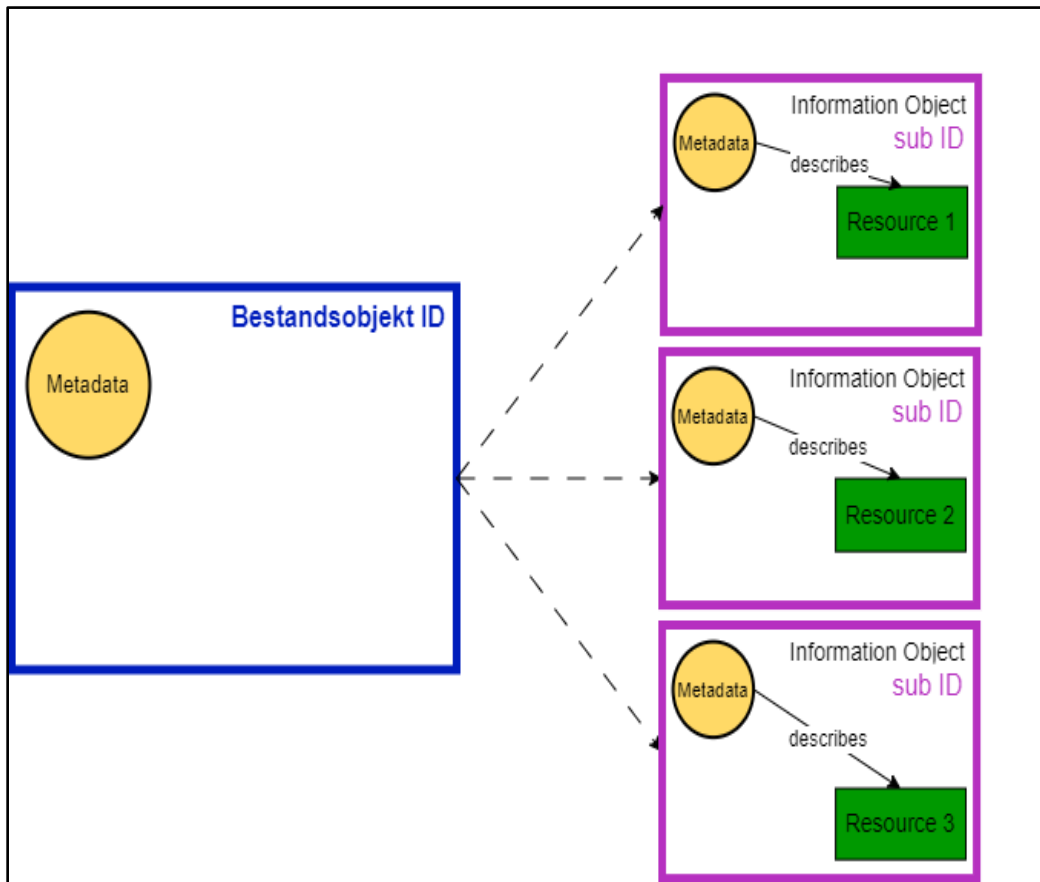


Abb. 6: Bestandsobjekt mit ID und daraus modellierte Informationsobjekte mit eigener ID.

In Archiven wird die sog. Tektonik für eine hierarchische Gliederung der Bestände in verschiedene Gruppierungen und die Klassifikation zur Binnenstrukturierung von Beständen bzw. Findbüchern verwendet. Das Archivgut wird nach Kriterien wie z.B. Provenienz oder zeitliche Zäsur in die verschiedenen Stufen der Tektonik und Klassifikation eingeordnet. Dabei beinhaltet jede Gliederungsebene eigene Informationen. Die Tektonik ermöglicht als Findmittel den systematischen Zugriff auf untergliederte Bestände und auf das jeweilige Archivgut sowie auf die darin enthaltenen Informationen. Dabei ist die Information der übergeordneten Ebenen für die Kontextualisierung des gesuchten Archivguts relevant bzw. ebenfalls Teil der Erschließungsinformation auf den untergeordneten Hierarchieebenen.

Um aus derartig strukturierten Datenbeständen ein Informationsobjekt zu modellieren, ist es notwendig, die Informationen aus allen darüber liegenden Ebenen der Tektonik sowie der Klassifikation und die Informationen für die Ressource bzw. die Archivalie selbst in ein Metadatenobjekt zusammenzuführen, das mit der Archivalie eindeutig in Beziehung steht und mit einer eindeutigen ID ausgezeichnet ist.

Im weiteren Verlauf des Textes gehen wir davon aus, dass alle Daten in den Datenquellen, die am NFDI4Memory Data Space partizipieren, als Informationsobjekte modelliert werden können. Die Ausgangslage ist dabei voraussichtlich sehr heterogen.

Der NFDI4Memory Data Space

Der NFDI4Memory Data Space ist als zentrale Anlaufstelle für die NFDI4Memory Community konzipiert und stellt eine innovative Plattform zur Verfügung, die das Auffinden, den Zugang und die Nutzung der vielfältigen Datenquellen und Dienste der beteiligten Institutionen vereinfacht und den Austausch innerhalb der wissenschaftlichen Community erleichtert. Konzeptionell besteht der NFDI4Memory Data Space aus zwei Komponenten. Die erste Komponente ist eine Menge an Schnittstellendefinitionen, die eine dezentrale Einbindung der Key Services und Datenquellen vorsieht, die zunächst vor allem von den an NFDI4Memory beteiligten Co-Applicants und Participants bereitgestellt werden (siehe oben). Die zweite Komponente ist der NFDI4Memory Knowledge Graph, welcher als zentrale Informationsquelle für den NFDI4Memory Data Space dient. Der NFDI4Memory Knowledge Graph kann in den Research Information Graph mit Informationen zu Diensten und Datenquellen sowie den teilnehmenden Institutionen und den Research Data Graph unterteilt werden, welcher Informationen über die verfügbaren Daten beinhaltet (siehe hierzu unten ausführlich den Abschnitt „NFDI4Memory Ontology (MemO und Knowledge Graph“).

Abbildung 7 zeigt eine schematische Darstellung des NFDI4Memory Data Space: Der NFDI4Memory Knowledge Graph nimmt eine Schlüsselrolle im Data Space ein und wird zentral bereitgestellt (Mitte), die Schnittstellen sorgen für die dezentrale Anbindung der teilnehmenden Datenquellen und Dienste (gerahmte Rechtecke) und ermöglichen die Suche über eine Weboberfläche mit REST- und SPARQL-Endpunkt (gerahmte Rechtecke links). Die Suchschnittstellen bieten Nutzenden die Möglichkeit, gezielt nach Informationen zu suchen oder die Informationen des NFDI4Memory Knowledge Graph explorativ zu erkunden. Die Schnittstellen werden im Abschnitt Query API & SPARQL weiter beschrieben.

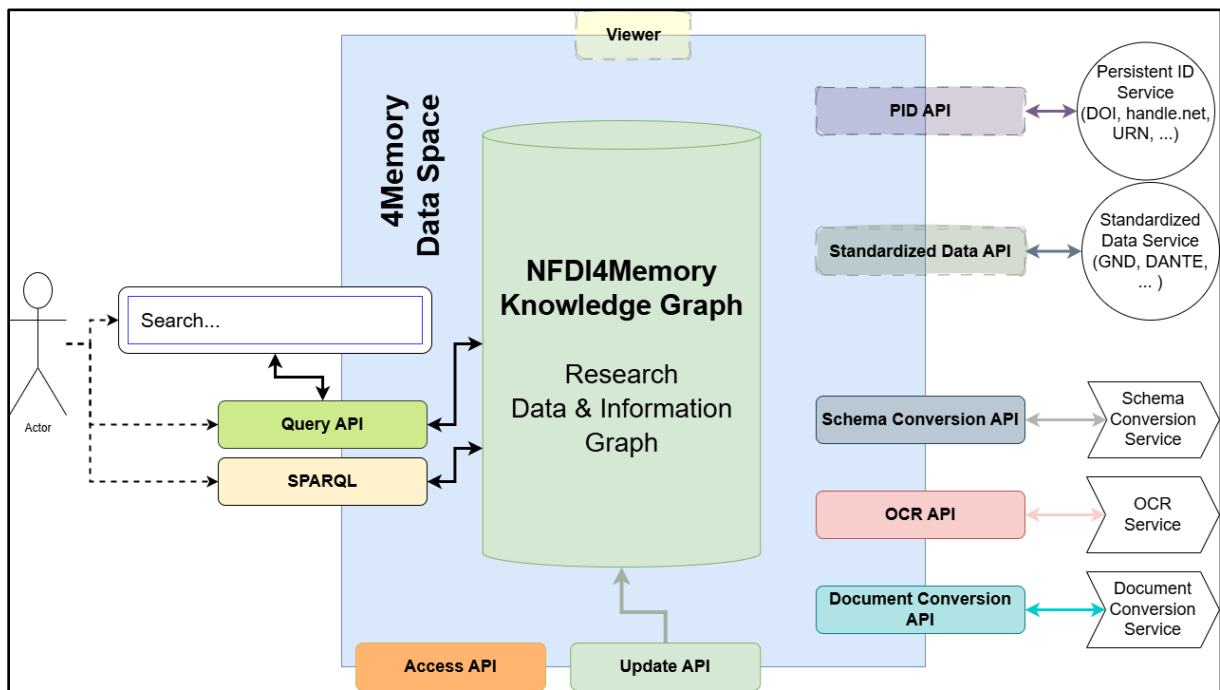


Abb. 7: Schematische Darstellung des NFDI4Memory Data Space.

Daten, die über die Such-Schnittstellen gefunden werden, können über die Access API (gerahmtes Rechteck unten links) heruntergeladen werden (siehe Abschnitt Access API). Die Update API (gerahmtes Rechteck unten rechts) ist die wichtigste Schnittstelle für teilnehmende Institutionen, über sie teilt eine Institution dem NFDI4Memory Knowledge Graph mit, welche Daten und Dienste sie dem NFDI4Memory Data Space zur Verfügung stellt. Sie muss von jeder teilnehmenden Institution implementiert werden. Abbildung 8 zeigt beispielhaft eine Institution, die dem NFDI4Memory Data Space drei Dienste bereitstellt: Zugriff auf ihre Daten, einen OCR-Dienst und einen Schema-Konvertierungsdienst. Dazu implementiert sie drei Schnittstellen: Access API, OCR API und Schema Conversion API. Damit die Daten und Dienste im NFDI4Memory Knowledge Graph auffindbar sind und aktuell bleiben, wird zusätzlich die Update-Schnittstelle implementiert.

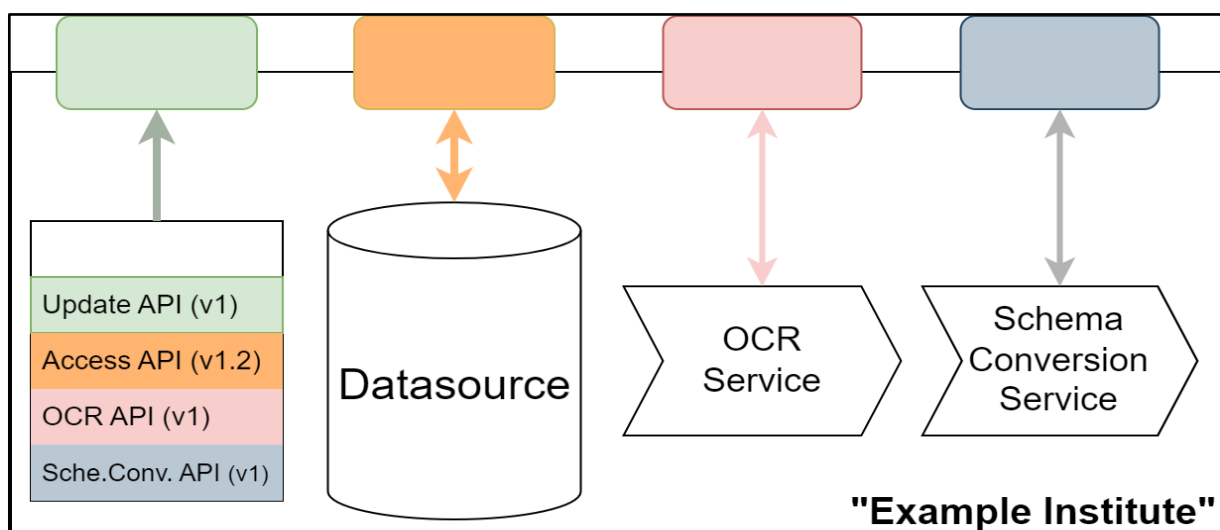


Abb. 8: Exemplarisches Institut nimmt am NFDI4Memory Data Space teil und stellt als Dienste den Zugriff auf Daten, einen OCR-Dienst und einen Schema-Konvertierungsdienst zur Verfügung.

Service-Schnittstellen (gerahmte Rechtecke rechts) sollen die Interaktion mit häufig benötigten Diensten vereinheitlichen. So können Institutionen ihre Dienste einfach in den NFDI4Memory Data Space integrieren und Forschende unabhängig von der konkreten Implementierung diese Dienste nutzen. Einen besonderen Vorteil bietet dies, wenn diese Dienste in Workflows und Anwendungen genutzt werden: Einmal programmierte Funktionen lassen sich transparent auf verschiedenen Implementierungen eines Dienstes ausführen. In diesem Entwurf des NFDI4Memory Data Space beschränken wir uns auf drei Service-Schnittstellen, diese werden im Detail im Abschnitt Service-Schnittstellen beschrieben.

Weitere mögliche APIs und Dienste (schraffierte Rechtecke) sind für den NFDI4Memory Data Space als Ideen für die zukünftige Erweiterung angedacht (z.B. eine PID API zur Beantragung von persistenten Identifiern oder eine standardisierte Schnittstelle zur Nutzung von für die Geschichtswissenschaften relevanten Vokabularen etc.).

Mithilfe der Viewer-Schnittstelle sollen spezialisierte Anzeigeprogramme (Viewer) für verschiedenste Forschungsdaten registriert werden können, sodass Nutzenden beim Öffnen eines Forschungsdatensatzes verschiedene Programme zur Auswahl stehen. Ein Viewer kann Nutzende unterstützen, indem verschiedene Ansichten eines Datensatzes ermöglicht werden. Viewer profitieren von den einheitlichen Schnittstellen des NFDI4Memory Data Space beim Zugriff auf Daten und Dienste.

Damit sich eine Institution am NFDI4Memory Data Space beteiligen kann, muss ihre Infrastruktur mindestens die Update API implementieren. Wurde die Implementierung erfolgreich verifiziert, wird die Institution in den NFDI4Memory Knowledge Graph aufgenommen und ihre Dienste und Datenquellen können über den NFDI4Memory Data Space gefunden und genutzt werden.

In den folgenden Abschnitten werden die beiden Komponenten NFDI4Memory Knowledge Graph und Schnittstellen des NFDI4Memory Data Space im Detail vorgestellt.

NFDI4Memory Ontology (MemO) und Knowledge Graph

Der NFDI4Memory Knowledge Graph²⁰ spielt eine Schlüsselrolle im NFDI4Memory Data Space. Ihm liegt die NFDI4Memory Ontology (MemO)²¹ zugrunde, die als ein disziplinspezifisches Modul der NFDI Core Ontology²² konzipiert ist und neben der NFDI Core

20 <https://nfdi.fiz-karlsruhe.de/4memory/> (aufgerufen am 30.07.2025).

21 Ondraszek, Sarah et al. „MemO v.1.0.0, Module Release: 2025-01-15“, <https://nfdi.fiz-karlsruhe.de/4memory/ontology/> (aufgerufen am 30.07.2025).

22 NFDIcore 3.0.1: <https://ise-fizkarlsruhe.github.io/nfdicore/docs/> (aufgerufen am 30.07.2025).

Ontology auch auf der NFDI4Culture Ontology (CTO)²³ und der Basic Formal Ontology (BFO)²⁴ aufbaut. Der NFDI4Memory Knowledge Graph wird unterteilt in den Research Information Graph (RIG) und den Research Data Graph (RDG). Der RIG beinhaltet Informationen über die beteiligten Institutionen sowie Informationen über Dienste, die Nutzenden des NFDI4Memory Data Space zur Verfügung stehen. Der RDG enthält einen Katalog über alle Daten in den Datenquellen der am NFDI4Memory Data Space beteiligten Institutionen. Mithilfe des RDG können die Datenquellen durchsucht und auf die Daten zugegriffen werden. Dadurch ist der NFDI4Memory Knowledge Graph die erste Anlaufstelle für Forscher:innen, um mit Recherchen zu beginnen.

Nutzende des NFDI4Memory Data Space können über zwei Schnittstellen mit dem NFDI4Memory Knowledge Graph interagieren: Zum einen über eine Weboberfläche, die ermöglicht, den NFDI4Memory Knowledge Graph explorativ und mit Suchbegriffen zu durchsuchen. Dabei können die angezeigten Elemente durch Facettierung eingeschränkt werden. Die Weboberfläche nutzt eine REST-API (siehe Query API & SPARQL), die Nutzende auch in eigene Anwendungen einbinden können. Zum anderen können Nutzende mit Erfahrung in semantischen Technologien eine SPARQL-Schnittstelle²⁵ verwenden. Diese Schnittstelle erlaubt es, komplexe Suchanfragen zu stellen oder den NFDI4Memory Knowledge Graph zu traversieren.

Damit die Informationen im NFDI4Memory Knowledge Graph aktuell und relevant bleiben, muss der NFDI4Memory Knowledge Graph regelmäßig aktualisiert werden. Dazu müssen beteiligte Institutionen eine Update-Schnittstelle (siehe unten Abschnitt Update API) bereitstellen, die in zyklischen Abständen abgerufen wird. Folgende Informationen werden über den Aufruf der Schnittstelle abgerufen und in den KG aufgenommen: Änderungen im Bestand, Änderungen an angebotenen Diensten und an den Stammdaten der Institution.

Schnittstellen

In diesem Abschnitt wird die Schnittstellenkomponente des NFDI4Memory Data Space erläutert. Generell sind alle Schnittstellen webbasiert und setzen auf dem HTTP-Protokoll auf. Um eine stetige Verbesserung des NFDI4Memory Data Space zu gewährleisten und Innovation nicht einzuschränken, werden alle vorgestellten Schnittstellen versioniert. Institutionen, die am NFDI4Memory Data Space teilnehmen, müssen angeben, welche

23 NFDI4Culture ontology (CTO) 3.0: <https://nfdi.fiz-karlsruhe.de/4culture/> (aufgerufen am 30.07.2025).

24 BFO 2020: <https://github.com/BFO-ontology/BFO-2020> (aufgerufen am 30.07.2025).

25 <https://www.w3.org/TR/rdf-sparql-query/> (aufgerufen am 30.07.2025).

Schnittstellen und entsprechenden Versionen angeboten werden. Die Schnittstellen werden in zwei Kategorien eingeteilt: die Core-Schnittstellen und die Service-Schnittstellen.

Core-Schnittstellen

Die Core-Schnittstellen (SPARQL, Query, Update und Access API) werden für die grundlegende Funktionalität des NFDI4Memory Data Space benötigt. Mit ihrer Hilfe ist es möglich, den gemeinsamen Bestand im NFDI4Memory Knowledge Graph aufzubauen, zu durchsuchen, aktuell zu halten und auf die Informationsobjekte eines teilnehmenden Instituts direkt zuzugreifen.

Query API & SPARQL

Die Query API sowie ein SPARQL-Endpunkt werden zentral innerhalb des NFDI4Memory Data Space bereitgestellt. Die Query API stellt einen REST-konformen Endpunkt dar, der die Durchführung einfacher, facettierter Suchanfragen auf dem NFDI4Memory Knowledge Graph ermöglicht. Sie dient nicht nur als Grundlage für eine zentral bereitgestellte Weboberfläche, die den Nutzenden eine intuitive Suchfunktionalität für den KG bietet, sondern kann auch in Drittanwendungen integriert werden. Die Spezifikationen dieser Schnittstelle sind im Abschnitt Query API detailliert beschrieben.

Der SPARQL-Endpunkt bietet ebenfalls eine dedizierte Weboberfläche, welche einen integrierten SPARQL-Editor umfasst. Diese Schnittstelle richtet sich an Nutzende mit fundierten Kenntnissen in semantischen Technologien und erlaubt die Durchführung gezielter sowie komplexer Abfragen des NFDI4Memory Knowledge Graph mittels der SPARQL-Abfragesprache. Der SPARQL-Endpunkt kann ebenso in eigene Anwendungen eingebunden werden. Die entsprechenden technischen Details finden sich im Abschnitt SPARQL.

Es besteht zudem die Möglichkeit, dass durch Drittentwickler:innen Anwendungen konzipiert und implementiert werden, welche beispielsweise eigene Datenbestände mit den Informationen aus dem NFDI4Memory Knowledge Graph aggregieren, um dadurch spezialisierte Suchlösungen mit erweitertem Funktionsumfang bereitzustellen.

Update API

Die Update API gewährleistet die Aktualität des NFDI4Memory Knowledge Graph und wird in regelmäßigen Intervallen (Harvesting) von der NFDI4Memory Knowledge Graph-Infrastruktur bei den beteiligten Institutionen aufgerufen. Sie stellt die einzige verpflichtende API dar, die von Institutionen implementiert werden muss, die aktiv am NFDI4Memory Data Space teilnehmen. Über diese Schnittstelle werden alle relevanten Informationen einer Institution für den NFDI4Memory Data Space ausgetauscht. Die zu übertragenden Daten umfassen:

1. Stammdaten der Institution
2. API Advertisements
3. Updateobjekte

Stammdaten der Institution

Die Stammdaten einer Institution sind weitgehend statisch und ändern sich in der Regel nur selten. Sie umfassen grundlegende Informationen wie den Namen der Institution und Kontaktmöglichkeiten. Über diese Kontakte können beispielsweise Fragen zu Informationsobjekten im NFDI4Memory Knowledge Graph oder zu angebotenen Diensten gestellt werden. Das Schema für die Stammdaten wird im Abschnitt Update API im Kapitel Spezifikationen definiert.

API Advertisements

API Advertisements enthalten Informationen zu Schnittstellen, die eine Institution implementiert und im NFDI4Memory Data Space bereitstellt. Zu den Inhalten eines API Advertisements gehören:

- Implementierte API und API Version
- Ggf. weitere Informationen zur API

Durch regelmäßige Aktualisierungen der API Advertisements können neue Schnittstellen oder Funktionen dynamisch hinzugefügt sowie nicht mehr angebotene Funktionen entfernt werden.

Updateobjekte

Wie oben im Abschnitt Datenstruktur beschrieben, gehen wir davon aus, dass die Daten einer Datenquelle, die in den NFDI4Memory Data Space aufgenommen werden soll, als Informationsobjekte modelliert werden können. Über die Update API werden die Informationsobjekte einer Datenquelle in den NFDI4Memory Knowledge Graph integriert. Abbildung 9 illustriert die Transformation eines Informationsobjekts (rechts; lila umrandet) in ein Data-Space-Objekt (links; grau umrandet):

- **Data-Space-Objekt:** Modelliert die Informationen, die im NFDI4Memory Knowledge Graph zu einem Informationsobjekt hinterlegt sind. Die grau gekennzeichneten Felder sind Informationen, die zum Betreiben des KG notwendig sind und können nicht von der datengebenden Institution beeinflusst werden.
- **Updateobjekt** (Mitte; blau umrandet): Dieses wird von der Institution über die Update API an den NFDI4Memory Knowledge Graph übermittelt und enthält:
 - den Identifier des Informationsobjekts
 - eine Menge an Metadaten
 - URLs, unter denen die Metadatenobjekte und die ggf. zugehörige Ressource heruntergeladen werden können

Die Struktur des Updateobjekts ist in den Spezifikationen im Abschnitt Update API definiert. Es ist möglich, die Updateobjekte in verschiedenen Formaten zu übertragen. Die Transformation eines Updateobjekts in ein Data-Space-Objekt übernimmt die Knowledge Graph-Infrastruktur. Es ist anzumerken, dass die Metadaten, die eine Institution aus einem Informationsobjekt in ein Updateobjekt schreibt, noch nicht fertig definiert sind.²⁶ Dieser Teil wird in Task Area Data Connectivity im Zuge des Schemas des NFDI4Memory Knowledge Graph definiert werden. Hier gilt es zu beachten: je mehr Metadaten das Updateobjekt beinhaltet, desto detaillierter können Suchen auf dem NFDI4Memory Knowledge Graph durchgeführt werden. Gleichzeitig vervielfältigen sich aber auch die Metadaten im NFDI4Memory Knowledge Graph, was zu einer redundanten Datenhaltung führt.

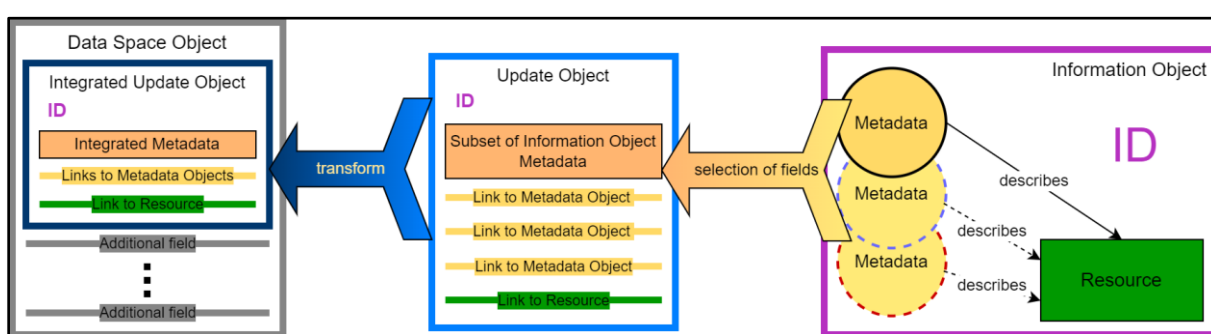


Abb. 9: Vom Informationsobjekt zum Data-Space-Objekt

Die technischen Spezifikationen zur Update API werden im Kapitel Spezifikationen im Abschnitt Update API erläutert.

Access API

Die Access API dient der Standardisierung des Zugriffs auf Metadatenobjekte und die zugehörigen Ressourcen von Informationsobjekten. Mit ihrer Implementierung machen Institutionen ihre Daten bzw. Informationsobjekte verfügbar und zugänglich und bieten Nutzenden beim Aufrufen von URLs ein einheitliches Anfrage- und Antwortverhalten gemäß den Spezifikationen der Access API. Die API macht keine Vorgaben hinsichtlich des URI-Schemas, definiert aber das Verhalten von Anfragen und Antworten.

Im Gegensatz zur Übermittlung von Daten an den NFDI4Memory Knowledge Graph über die Update API, wird bei der Bereitstellung von Metadatenobjekten und Ressourcen durch Institutionen über die Access API kein spezifisches Metadatenschema vorgeschrieben oder die Konvertierung der Ressourcen in ein vorgegebenes Format gefordert. Abgerufene Metadatenobjekte und Ressourcen werden direkt von der zugehörigen Institution in exakt der

²⁶ Eine Auswahl an Metadatenfelder in Form eines Minimalsets wird noch bereitgestellt werden. Vgl. hierzu vorerst die NFDI4Culture Guideline „Datenintegration in den Culture Knowledge Graph“: <https://nfdi4culture.de/id/E6283> (aufgerufen am 30.07.2025).

Form übertragen, in der sie dort auch gespeichert oder dem NFDI4Memory Data Space bereitgestellt werden. Der Zugriff auf den Inhalt eines Informationsobjekts erfolgt unverändert („raw“). Das soll es Institutionen ermöglichen, ihre Daten ohne zu großen Aufwand in den NFDI4Memory Data Space zu integrieren.

Die Spezifikation der Access API ist im Abschnitt Access API beschrieben.

Service-Schnittstellen

Die Service-Schnittstellen bieten Nutzenden einen einheitlichen Zugang zu häufig genutzten Diensten. Dadurch können beteiligte Institutionen unterschiedliche Implementierungen derselben Dienste bereitstellen, ohne dass die Nutzenden für jede Variante eine neue Bedienweise erlernen müssen. Außerdem können Nutzende ihre bisherigen Programme oder Skripte weiterhin verwenden bzw. lediglich die URL anpassen. Die im NFDI4Memory Data Space verfügbaren Dienste und zugehörigen Informationen, wie die URL, können über den NFDI4Memory Knowledge Graph (KG) entdeckt werden.

In den ersten Versionen des NFDI4Memory Data Spaces werden wir uns auf drei zentrale Dienste konzentrieren. In späteren Iterationen können bei Bedarf weitere Dienste hinzugefügt oder bestehende Service-APIs optimiert werden.

OCR API

Geschichtswissenschaftler:innen arbeiten oft mit historischen Dokumenten, die nicht in digitalen und durchsuchbaren Formaten vorliegen. OCR- und HTR-Tools (Optical Character Recognition bzw. Handwritten Text Recognition) sind daher essentiell, um gedruckte oder handschriftliche Texte in maschinenlesbare Formate umzuwandeln. Dies ermöglicht eine effiziente Textanalyse, da große Mengen an Quellenmaterial schneller durchsucht und relevante Inhalte leichter identifiziert werden können. Mit OCR-Tools lassen sich Texte auch für Sprach- oder Übersetzungsanalysen nutzen, was besonders bei mehrsprachigen historischen Quellen hilfreich ist. Mithilfe der Anpassung von OCR-Parametern und spezialisierten Algorithmen können auch die Ergebnisse für z.B. historische Texte mit unterschiedlichem Erhaltungszustand oder mit speziellen Schriftarten verbessert werden. Dennoch ist eine einheitliche und leicht zugängliche OCR-Schnittstelle für Geschichtswissenschaftler:innen sinnvoll, da sie den Zugang zu modernen Analysemethoden erleichtert und die Arbeit mit digitalen Texten zentralisiert.

Die OCR API bietet den Nutzenden des NFDI4Memory Data Space eine einheitliche Schnittstelle, um OCR-Funktionen implementierungsunabhängig für ihr Forschungsmaterial zu nutzen und Institutionen die Möglichkeit, OCR-Dienste den Nutzenden des NFDI4Memory

Data Space zur Verfügung zu stellen. Die Schnittstelle wird in dieser Version des Blue Papers noch sehr allgemein beschrieben.

Document Conversion API

Historiker:innen arbeiten oft mit vielfältigen Quellenmaterialien und Publikationsformaten – von wissenschaftlichen Artikeln und Büchern bis hin zu digitalen Archiven oder Webressourcen. Ein Dokumentenkonverter erleichtert dabei die Arbeit, verschiedenste Dokumente in ein einheitliches, frei wählbares Format zu bringen (sofern das Format vom Konverter unterstützt wird). Gerade in den Geschichtswissenschaften, wo der Brückenschlag zwischen traditionellen Texten und digitalen Formaten immer wichtiger wird, ist ein Dokumentenkonverter unverzichtbar, um Barrieren zwischen Formaten zu überwinden und wissenschaftliches Arbeiten zukunftsfähig zu gestalten.

Die Document Conversion API definiert eine einheitliche Schnittstelle, um ein implementierungsunabhängiges Integrieren von Dokumentenkonvertern in den NFDI4Memory Data Space zu ermöglichen.²⁷

Schema Conversion API

Ein Schema-Konverter ist in einem heterogenen Metadaten-Universum unverzichtbar, da er die Interoperabilität zwischen Systemen mit unterschiedlichen Metadatenschemata ermöglicht. Unterschiedliche Standards wie z.B. Dublin Core, MARC oder schema.org können durch einen Konverter harmonisiert werden. Da die Access API des NFDI4Memory Data Space keine Harmonisierung vorschreibt, ist es wahrscheinlich, dass eine für ein Projekt kompilierte Liste an historischen Quellen schwer vergleichbare Metadaten aufweist. Ein Schema-Konverter hilft, die Metadaten in ein vergleichbares Schema und Format zu transformieren, um eine bessere Vergleichbarkeit zu ermöglichen, verfügbare Mappings vorausgesetzt.

Die Schema Conversion API definiert eine einheitliche Schnittstelle, um ein implementierungsunabhängiges Integrieren von Schema-Konvertern in den NFDI4Memory Data Space zu ermöglichen.

²⁷ Auf diese Weise können Forscher:innen benötigte Dokumentenkonverter selbst integrieren. Derzeit planen wir, in der ersten Förderphase einen Konverter auszuwählen und exemplarisch in den NFDI4Memory Data Space zu integrieren.

Spezifikationen

Alle hier beschriebenen APIs basieren auf dem HTTP/S Protokoll in Version 1.1 (RFCs 7230-7235²⁸) oder höher. Zusätzlich werden die HTTP Status Codes benutzt und verstanden wie in RFC 9110²⁹ und ergänzenden RFC 6585³⁰.

Query API (v1.0)

Die Query API ermöglicht die Durchführung einer textuellen Suche innerhalb eines Knowledge Graphs.

Endpunkt

URL: `/api/v1/search`

Methoden: `GET`

Parameter

- **query** (string):
 - Beschreibung: Der Suchbegriff oder Text, der im Knowledge Graph gesucht werden soll.
 - Beispiel: `query=Künstliche Intelligenz`
- **page** (integer, optional):
 - Beschreibung: Der Index der aktuellen Seite, um die Ergebnisse zu paginieren. Standardwert ist `1`. Index beginnt mit `1`.
 - Beispiel: `page=2`
- **pageSize** (integer, optional):
 - Beschreibung: Die Anzahl der Ergebnisse pro Seite. Standardwert ist `10`.
 - Beispiel: `pageSize=25`
- **typeFilter** (string, optional):
 - Beschreibung: Einschränkung der Suche auf bestimmte Objekttypen im Knowledge Graph. Akzeptiert eine kommaseparierte Liste von Typen. Die Liste der Typen sind alle Individuen der Klasse `https://nfdi.fiz-karlsruhe.de/ontology/#ObjectType` aus dem KG-Schema.
 - Beispiel: `typeFilter=audio,video`

Antwort

Die Antwort erfolgt im JSON-Format.

Felder der Antwort

- **results** (Array):

28 <https://datatracker.ietf.org/doc/html/rfc7230> (aufgerufen am 30.07.2025)

29 <https://datatracker.ietf.org/doc/html/rfc9110#name-status-codes> (aufgerufen am 30.07.2025)

30 <https://datatracker.ietf.org/doc/html/rfc6585> (aufgerufen am 30.07.2025)

- Liste der Suchergebnisse.
- Struktur der Einträge:
 - **id** (string): Eindeutige ID des Objekts im Data Space.
 - **name** (string): Name oder Titel des Objekts.
 - **description** (string, optional): Kurzbeschreibung des Objekts.
 - **type** (string): Der Typ des Objekts (z. B. audio, video, text). Siehe KG-Schema <https://nfdi.fiz-karlsruhe.de/ontology/#ObjectType> Individuen.
- **pagination** (Objekt):
 - **currentPage** (integer): Die aktuelle Seite der Ergebnisse.
 - **pageSize** (integer): Anzahl der Ergebnisse pro Seite.
 - **totalPages** (integer): Gesamtanzahl der Seiten.
 - **totalResults** (integer): Gesamtanzahl der Ergebnisse.

Statuscodes

Hier die wichtigsten Statuscodes mit Beschreibung, weitere Codes möglich gemäß den Standard-HTTP-Statuscodes:

- **200 OK:** Die Anfrage wurde erfolgreich bearbeitet. Auch bei leerem Suchergebnis.
- **400 Bad Request:** Fehlende oder ungültige Parameter.
- [Beispielaufruf](#)

```
GET /api/v1/search?query=Mozart&page=1&pageSize=2&typeFilter=audio,video
HTTP/1.1

Host: api.example.com
Accept: application/json

HTTP/1.1 200 OK
Content-Type: application/json
{
  "results": [{
    "id": "https://4memory.de/dataspace/fiz-12345",
    "name": "Die Schuldigkeit des ersten Gebots",
    "description": "Das Werk KV 35 ist ein 1767 komponierte erste Teil eines geistlichen Singspiels.",
    "type": "audio"
  }, {
    "id": "https://4memory.de/dataspace/fiz-67890",
    "name": "La clemenza di Tito",
    "description": "Die Milde des Titus ist eine Opera seria (KV 621) in zwei Akten und sieben Bildern.",
    "type": "audio"
  }],
  "pagination": {
```

Update API (v1.0)

In dieser Version des Blue Papers wird nur eine Möglichkeit definiert, die Update API zu implementieren. In kommenden Versionen soll es dazu weitere Möglichkeiten geben. Die Update API setzt auf der OAI-PMH³¹ (Open Archives Initiative Protocol for Metadata Harvesting) Schnittstelle auf und erweitert die übertragenen Informationen um die Anforderungen des Data Space. Folgende Informationen müssen übertragen werden:

1. Stammdaten der Institution
2. API Advertisements
3. Updateobjekte

Für die ersten zwei Punkte wird die **Identify**-Funktion der OAI-PMH Spezifikationen verwendet. Die Stammdaten gleichen den Informationen in der Standardantwort eines **Identify**-Aufrufs. Die dem NFDI4Memory Data Space bereitgestellten Schnittstellen werden

31 <https://www.openarchives.org/OAI/openarchivesprotocol.html> (aufgerufen am 30.07.2025)

durch das Erweitern des **Description**-Elements in der **Identify**-Antwort mit einem eigenen API-Advertisement Schema übermittelt. Das XML-Schema ist im Anhang zu diesem Dokument zu finden.

- Update-API Advertisement: Informationen über die Update API Implementierung einer Institution
 - **version** (string)
 - Beschreibung: Gibt an, welche Version der Update API implementiert ist.
 - **implementation-type** (Enum<string>)
 - Beschreibung: Gibt an, in welcher Implementierung die Update API umgesetzt ist.
 - Werte: OAI-PMH
 - **url** (string)
 - Beschreibung: Die URL, unter der die Update API verfügbar ist.
 - **doc-url** (string, optional)
 - Beschreibung: Optional ein Link zur API Dokumentation der Implementierung.
- Access-API Advertisement: Informationen über die Access-API Implementierung
 - **version** (string)
 - Beschreibung: Gibt an, welche Version der Access-API implementiert ist.
 - **metadata-prefix** (string, optional)
 - Beschreibung: Gibt an, mit welchem Metadaten-Prefix die Data Space Updateobjekte abgerufen werden können.
 - Default: **n4m-ds**
 - **doc-url** (string, optional)
 - Beschreibung: Optional ein Link zur API Dokumentation der Implementierung.
- Service-API Advertisements: Informationen über die bereitgestellten Dienste einer Institution
 - **version** (string)
 - Beschreibung: Gibt an, in welcher Version eine bestimmte Service Schnittstelle implementiert ist.
 - **service-id** (string)
 - Beschreibung: Eindeutiger, persistenter, frei wählbarer Identifier (innerhalb einer teilnehmenden Institution) für einen bereitgestellten Service. Da ein Service-Typ (bspw. OCR API) von einer Institution mehrmals bereitgestellt werden kann, werden mithilfe der Service-ID diese Services unterscheidbar.
 - **url** (string)
 - Beschreibung: Die URL, unter der ein Service verfügbar ist.
 - **doc-url** (string)
 - Beschreibung: Link zur API Dokumentation des Services. Dieses Feld ist nicht optional. Da ein Service mit den richtigen Parametern effizienter verwendet werden kann, ist es notwendig, diese Parameter zu dokumentieren, sodass Nutzende diese Parameter nachlesen können.

Eine Beispielantwort eines **Identify**-Aufrufs wird im Anhang bereitgestellt.

Wird im API Advertisement eine Implementierung der Access-API mitgeschickt, so ruft die Harvesting-Infrastruktur des KG die **ListRecords**-Methode der OAI-PMH Schnittstelle mit dem im Access-API Advertisement angegebenen Metadaten-Prefix auf und harvestet die dort zurückgegebenen Objekte. Falls kein Metadaten-Prefix im Access-API Advertisement angegeben wird, wird die **ListRecords**-Methode mit dem Metadaten-Prefix **n4m-ds** aufgerufen. Das unter dem Metadaten-Prefix erwartete Schema wird momentan noch spezifiziert und diesem Dokument nachgereicht.

Access API (v1.0)

Die Access API bietet die Möglichkeit, Ressourcen oder zugehörige Metadatenobjekte abzurufen. Die URL der Schnittstelle ist frei wählbar und wird vom jeweiligen Institut frei gewählt.

Endpunkt

URL: Frei wählbar von der datengebenden Institution, wird im Update-Objekt mitgegeben.

Methoden: **GET**, **HEAD**

GET-Methode

Mit der **GET**-Methode wird die angeforderte Ressource oder das angeforderte Metadatenobjekt als Binärdaten zurückgegeben. Die Antwort enthält Header-Felder mit Metainformationen zum Download.

Header-Felder der Antwort

1. **Content-Type** (string): Der MIME-Typ der Binärdaten (z. B. **application/pdf**, **image/png**).
2. **Content-Length** (long): Die Größe der Binärdaten in Bytes.
3. **Content-Disposition** (string, recommended): Gibt an, dass die Binärdaten heruntergeladen werden sollen und optional der empfohlene Dateiname.
 - Beispiel: **attachment; filename="report.pdf"**

HEAD-Methode

Mit der **HEAD**-Methode werden nur die Header-Felder der Binärdaten ohne deren Inhalt zurückgegeben. Zusätzlich wird beim Abruf von Metadatenobjekten ein **Link**-Header gesetzt, der auf das verwendete Schema verweist, das in dem Metadatenobjekt genutzt wird. Dazu muss die Link-Relation **describedby** genutzt werden.

Zusätzliche Header-Felder (nur bei Metadatenobjekten notwendig)

- **Link** (string): Ein Link zum genutzten Schema des Metadatenobjekts.
 - Schema: **<http://example.com/my-ontology>; rel="describedby"**
 - Beispiel: **http://www.loc.gov/MARC21/slim; rel="describedby"**
 - Das Metadatenobjekt nutzt das MARC21 Schema.

Antwort

Die Antwort ist der Download der angeforderten Ressource oder des angeforderten Metadatenobjekts.

Statuscodes

Hier die wichtigsten Statuscodes mit Beschreibung, weitere Codes möglich gemäß den Standard-HTTP-Statuscodes:

- **200 OK:** Die Anfrage wurde erfolgreich bearbeitet.
- **301 Moved Permanently:** Die Daten sind unter einer anderen Location abrufbar. Die neue Location muss dem KG im nächsten Update mitgeteilt werden.
- **404 Not Found:** Die angeforderten Daten existieren nicht (mehr). Ein neuer Link zu einer Ressource oder einem Metadatenobjekt muss im KG hinterlegt oder entfernt werden, falls die Daten nicht mehr länger verfügbar sind.

Beispielaufruf

Get-Aufruf auf eine Ressource:

```
GET /api/v1/files/12345 HTTP/1.1
Host: institut.example.com
Accept: application/octet-stream

HTTP/1.1 200 OK
Content-Type: audio/flac
Content-Length: 1048576
Content-Disposition: attachment; filename="La clemenza di Tito.flac"

<Binärdaten>
```

Head-Aufruf auf ein Metadatenobjekt:

```
GET /api/v1/metadata/12345 HTTP/1.1
Host: institut.example.com
Accept: application/octet-stream

HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 5324
Link: http://www.lido-schema.org; rel="describedby"
```

Schema Converter API (v1.0)

Die Schema Converter API ermöglicht die Transformation von strukturierten Textdateien mithilfe eines Mappings in andere Formate. Die unterstützten Mapping-Typen (bspw. XSLT oder JSONata) werden vom Diensteanbieter festgelegt und sind in der API-Dokumentation beschrieben. Der Link zur API-Dokumentation wird im Rahmen eines API-Advertisements im KG gespeichert.

Endpunkt

URL: Vom Dienstanbieter frei wählbar.

Methoden: POST

Parameter

Query-Parameter

- **sourceUrls** (string[]):
 - Beschreibung: Eine oder mehrere URLs, die auf strukturierte Textdateien verweisen, die transformiert werden sollen. Kann mit **sourceFiles** kombiniert werden.
 - Beispiel: `sourceUrls=https://example.com/file1.xml,https://example.com/file2.json`
- **mappingUrl** (string):
 - Beschreibung: Das Mapping, das für die Transformation verwendet wird. Kann nicht mit **mappingFile** kombiniert werden.
 - Beispiel: `mappingFile=https://example.com/lido-mapping.xml`
- **mappingType** (string):
 - Beschreibung: Der Typ des Mappings (z. B. **XSLT** oder **JSONata**).
 - Hinweis: Eine Liste der unterstützten Mapping-Typen muss in der API-Dokumentation der Implementation beschrieben sein.
 - Beispiel: `mappingType=XSLT`

Request Body

Enthält der Request Body ein oder mehrere Dateien muss er im multipart/form-data Format gesendet werden.

- **sourceFiles**(file[]):
 - Beschreibung: Eine oder mehrere strukturierte Textdateien, die transformiert werden sollen. Kann mit **sourceUrls** kombiniert werden.
 - Beispiel: Hochladen von XML- oder JSON-Dateien.
- **mappingFile** (file):
 - Beschreibung: Das Mapping, das für die Transformation verwendet wird. Kann entweder als Datei oder als URL übergeben werden. Kann nicht mit **mappingUrl** kombiniert werden.
 - Beispiel: Hochladen eines XSLT- oder JSONata-Files oder Angabe einer URL.

Kombinationsmöglichkeiten

Die Parameter für die zu transformierenden Textdateien können kombiniert werden, sodass es möglich ist, URLs und Dateiuploads in einer Anfrage zu haben.

Der Parameter zum Mapping-File kann nicht kombiniert werden. Es wird immer versucht, alle Textdateien mit dem gegebenen Mapping-File zu transformieren.

Antwort

Eine konkrete Implementierung kann mehrere Antwortformate unterstützen. Mindestens `application/zip` muss von jeder Implementierung unterstützt werden.

Statuscodes

Hier die wichtigsten Statuscodes mit Beschreibung, weitere Codes möglich gemäß den Standard-HTTP-Statuscodes:

- **400 Bad Request:** Fehlende oder ungültige Parameter(-kombinationen).
- **404 Not Found:** Zu einer oder mehreren gegebenen URLs kann keine Datei geladen werden.
- **415 Unsupported Media Type:** Der angegebene Mapping-Typ wird nicht unterstützt.

Beispielanfrage

```
POST /api/v1/schema-
converter?sourceUrls=https%3A%2F%2Fexample.com%2Ffile1.xml&mappingType=XSL
T HTTP/1.1
Host: api.example.com
Content-Type: multipart/form-data

--boundary
Content-Disposition: form-data; name="sourceFiles"; filename="file2.xml"
Content-Type: application/xml

<Inhalt der Datei file2.xml>
--boundary
Content-Disposition: form-data; name="mappingFile"; filename="mapping.xml"
Content-Type: application/xml
```

OCR API (v1.0)

Die OCR API bietet die Möglichkeit, Zeichen in Bilddigitalisaten in maschinenlesbaren Text zu verwandeln. Sie unterstützt verschiedene Eingabearten und erlaubt die Angabe von spezifischen Optionen für die Texterkennung.

Endpunkt

URL: Frei von der dienst anbietenden Institution wählbar.

Methoden: `POST`

Parameter

Query-Parameter

- **sourceUrls** (string[]):
 - Beschreibung: URLs zu einem oder mehreren Bildern, die konvertiert werden sollen.
 - Beispiel:
`sourceUrls=https://example.com/image1.png,https://example.com/image2.jpg`
- **sourceIds** (string[]):
 - Beschreibung: Data Space IDs von Bild-Ressourcen. Der Service versucht, anhand dieser IDs an die Ressourcen-URLs der Bilder zu gelangen, diese herunterzuladen und diese zu konvertieren.
 - Beispiel: `sourceIds=https://4memory.de/dataspace/fiz-12345,https://4memory.de/dataspace/fiz-67890`

Request Body

Enthält der Request Body **sourceFiles** muss im multipart/form-data Format gesendet werden.

- **sourceFiles** (file[]):
 - Beschreibung: zu konvertierende Bilder, die direkt im Request Body hochgeladen werden.
- **options** (JSON-Objekt):
 - Beschreibung: Ein JSON-Objekt mit spezifischen Optionen für den implementierten OCR-Converter.
 - Details: Die Struktur und möglichen Felder dieses Objekts sind in der API-Dokumentation der Implementierung beschrieben. Die Dokumentation kann über die `doc-url` im API-Advertisement gefunden werden.

Die Parameter **sourceUrls**, **sourceIds**, **sourceFiles** können beliebig miteinander kombiniert werden.

Unterstützte Bildformate müssen in der Implementierungsdokumentation spezifiziert werden.

Antwort

Die API liefert den erkannten Text und optionale Metadaten in den in der Implementierungsdokumentation beschriebenen Formaten.

Statuscodes

Hier die wichtigsten Statuscodes mit Beschreibung, weitere Codes möglich gemäß den Standard-HTTP-Statuscodes:

- **400 Bad Request:** Fehlende oder ungültige Parameter.
- **404 Not Found:** Eine im **sourceUrls** oder **sourceIds** Parameter angegebene Ressource konnte nicht gefunden werden.
- **415 Unsupported Media Type:** Eine Ressource hat ein nicht unterstütztes Bildformat.

Beispielanfrage

```
POST
/api/v1/ocr?sourceUrls=https%3A%2F%2Fexample.com%2Fimage.jpg&sourceIds=https%
3A%2F%2F4memory.de%2Fdataspace%2Ffiz1234 HTTP/1.1
Host: api.example.com
Content-Type: multipart/form-data

--boundary
Content-Disposition: form-data; name="sourceFiles"; filename="image2.png"
Content-Type: image/png

<Inhalt der Datei image2.png>
--boundary
Content-Disposition: form-data; name="options"
Content-Type: application/json
```

Document Converter API (v1.0)

Die Document Converter API bietet die Möglichkeit, die Dateiformate von Textdokumenten umzuwandeln. Sie unterstützt verschiedene Eingabearten und erlaubt die Angabe von spezifischen Optionen für die Dokumentkonvertierung.

Endpunkt

URL: Frei von der dienst anbietenden Institution wählbar.

Methoden: POST

Parameter

Query-Parameter

- **sourceUrls** (string[]):
 - Beschreibung: URLs zu einem oder mehreren Textdokumenten, die konvertiert werden sollen.
 - Beispiel: `sourceUrls=https://example.com/doc1.docx,https://example.com/doc2.docx`
- **sourceIds** (string[]):
 - Beschreibung: Data Space IDs von Textdokumenten-Ressourcen. Der Service versucht, anhand dieser IDs an die Ressourcen-URLs der Textdokumente zu gelangen, diese herunterzuladen und zu konvertieren.
 - Beispiel: `sourceIds=https://4memory.de/dataspace/fiz-12345,https://4memory.de/dataspace/fiz-67890`
- **outputFormat** (string):
 - Beschreibung: Das Ausgabeformat, in das alle Eingabe-Textdokumente konvertiert werden sollen.

- Beispiel: `outputFormat=PDF`

Request Body

Enthält der Request Body **sourceFiles** muss im multipart/form-data Format gesendet werden.

- **sourceFiles** (file[]):
 - Beschreibung: zu konvertierende Textdokumente, die direkt im Request Body hochgeladen werden.
- **options** (JSON-Objekt):
 - Beschreibung: Ein JSON-Objekt mit spezifischen Optionen für den implementierten Document-Converter.
 - Details: Die Struktur und möglichen Felder dieses Objekts sind in der API-Dokumentation der Implementierung beschrieben. Die Dokumentation kann über die `doc-url` im API-Advertisement gefunden werden.

Die Parameter **sourceUrls**, **sourceIds**, **sourceFiles** können beliebig miteinander kombiniert werden.

Unterstützte Textdokumentformate für Eingabe und Ausgabe müssen in der Implementierungsdokumentation spezifiziert werden.

Antwort

Die API liefert bei einem Eingabedokument das konvertierte Ausgabedokument im gewünschten Format zum Download zurück. Bei mehreren Eingabedokumenten werden die konvertierten Ausgabedokumente gepackt zurückgeliefert. Der Service muss mindestens `application/zip` als Packformat unterstützen.

Statuscodes

Hier die wichtigsten Statuscodes mit Beschreibung, weitere Codes möglich gemäß den Standard-HTTP-Statuscodes:

- **400 Bad Request:** Fehlende oder ungültige Parameter.
- **404 Not Found:** Eine im **sourceUrls** oder **sourceIds** Parameter angegebene Ressource konnte nicht gefunden werden.
- **412 Precondition Failed:** Die Eingabedokumente haben nicht dasselbe Dateiformat.
- **415 Unsupported Media Type:** Ein Eingabedokument hat ein nicht unterstütztes Dokumentenformat. Oder das angegebene Ausgabeformat wird nicht unterstützt.

Beispielanfrage

```
POST /api/v1/doc-conv?sourceUrls=https%3A%2F%2Fexample.com%2Fdoc.docx&sourceIds=https%3A%2F%2F4memory.de%2Fdataspace%2Ffiz1234 HTTP/1.1
Host: api.example.com
Content-Type: multipart/form-data

--boundary
Content-Disposition: form-data; name="sourceFiles"; filename="doc2.docx"
Content-Type: application/vnd.openxmlformats-officedocument.wordprocessingml.document

<Inhalt der Datei doc2.docx>
--boundary
Content-Disposition: form-data; name="options"
Content-Type: application/json
```

Anmerkungen

Zu den Nutzenden des NFDI4Memory Data Space

Es ist uns bewusst, dass APIs nicht dazu gedacht sind, von einem Nutzer händisch genutzt zu werden. Der NFDI4Memory Data Space ist in erster Linie als Interoperabilitätsschicht zwischen verteilten Institutionen zu sehen und soll bevorzugt mithilfe von Software genutzt werden. Dabei ist Software als breit gefächelter Begriff zu verstehen, der von Institutionen bereitgestellte Desktopprogramme bis hin zu kleine von Forschenden geschriebene Skripte einschließt.

Zu Sicherheitsaspekten des NFDI4Memory Data Space

Uns ist bewusst, dass diese Version des Blue Papers noch keinerlei Überlegungen zu den Sicherheitsaspekten des NFDI4Memory Data Space anstellt. Diesem wesentlichen Punkt werden wir uns in den folgenden Iterationen des NFDI4Memory Data Space und den nächsten Versionen des Blue Papers widmen. Besonders hervorzuheben sind dabei Authentifizierung und Autorisierung der Nutzer:innen, um u.a. Zugriffe sicher zu gestalten und Datenintegrität und -verfügbarkeit sicher zu stellen. Da der NFDI4Memory Knowledge Graph einen Single-Point of Truth und Single-Point of Trust darstellt, ist es außerordentlich wichtig, dass keine falschen, manipulierten oder in irgendeiner Art bösartigen Informationen in den NFDI4Memory Knowledge Graph aufgenommen werden. Wichtige Anknüpfungspunkte werden NFDI-weite Lösungen, wie z. B. der Dienst IAM4NFDI³² und die Überlegungen der Sektion "Common

³² Im Rahmen des Base4NFDI-Projekts wird eine Authentifizierungs- und Autorisierungsinfrastruktur (AAI) für die NFDI entwickelt, siehe: <https://base4nfdi.de/projects/iam4nfdi> (aufgerufen am 30.07.2025).

Infrastructures”³³ sein. Zudem wird im Austausch mit der NFDI4Memory-Community ein Rights Managment Policy Framework erarbeitet, welches Richtlinien für die Nutzung des NFDI4Memory Data Space festlegen soll.

Anhang

Beispiel Identify-Antwort in der OAI-PMH Update API

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
>
  <Identify>
    <repositoryName>My Repository</repositoryName>
    <baseURL>https://example.com/oai</baseURL>
    <protocolVersion>2.0</protocolVersion>
    <adminEmail>admin@example.com</adminEmail>
    <earliestDatestamp>2020-01-01T00:00:00Z</earliestDatestamp>
    <deletedRecord>persistent</deletedRecord>
    <granularity>YYYY-MM-DD</granularity>
    <description>
      <api-advertisements
        xmlns="https://4memory.de/dataspace/ApiAdvertisement"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      >
        <update-api>
          <version>1.0</version>
          <implementation-type>OAI-PMH</implementation-type>
          <url>https://example.com/oai</url>
        </update-api>
        <access-api>
          <version>1.0</version>
          <metadata-prefix>4memory-dataspace</metadata-prefix>
        </access-api>
      </api-advertisements>
    </description>
  </Identify>
</OAI-PMH>
```

33 <https://www.nfdi.de/section-infra/> (aufgerufen am 30.07.2025).

```

<services>

  <ocr-api>

    <version>1.0</version>

    <service-id>abcdf123456</service-id>

    <url>https://api.example.com/ocr1</url>

    <doc-url>https://api.example.com/ocr1/api-docs</doc-url>

  </ocr-api>

  <ocr-api>

    <version>1.0</version>

    <service-id>987564fedcba</service-id>

    <url>https://api.example.com/ocr2</url>

    <doc-url>https://api.example.com/ocr2/api-docs</doc-url>

  </ocr-api>

  <schema-converter-api>

    <version>1.0</version>

    <service-id>schema-converter</service-id>

    <url>https://api.example.com/schema-converter</url>

    <doc-url>https://api.example.com/schema-converter/api-docs</doc-url>

  </schema-converter-api>

</services>

</api-advertisements>

</description>

</Identify>

</OAI-PMH>

```

API Advertisement Schema (in XSD)

```

<?xml version="1.0" encoding="UTF-8"?>

<schema

  xmlns="http://www.w3.org/2001/XMLSchema"

  xmlns:n4m="https://4memory.de/dataspace/ApiAdvertisement"

  targetNamespace="https://4memory.de/dataspace/ApiAdvertisement"

  elementFormDefault="qualified"

  version="1.0"

```



```

>
<!-- GENERAL TYPES -->
<complexType name="apiBaseType" abstract="true">
  <annotation>
    <documentation>The base type for each advertisement. Forces each advertisement to have a
      version element.</documentation>
  </annotation>
  <sequence>
    <element name="version" minOccurs="1" maxOccurs="1">
      <annotation>
        <documentation>Element indicates the implemented API version.</documentation>
      </annotation>
      <simpleType>
        <restriction base="string">
          <pattern value="[0-9]+(\.[0-9]+)*"></pattern>
        </restriction>
      </simpleType>
    </element>
  </sequence>
</complexType>

<simpleType name="urlBaseType">
  <restriction base="string">
    <pattern value="https?://([a-zA-Z0-9-]+\.[a-zA-Z]{2,})(:[0-9]{1,5})?(/*.*)?"/>
  </restriction>
</simpleType>

<simpleType name="urlType">
  <annotation>
    <documentation>Declares the URL for reaching a core or service API.</documentation>
  </annotation>
  <restriction base="n4m:urlBaseType" />
</simpleType>

<simpleType name="docuUrlType">
  <annotation>

```

```

    <documentation>Declares the URL for reaching an OpenAPI or WSDL documentation.</documentation>
  </annotation>

  <restriction base="n4m:urlBaseType" />
</simpleType>

<!-- ##### -->
<!-- UPDATE API -->
<simpleType name="updateApiImplementationType">
  <restriction base="string">
    <enumeration value="OAI-PMH" />
  </restriction>
</simpleType>

<complexType name="updateApiType">
  <complexContent>
    <extension base="n4m:apiBaseType">
      <sequence>
        <element name="implementation-type" type="n4m:updateApiImplementationType" minOccurs="1"
          maxOccurs="1">
          <annotation>
            <documentation>This element states the kind of implementation used by the
              infrastrucutre to implmenet the Update API.</documentation>
          </annotation>
        </element>
        <element name="url" type="n4m:urlType" minOccurs="1" maxOccurs="1" />
        <element name="doc-url" type="n4m:docuUriType" minOccurs="0" maxOccurs="1" />
      </sequence>
    </extension>
  </complexContent>
</complexType>

<!-- ##### -->
<!-- ACCESS API -->
<complexType name="accessApiType">
  <annotation>
    <documentation>The Access API advertisements type.</documentation>
  </annotation>

```

```

<complexContent>
  <extension base="n4m:apiBaseType">
    <sequence>
      <element name="metadata-prefix" type="string" minOccurs="0" maxOccurs="1" default="n4m-ds">
        <annotation>
          <documentation> This value is used as metadata-prefix when harvesting the data from
            the OAI provider. If this field is not set, the default metadata-prefix 'n4m-ds' is
            used. </documentation>
        </annotation>
      </element>
      <element name="doc-url" type="n4m:docUrlType" minOccurs="0" maxOccurs="1" />
    </sequence>
  </extension>
</complexContent>
</complexType>
<!-- ##### -->
<!-- SERVICE BASE -->
<complexType name="baseServiceApiType" abstract="true">
  <annotation>
    <documentation> This type is the base type for all Service API advertisements. A Service API
      advertisement has to have an ID to distinct multiple services of the same type from each
      other. This ID must be persistent between updates, so that the Research Infromation Graph
      can update exisiting service advertisements. </documentation>
    </annotation>
    <complexContent>
      <extension base="n4m:apiBaseType">
        <sequence>
          <element name="service-id" minOccurs="1" maxOccurs="1">
            <annotation>
              <documentation> An ID to identify the service. It must be unique for each service
                advertisement! If multiple services have the same ID, only the last service will be
                included. Must be at least 8 character long an is only allowed to contain
                alphanumeric values. </documentation>
            </annotation>
          </simpleType>

```

```

    <restriction base="string">
      <minLength value="8" />
      <pattern value="[a-zA-Z0-9_-]+"></pattern>
    </restriction>
  </simpleType>
</element>

<element name="url" type="n4m:urlType" minOccurs="1" maxOccurs="1" />
<element name="doc-url" type="n4m:docUrlType" minOccurs="1" maxOccurs="1" />
</sequence>
</extension>
</complexContent>
</complexType>
<!-- ##### -->
<!-- SCHEMA CONVERTER API TYPE -->
<complexType name="schemaConverterApi">
  <annotation>
    <documentation>The Schema Converter Service API advertisement type.</documentation>
  </annotation>
  <complexContent>
    <extension base="n4m:baseServiceApiType">
    </extension>
  </complexContent>
</complexType>
<!-- ##### -->
<!-- OCR API TYPE -->
<complexType name="ocrApi">
  <annotation>
    <documentation>The OCR Service API advertisement type.</documentation>
  </annotation>
  <complexContent>
    <extension base="n4m:baseServiceApiType">
    </extension>
  </complexContent>
</complexType>
<!-- ##### -->

```

```

<!-- DOCUMENT CONVERTER API TYPE -->
<complexType name="documentConverterApi">
  <annotation>
    <documentation>The Document Converter Service API advertisement type.</documentation>
  </annotation>
  <complexContent>
    <extension base="n4m:baseServiceApiType">
    </extension>
  </complexContent>
</complexType>

<!-- ##### -->
<!-- ADVERTISEMENT ELEMENT -->
<element name="api-advertisements">
  <complexType>
    <all>
      <element name="update-api" type="n4m:updateApiType" minOccurs="1" maxOccurs="1">
        <annotation>
          <documentation>Describes the implemented Update API.</documentation>
        </annotation>
      </element>
      <element name="access-api" type="n4m:accessApiType" minOccurs="0" maxOccurs="1">
        <annotation>
          <documentation>Describes the implemented Access API.</documentation>
        </annotation>
      </element>
      <element name="services" minOccurs="0" maxOccurs="1">
        <annotation>
          <documentation> Contains all services provided by an institution for the 4Memory Data
            Space. If an institution provides multiple instances of a service type each instance
            must be advertised explicitly. </documentation>
        </annotation>
        <complexType>
          <choice maxOccurs="unbounded">
            <element name="schema-converter-api" type="n4m:schemaConverterApi" minOccurs="0">
              <annotation>

```

```
<documentation>Describes the implemented Schema Converter API.</documentation>
</annotation>
</element>
<element name="ocr-api" type="n4m:ocrApi" minOccurs="1" maxOccurs="1">
  <annotation>
    <documentation>Describes the implemented OCR API.</documentation>
  </annotation>
</element>
<element name="document-converter-api" type="n4m:documentConverterApi" minOccurs="0">
  <annotation>
    <documentation>Describes the implemented Document Converter API.</documentation>
  </annotation>
</element>
</choice>
</complexType>
</element>
</all>
</complexType>
</element>
</schema>
```