GWDG
BERICHT
78

Oliver Schmitt
Andreas Siemon
Ulrich Schwardmann
Marcel Hellkamp

GWDG Object Storage and Search Solution for Research

Common Data Storage Architecture (CDSTAR)

Oliver Schmitt
Andreas Siemon
Ulrich Schwardmann
Marcel Hellkamp

GWDG Object Storage and Search Solution for Research

Common Data Storage Architecture (CDSTAR)

GWDG-Bericht Nr. 78
Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

Impressum

Gesellschaft für wissenschaftliche
Datenverarbeitung mbH Göttingen
Am Faßberg 11
37077 Göttingen

Telefon: 0551 201-1510
Telefax: 0551 201-2150
E-Mail: gwdg@gwdg.de
WWW: www.gwdg.de

 ISSN 0176-2516
www.gwdg.de/gwdg-berichte
Titelbild: © karelnoppe – Fotolia.com

Table of Contents

1 Introduction

This report describes CDSTAR, which is GWDG's system for storing and searching objects in research projects. The goal of the project is to provide a storage system for structured and unstructured data. One of our main assumptions is that the data originates from research projects with diverse scientific backgrounds. The use cases that provide the requirements for CDSTAR span digital humanities, astronomy, and medical information management. The goal of CDSTAR is to provide an easy-to-use storage system that can store, modify, annotate, search, and access a large amounts of diverse data.

The amount of data is growing rapidly in all disciplines of science. In many areas the number of relations between data and resources become an essential factor for successful research. Therefore it is necessary to develop strategies for science and research to safeguard and resuse unique data, to conserve relationships in the data, and to verify research results. To realize these strategies, research data has to be loaded into an information system, which can cover the entire data lifecycle. Additionally data availability and persistent identifiers need to be provided for long periods, during which hardware and software infrastructures are subject to change. Thus appropriate abstraction layers and sustainable service interfaces that follow open industry standards are a necessary choice.

To ensure technological sustainability not only in the frontend, which faces the clients, but also in the backend, CDSTAR builds on Cloud technologies. As such it complements GWDG's Compute Cloud with an elastic storage service. Thus CDSTAR can be compared to a number of popular commercial storage services. CDSTAR mimics Amazon S3 [1] and Microsoft Azure Blob Storage [2] but has been tailored to the needs of data management in science and research and thus supports the requirements for good scientific practice [3]. CDSTAR also integrates an enterprise-grade search engine that operates on metadata and full text. This search engine indexes a wide range of file formats, including PDF-files, Microsoft Office files, XML-documents, archives, video, and audio files (over 40 major formats) [4]. All functions of GWDG CDSTAR can be accessed through its RESTful-interface [5]. As a consequence CDSTAR can be used by many programming platforms and programming languages, such as desktop clients, HTML5-applications, traditional web-applications, mobile clients for iOS and Android, command line tools, and enterprise portals. With its RESTful-interface CDSTAR is a good

integration platform to connect applications such as web applications, desktop applications, and even mobile applications. GWDG CDSTAR can be used as long term archive, as data integration platform or as application storage for applications in science and research.

Object Storage and Search – Ready for Science and Research

The GWDG has built a custom object storage solution for science and research. This solution addresses the specific requirements of research data management according to the good scientific practice. This integrates the ability of storing metadata along the research data in a flexible metadata schema that can be tailored for the specific use in different scientific disciplines. Additionally the data that are stored in GWDG CDSTAR can be registered automatically at the EPIC Persistent Identifier (PID) service [6], where one instance is hosted at the GWDG. The EPIC service gives data sets a unique, globally resolvable identifier as an additional abstraction layer that allows citing data sets in scientific publication. A role-based security concept allows the protection of data sets with an individual set of permissions and rights for each user. User identification can be integrated within the GWDG identity management, application specific user data bases or LDAP [7]. This allows the creation of single-sign-on scenarios and a smooth operation between scientific applications and the GWDG CDSTAR.

GWDG CDSTAR object storage allows the usage of different storage scenarios starting from small data sets to large data sets. In contrast to generic industry offered object storages such as Amazon S3, the GWDG object storage allows the researchers to have a solution for their specific needs. The GWDG offers a set of storage backend that let customers select different venues for data storage, back scenarios and replication to remote data centers of the GWDG or partner data centers. Although CDSTAR follows the cloud paradigm, the researcher has control over its data and can use this storage solution to store and search its data.

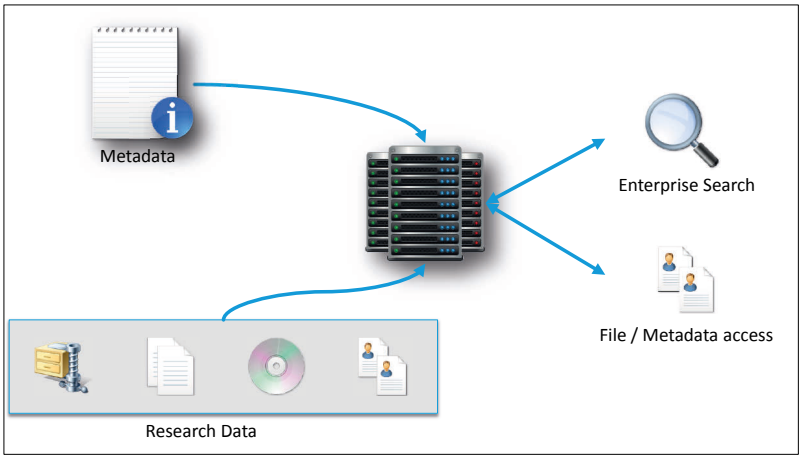


Figure 1: GWDG CDSTAR features

Supporting Research Data

GWDG CDSTAR covers all phases of research data management. Scientific data can be uploaded, read, updated and deleted through the REST-interface under the usage HTTP (Hyper Text Transfer Protocol) [7], an open well-supported transfer protocol, that is integrated in most programming languages and software frameworks. As scientific data needs to be enriched with descriptions such as author, time, date, parameters or settings of equipment, rigs or experiments, GWDG CDSTAR allow a metadata annotation of each data set. The JSON-based (JavaScript Object Notation) [9] metadata schema and handling allows a flexible support of research use cases and an iterative development of metadata schema. GWDG STAR automatically assigns Persistent Identifiers (PID) to data sets in order to make them citable in scientific papers. So research data sets and results can be published to the internet with one mouse click and can be retrieved via the international Handle-powered PID system. But GWDG CDSTAR also handles the end of the data life cycle, where data may be deleted or archived on tape for long term preservation. To archive this, research data and associated metadata sets stored in GWDG CDSTAR can be transferred into the GWDG archive system as compact data objects consisting of files, JSON-formatted metadata and information about file owners and original permissions. The use of JSON data for-

matting ensures a long-term usage and reuse of data in future research projects. Also a selective data deletion is possible that can be defined by multi-criteria selection, such as object size, age or author.

The remainder of the document is structured as follows. Section 2 presents the main features of CDSTAR. Section 3 provides a technical documentation including a detailed description of CDSTAR's RESTful interface. Section 4 concludes this report by analysing its main results.

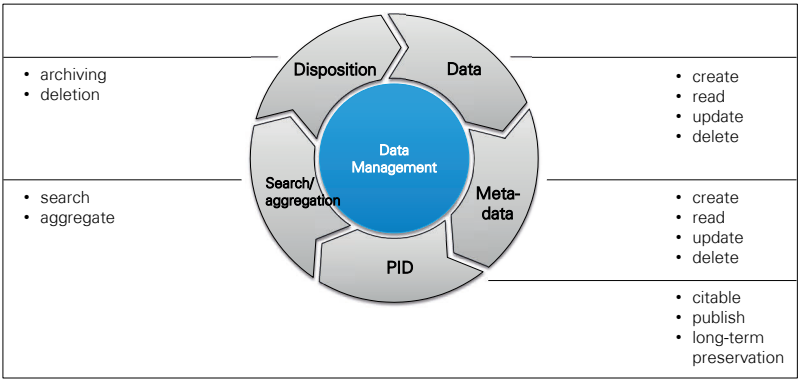


Figure 2: Research data management cycle [9]

2 Features

2.1 Data Hub and Storage

GWDG CDSTAR object storage is an abstraction of storage systems such as hard disks, cloud storage or specialized scientific research data repositories. GWDG CDSTAR offers a unique lightweight REST interface to access different data storage technologies. This allows an access from different applications such as web applications, desktop clients or mobile applications. GWDG CDSTAR is the central data hub for research data management. The CDSTAR server component has access to data backend of choice, suitable for scientific use cases, concerning data size, backup plan, replication, metadata handling or data processing. With choosing CDSTAR researchers can benefit from iRODS [9] as scientific data repository software that allows replication of data to our partner data centers without integrating the iRODS client software. You can also benefit of using the GWDG storage solution for large data sets with caring less about storage space, backup or outsourcing to tape.

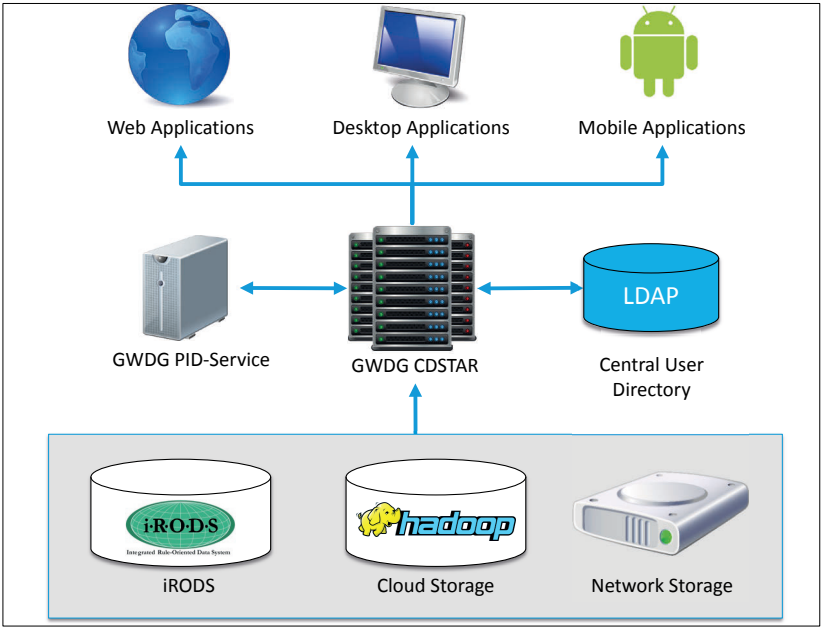


Figure 3: GWDG CDSTAR as data hub

GWDG CDSTAR can also use LDAP servers for user authentication. Hence, user accounts can be used from existing applications or GWDG accounts can be used for data access.

2.2 Management of structured and unstructured Data and its Metadata

GWDG CDSTAR is able to handle all kind of formats and data structures [4]. Research data that is strictly structured with XML is also processed such as semi- or unstructured data. The object store is able to store and retrieve all kind of file formats. File formats that can be parsed by the search engine are incorporated into the full-text enabled search index. Metadata is processed by CDSTAR and all its components in the JSON-format. This means that research metadata can be structured in an easy understandable and readable way. The generation of JSON is possible out of XML or any other file format by using standard libraries that are integrated in almost all programming languages or frameworks. So the difficult task of aligning metadata formats from the research application to the data management system has been made easy with GWDG CDSTAR.

2.3 Enterprise-grade Object Search and Information Retrieval in Science

GWDG CDSTAR also features a very powerful enterprise search for research data and metadata that is fast, scalable and allows users to get new insights into research data. The Enterprise search uses big data technology to scan uploaded data and metadata in real-time for full text content recognition and search over structured metadata. The search engines uses the metadata associated to the research data for creating search results and also incorporates results from the full-text search. The search engine can be queried also via the REST-interface to create sophisticated search queries. With GWDG CDSTAR researchers and information system providers do not only have an object store, they also give the users the possibility to explore existing data and to find data sets. The search engine itself uses industry proven-components that delivery high performance by answering even complex search queries. The search index is also updated in real-time, when data have been uploaded, changed or deleted. With the fast-responsive search engine, usage scenarios as search-as-you-type or recommendation system can be implemented into new and existing scientific application without hassle.

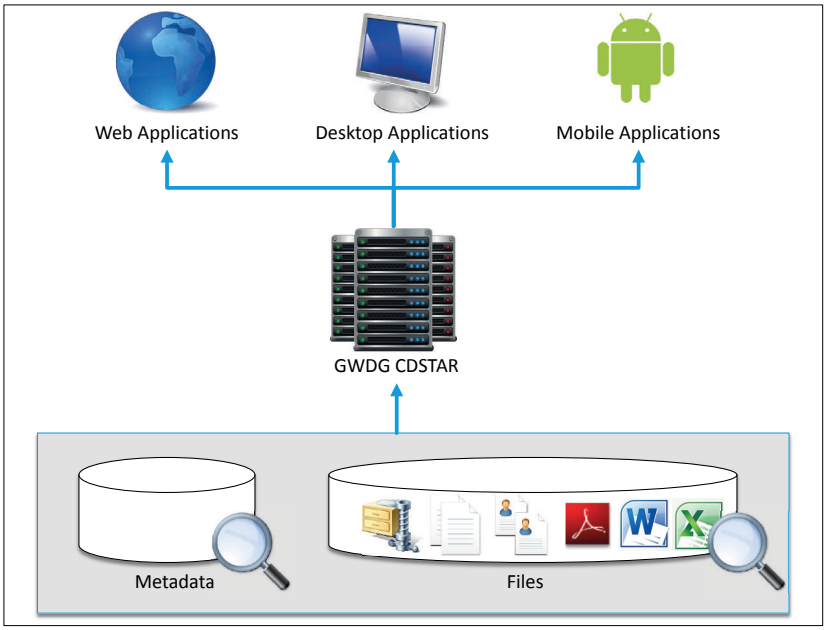


Figure 4: GWDG CDSTAR enterprise search over research data

2.4 Securing the Access to Object Storage

GWDG CDSTAR allows the usage of different mechanism to secure data, metadata and data aggregations. This allows a flexible usage of data according to the need of your organization or project for secure data access and the integration into existing application environments with existing role and data access policies.

2.4.1 Roles and Permissions

Strong default role models and permissions

GWDG CDSTAR has a flexible model for roles and permissions. By default CDSTAR comes with a role model where access to the system, data and metadata can be secured by users owning different roles. With this model users can act in different functions such as researchers, senior or student researcher, team leader, content admins, data curators, operators etc. Hierarchical role models can be transferred directly to CDSTAR. Also by default, the permissions are enforced on object level. This means that the permissions on

CDSTAR objects cover all attached resources such as metadata, bitstreams or collections.

Custom role models and permissions for complex requirements in research

Researchers have often special requirements towards information systems. Therefore special permissions and role system have to be installed to fulfill requirements related to information security. As GWDG CDSTAR supports a modular system of role and permissions custom role models and object permissions can be implemented. With custom role models the level of object security can be extended to individual protection down every associated object resource. The GWDG staff can help researchers, software architects, developers and managers to select and run the right configuration of role models and permissions in the CDSTAR installation to cover the specific needs in information security in research efforts.

2.4.2 Integration with Identity Management

By default GWDG CDSTAR has an application specific database for storing users, roles and passwords. But in order to have a good integration of this data management software into existing and future applications desktop clients, portals, CLI tools and mobile clients) in research and supplementary functions, GWDG CDSTAR also allows integration into LDAP directories. Beside classical LDAP products it also allows integration into Microsoft Active Directory or the GWDG or MPG LDAP infrastructure.

Single sign on allows a smooth switch for the users between the data management and the application without entering password and user name again and again. Single sign on will be available through Shibboleth, the OpenSAML [12] implementation that is widely used in the academic area.[1]

[1] Shibboleth is available as individual option on request.

3 Technical Documentation

GWDG CDSTAR uses a uniform REST-API to manage all operation for data management, administration and search. It also offers an administrative web interface that allows monitoring and browsing CDSTAR and the stored data in the web browser. All services may be accessed from directly over the Internet/LAN or direct point-to-point connection from any application that can send an HTTP/HTTPS request and receive an HTTP/HTTPS response. Although CDSTAR is encrypting its connection default by HTTPS, HTTP may be activated for special scenarios.

3.1 Current Version of the System

The technical documentation currently reflects the features of GWDG CDSTAR Build 140, dated on 12/05/2013. Future releases maintain the compatibility of the technical descriptions below, to ensure software compatibility over the next years and to ensure future operations. Future additions and changes to the features will be communicated in advances and super-seeded REST operations will be marked as deprecated with information about using current REST operations and migration strategy. The development of GWDG CDSTAR aims to have a stable API over a long-term period.

3.2 Core Concepts of GWDG CDSTAR

GWDG CDSTAR provides storage for entities, such as binary files and text files. The five resources supported by the REST API are *objects, bitstreams, metadata, search and access control* modification. Files (in the following called bitstreams) and metadata must belong to an object. CDSTAR offers two type objects plain objects for storing data and metadata and collection-objects for linking different objects. Objects support metadata that can be attached to objects as JSON-file. Using the REST-API plain objects are can store up two million Bitstreams per object and over four billion objects[2]. CDSTAR objects currently do not support partial updates of files, metadata or access rights meaning that every update request has to upload an entire binary stream. Every object is marked by default with a persistent identifier and can be addressed through an URI. Actions on objects are performed by using HTTP methods to provide operations on the resources. For this GWDG CDSTAR uses the acronym CRUD to describe all necessary actions that are

[2] Dependent on the storage backend and its configuration

applicable on REST-resources such as objects, bitstreams, collections and object permissions. The acronym CRUD consists of CREATE (HTTP-Post), READ (HTTP-Get), UPDATE (HTTP-PUT) and DELETE (HTTP-Delete).

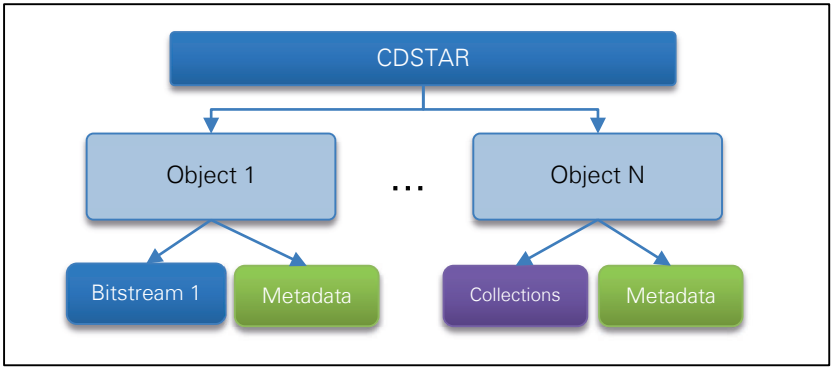


Figure 5: Access to bitstreams, metadata and collections

Generally, the URIs are specified as

```
<scheme>://<service-base-url>/<service_route>/[<UID>]
```

The term `service_route` is used to distinguish between the different API-calls.

3.3 Service-Routes

CDSTAR offers different service routes to interact with the API.

1. **/objects/**
   Creating object and reading object attributes
2. **/bitstreams/**
   Creating, Reading, Updating and Deletion (CRUD) of Bitstreams associated with objects in /objects. 0..n bitstreams can be attached to an object. Bitstreams can only be attached to plain objects – not collections.
3. **/metadata/**
   Creating, Reading, Updating and Deletion (CRUD associated with objects in /objects. Only one set of metadata can be attached to an object.

4. /collections/
   Creating, Reading, Updating and Deletion (CRUD of unidirectional links associated with objects and other collections.
5. /search/
   Get search results of all indexed documents and metadata.
6. /landing/
   Show a HTML-representation of an object or collection with JSON-metadata and hyperlinks to associated bitstreams and and linked objects
7. /accesscontrol/
   Show and manipulates the permissions of an object and all associated bitstreams and metadata.
8. /dariah/
   GWDG CDSTAR also offers access to bitstreams using the DARIAH Storage API – a specialized API for Digital Humanities maintained in the DARIAH project.

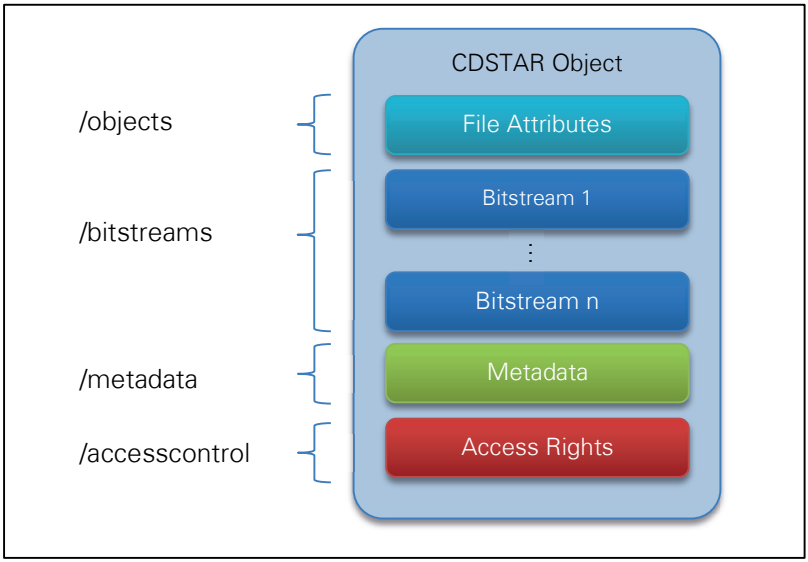


Figure 6: REST-API calls and object structures for plain objects

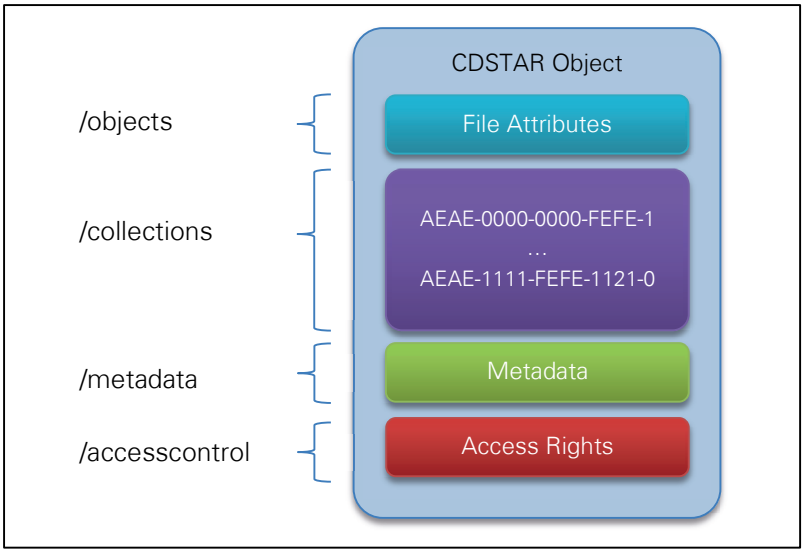


Figure 7: REST-API calls and object structures for collection-objects

3.4 Types of CRUD Operations and API Calls

In the following the basic principles of CRUD operations and API calls are explained. GWDG CDSTAR in synchronous and asynchronous manner. Synchronous means that the success (e.g. HTTP status code 201 for update with put) or fail (HTTP status code 409 for locked objects) of an action is reported immediately to the application invoking the REST-API call [8]. Asynchronous mode means that the result cannot of an action cannot be estimated immediately by CDSTAR. In this case, the API gets a HTTP status code 202 (accepted) [8] that tells the application to check again in a few seconds for the outcome of the API call by invoking the object properties in /objects. The default transfer mode is synchronous. The asynchronous transfer mode is used, if the backend of CDSTAR is facing a high load caused by numerous concurrent POSTs, PUT or DELETE operations. In this case, a successful end of the operation cannot be guaranteed as long as data is still being transferred. The asynchronous transfer mode will be automatically selected by GWDG CDSTAR if necessary.

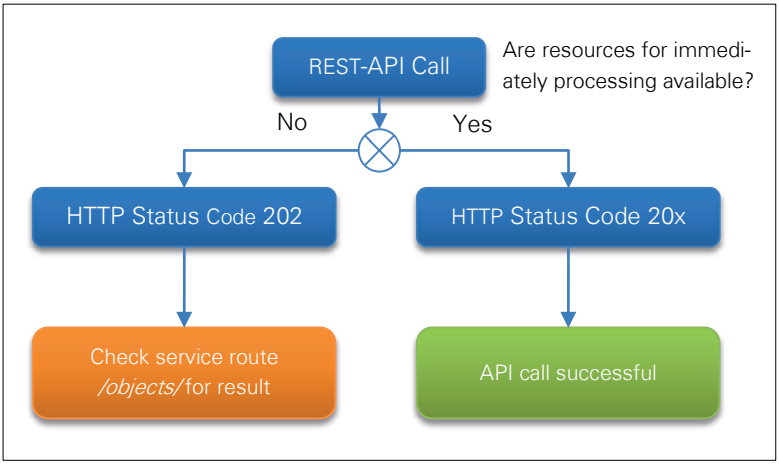


Figure 8: API call processing

The processing mode for all CRUD operations is explained in the following:

3.4.1 Synchronous Transfer with POST/PUT

The request is sent as a POST or PUT operation that contains the bitstream. CDSTAR responds with HTTP status 201 (Created) [8] and sends the location to the resource.

3.4.2 Asynchronous Transfer with POST/PUT

The request is sent as a POST or PUT operation that contains the bitstream. CDSTAR responds with HTTP status 202 (Accepted) [8] and sends HTTP-Headers with the location of a monitor resource. The client must check the monitor resource, to determine whether the operation has finished successfully.

Monitor requests can be performed on any object, no matter whether it is or was transferred synchronously or asynchronously.

3.4.3 Synchronous Processing of DELETE

A DELETE operation for a resource is requested. CDSTAR responds with HTTP-Code 204 (No Content) [8].

3.4.4 Asynchronous Processing of DELETE

A DELETE operation for a resource is requested. CDSTAR responds with HTTP status 202 (Accepted) and sends HTTP-Headers with the location of a monitor resource. The client must check the monitor resource, to determine whether the operation has finished successfully. Monitor requests can be performed on any object, no matter whether it is or was transferred synchronously or asynchronously.

3.5 Locking and concurrent Access

GWDG CDSTAR performs locking to avoid impairments of concurrent operations on the same object. Locking is implemented on object base. This means, an object and all associated resources (bitstreams, metadata and permissions) are locked, if a corresponding creation or update operation is pending. In general, all operations that alter the object's content or its state will cause a lock. However, parallel read accesses are always possible, if the object is not locked. GWDG CDSTAR follows the paradigm of optimistic locking, which implies that massive parallel reading is fast, while updates are comparatively costly. For instance, if client A updates the bitstream of an object, client B cannot request the object attributes of the same objects until Client X finished its operation. Following operations cause a lock of an object:

1. CREATE Object
2. UPDATE Object
3. DELETE Object
4. CREATE Bitstream
5. UPDATE Bitstream
6. DELETE Bitstream
7. CREATE Metadata
8. UPDATE Metadata
9. DELETE Metadata
10. CREATE Collection
11. UPDATE Collection
12. DELETE Collection
13. CHANGE Access-Control

3.6 Timestamping of Objects

The POSIX time format is used for all fields that represent a point in time; for example the last-modified fields. POSIX time is defined as the number of seconds that have elapsed since January 1, 1970, 00:00:00 GMT, not counting leap seconds. The time format is defined, widely used and well support in many standard tools, programming languages and frameworks. The timestamps are locale independent (no named months or weekdays as stated in [13], no attached time-zone, just a number). Reading and writing POSIX timestamps in various languages can be archived in following ways.

3.7 REST Operations

3.7.1 Objects

3.7.1.1 CREATE

Creation of objects is done by posting an empty data set to the resource /objects. By default, the service creates a plain object that is capable of holding bitstreams and metadata. To create collection-objects please refer to the section create at the chapter collections.

Request	
Service route	/objects/
HTTP verb	POST
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)

The operation was successful. The object has been created. For the new object a UID has been generated by the system.

Response	
Status	201 Created
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	{“uid”:<UID>, “ok”:”true”}

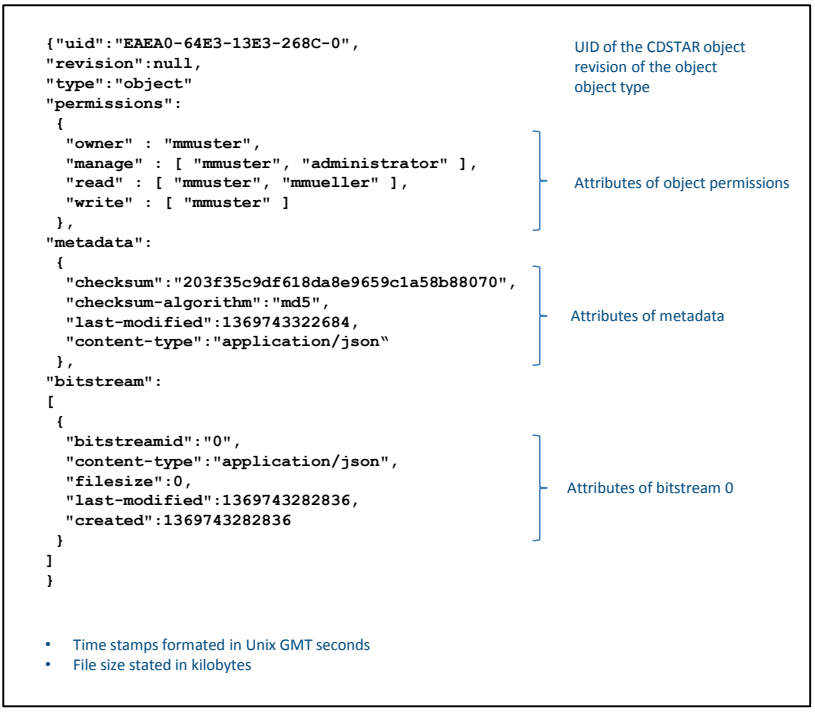
A problem fulfilling the request has been occurred. The client should try to access the object once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	<Error message>

3.7.1.2 READ

Reading objects shows the object attributes of an object. Objects are identified by a unique ID called UID. Every UID is identical to the Handle-PID consisting of a Handle prefix and a suffix, generated by the GWDG PID-service. The object attributes consists of information about the associated bitstreams, metadata, permissions.

Request	
Service Route	/objects/<UID>
HTTP verb	GET
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)



Listing 1: JSON object representation

The action is successful and the object attributes are delivered to the client.

Response	
Status	200 OK
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	Object attributes (see below)

The action failed due to insufficient permissions. The permissions on reading the object is needed to fulfill the action. Update the permissions of the object in order to get access to the resource or choose a different user with permissions on reading this object

Errors	
Status	403 forbidden
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”forbidden”, “reason”:”permission_denied”}

The action failed. The object was not found by GWDG CDSTAR. The object might be deleted in the meantime by concurrent accesses or a wrong UID has been submitted.

Errors	
Status	404 Object not found
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”not_found”, “reason”:”missing”}

The object or associated resources (bitstreams, metadata or access rights) is currently updated by a concurrent access. If the error 409 is received by client, the client should check the object later.

Errors	
Status	409 Conflict
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"conflict", "reason":"object update in progress"}

A problem fulfilling the request has been occurred. The client should try to access the object once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	<Error message>

3.7.1.3 UPDATE

Objects and their associated object attributes are maintained by CDSTAR automatically. Hence, CDSTAR only provides a GET and DELETE method. Manipulating File attributes relies on calling the specific API.

3.7.1.4 DELETE

If the deletion of an object is processed, all bitstreams metadata, permissions of the object are deleted as well. For deleting specific resources of an object use the respective delete API calls.

Request	
Service Route	/objects/<UID>
HTTP verb	DELETE
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)

If the action was successful and the object has been deleted from the server.

Response	
Status	204 No Content
Last Modified	DateTime
Cache-Control	no-store, no-cache
Body	<empty>

The action failed due to insufficient permissions. The permissions on writing the object is needed to fulfill the action. Update the permissions of the object in order to get access to the resource or choose a different user with permissions on writing this object.

Errors	
Status	403 forbidden
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”forbidden”, “reason”:”permission_denied”}

The action failed. The object was not found by GWDG CDSTAR. The object might be deleted in the meantime by concurrent accesses or a wrong UID has been submitted.

Errors	
Status	404 Object not found
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”not_found”, “reason”:”missing”}

The object or associated resources (bitstreams, metadata or access rights) is currently updated by a concurrent access. If the error 409 is received by client, the client should check the object later.

Errors	
Status	409 Conflict
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"conflict", "reason":"object update in progress"}

A problem fulfilling the request has been occurred. The client should try to access the object once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	<Error message>

3.7.2 Bitstreams

In the following the operations for attaching, updating and deleting bitstreams in objects are explained. Bitstreams are identified by bitstream IDs that are generated by GWDG CDSTAR by incrementing the last known bitstream ID by one. To access a bitstream a combination of the object UID and the bitstream id is necessary. GWDG CDSTAR generates for new bitstreams new bitstream IDs. GWDG CDSTAR never reuses bitstream IDs that have been used in an object.

3.7.2.1 CREATE

The content of a file is posted as body payload of an HTTP-request to the bitstream service route under the usage of an existing object UID and the explanation of the file mime-type. As a result, the bitstream is stored in the system and associated with the object.

Request	
Service route	/bitstreams/<UID>/
HTTP verb	POST
Data	<binary content>
Content-Type	Mime-Type of bitstream
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)

The operation was successful. The bitstreams has been stored. A bitstream ID has been generated.

Response	
Status	201 Created
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	{“uid”:<UID>, “bitstreamid”:<bitstreamid> “ok”:”true”}

The action failed due to insufficient permissions. The permissions on writing the object is needed to fulfill the action. Update the permissions of the object in order to get access to the resource or choose a different user with permissions on writing this object.

Errors	
Status	403 forbidden
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”forbidden”, “reason”:”permission_denied”}

The action failed. The object was not found by GWDG CDSTAR. The object might be deleted in the meantime by concurrent accesses or a wrong UID has been submitted.

Errors	
Status	404 Object not found
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”not_found”, “reason”:”missing”}

The object or associated resources (bitstreams, metadata or access rights) is currently updated by a concurrent access. If the error 409 is received by client, the client has to try the upload again later.

Errors	
Status	409 Conflict
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"conflict", "reason":"object update in progress"}

A problem fulfilling the request has been occurred. The client should try to upload the data once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	<Error message>

3.7.2.2 READ

To retrieve the binary content of a stored bitstream the object UID and the bitstream ID is needed.

Request	
Service route	/bitstreams/<UID>/ <bitstream ID>
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)

The action is successful and the object attributes are delivered to the client.

Response	
Status	200 OK
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	<Content Type of bitstream>
Body	<binary content>

The action failed due to insufficient permissions. The permissions on reading the object is needed to fulfill the action. Update the permissions of the object in order to get access to the resource or choose a different user with permissions on reading this object.

Errors	
Status	403 forbidden
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”forbidden”, “reason”:”permission_denied”}

The action failed. The object was not found by GWDG CDSTAR. The object might be deleted in the meantime by concurrent accesses or a wrong UID has been submitted.

Errors	
Status	404 Object not found
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”not_found”, “reason”:”missing”}

The object or associated resources (bitstreams, metadata or access rights) is currently updated by a concurrent access. If the error 409 is received by client, the client has to try the upload again later.

Errors	
Status	409 Conflict
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"conflict", "reason":"object update in progress"}

A problem fulfilling the request has been occurred. The client should try to upload the data once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	<Error message>

3.7.2.3 UPDATE

The content of a file is putted as body payload of an HTTP-request to the bitstream service route under the usage of an existing object UID, an existing bitstream ID and the explanation of the file mime-type. As a result, the bitstream is stored in the system and associated with the object.

Request	
Service route	/bitstreams/<UID>/ <bitstream ID>
HTTP verb	PUT
Data	<binary content>
Content-Type	Mime-Type of bitstream
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)

The operation was successful. The bitstreams has been stored. A bitstream ID has been generated.

Response	
Status	201 Created
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	{“uid”:<UID>, “bitstreamid”:<bitstreamid> “ok”:”true”}

The action failed due to insufficient permissions. The permissions on writing the object is needed to fulfill the action. Update the permissions of the object in order to get access to the resource or choose a different user with permissions on writing this object.

Errors	
Status	403 forbidden
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”forbidden”, “reason”:”permission_denied”}

The action failed. The object was not found by GWDG CDSTAR. The object might be deleted in the meantime by concurrent accesses or a wrong UID has been submitted.

Errors	
Status	404 Object not found
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”not_found”, “reason”:”missing”}

The object or associated resources (bitstreams, metadata or access rights) is currently updated by a concurrent access. If the error 409 is received by client, the client has to try the upload again later.

Errors	
Status	409 Conflict
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"conflict", "reason":"object update in progress"}

A problem fulfilling the request has been occurred. The client should try to upload the data once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	<Error message>

3.7.2.4 DELETE

For all deleted bitstreams, the object attributes show a null value in the array of bitstreams stored in the object attributes that are retrievable by the /object service route. The usage of null references marks bitstreams ID as previously used.

Request	
Service Route	/bitstreams/<UID>/ <bitstream ID>
HTTP verb	DELETE
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)

If the action was successful the bitstream is deleted from the server and the association to the file is deleted from the GWDG CDSTAR object.

Response	
Status	204 No Content
Last Modified	DateTime
Cache-Control	no-store, no-cache
Body	<empty>

The action failed due to insufficient permissions. The permissions on writing the object is needed to fulfill the action. Update the permissions of the object in order to get access to the resource or choose a different user with permissions on writing this object.

Errors	
Status	403 forbidden
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”forbidden”, “reason”:”permission_denied”}

The action failed. The object or the bitstream was not found by GWDG CDSTAR. The object or the bitstreams might be deleted in the meantime by concurrent accesses or a wrong UID or bitstreams ID has been submitted.

Errors	
Status	404 Object not found
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”not_found”, “reason”:”missing”}

The object or associated resources (bitstreams, metadata or access rights) is currently updated by a concurrent access. If the error 409 is received, the client should try to delete the bitstream later.

Errors	
Status	409 Conflict
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"conflict", "reason":"object update in progress"}

A problem fulfilling the request has been occurred. The client should try to access the object once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	<Error message>

3.7.3 Metadata

Metadata in GWDG CDSTAR use the service route /metadata for accessing metadata. Only one set of metadata can be added to an object. GWDG CDSTAR only processes JSON-formatted metadata. Therefore, the setting of the mime-type application/JSON is mandatory for creating and updating metadata sets. GWDG CDSTAR verifies and parses all uploaded metadata sets, if the comply with the JSON-format specification. Further semantic verification is not performed.

3.7.3.1 CREATE

The content of a metadata set is posted as body payload of an HTTP-request to the metadata service route under the usage of an existing object UID and the nomination of the content type application/JSON. As a result, the metadata set is stored in the system and associated with the object.

Request	
Service route	/metadata/<UID>
HTTP verb	POST
Data	<binary content>
Content-Type	Application/JSON
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)

The operation was successful. The metadata has been stored.

Response	
Status	201 Created
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	{“uid”:<UID>, “ok”:”true”}

The action failed due to insufficient permissions. The permissions on writing the object is needed to fulfill the action. Update the permissions of the object in order to get access to the resource or choose a different user with permissions on writing this object.

Errors	
Status	403 forbidden
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”forbidden”, “reason”:”permission_denied”}

The action failed. The object was not found by GWDG CDSTAR. The object might be deleted in the meantime by concurrent accesses or a wrong UID has been submitted.

Errors	
Status	404 Object not found
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”not_found”, “reason”:”missing”}

The object or associated resources (bitstreams, metadata or access rights) is currently updated by a concurrent access. If the error 409 is received by client, the client has to try the upload again later.

Errors	
Status	409 Conflict
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"conflict", "reason":"object update in progress"}

A problem fulfilling the request has been occurred. The client should try to upload the data once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	<Error message>

3.7.3.2 READ

Reading objects shows the metadata set stored for an object.

Request	
Service Route	/metadata/<UID>
HTTP verb	GET
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)

The action is successful and the metadata set is delivered to the client.

Response	
Status	200 OK
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	<metadata content>

The action failed due to insufficient permissions. The permissions on reading the object is needed to fulfill the action. Update the permissions of the object in order to get access to the resource or choose a different user with permissions on reading this object.

Errors	
Status	403 forbidden
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”forbidden”, “reason”:”permission_denied”}

The action failed. The object was not found by GWDG CDSTAR. The object might be deleted in the meantime by concurrent accesses or a wrong UID has been submitted.

Errors	
Status	404 Object not found
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”not_found”, “reason”:”missing”}

The object or associated resources (bitstreams, metadata or access rights) is currently updated by a concurrent access. If the error 409 is received by client, the client should check the object later.

Errors	
Status	409 Conflict
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"conflict", "reason":"object update in progress"}

A problem fulfilling the request has been occurred. The client should try to access the object once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	<Error message>

3.7.3.3 UPDATE

The content of a metadata is putted as body payload of an HTTP-request to the metadata service route under the usage of an existing object UID. As a result, the metadata is stored in the system and associated with the object.

Request	
Service route	/metadata/<UID>
HTTP verb	PUT
Data	<binary content>
Content-Type	Application/JSON
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)

The operation was successful. The metadata set has been stored.

Response	
Status	201 Created
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	{“uid”:<UID>, “ok”:”true”}

The action failed due to insufficient permissions. The permissions on writing the object is needed to fulfill the action. Update the permissions of the object in order to get access to the resource or choose a different user with permissions on writing this object.

Errors	
Status	403 forbidden
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"forbidden", "reason":"permission_denied"}

The action failed. The object was not found by GWDG CDSTAR. The object might be deleted in the meantime by concurrent accesses or a wrong UID has been submitted.

Errors	
Status	404 Object not found
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"not_found", "reason":"missing"}

The object or associated resources (bitstreams, metadata or access rights) is currently updated by a concurrent access. If the error 409 is received by client, the client has to try the upload again later.

Errors	
Status	409 Conflict
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"conflict", "reason":"object update in progress"}

A problem fulfilling the request has been occurred. The client should try to upload the data once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	<Error message>

3.7.3.4 DELETE

If the deletion of a metadata set is processed, the JSON-formatted data is deleted from the server and the association to the object is also deleted.

Request	
Service Route	/bitstreams/<UID>
HTTP verb	DELETE
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)

If the action was successful and the metadata set has been deleted from the server.

Response	
Status	204 No Content
Last Modified	DateTime
Cache-Control	no-store, no-cache
Body	<empty>

The action failed due to insufficient permissions. The permissions on writing the object is needed to fulfill the action. Update the permissions of the object in order to get access to the resource or choose a different user with permissions on writing this object.

Errors	
Status	403 forbidden
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”forbidden”, “reason”:”permission_denied”}

The action failed. The object was not found by GWDG CDSTAR. The object might be deleted in the meantime by concurrent accesses or a wrong UID has been submitted.

Errors	
Status	404 Object not found
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”not_found”, “reason”:”missing”}

The object or associated resources (bitstreams, metadata or access rights) is currently updated by a concurrent access. If the error 409 is received by client, the client should check the object later.

Errors	
Status	409 Conflict
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"conflict", "reason":"object update in progress"}

A problem fulfilling the request has been occurred. The client should try to access the object once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	<Error message>

3.7.4 Search

The REST API calls for the search engine encapsulate the search engine and use the domain specific query language of the software project *elasticsearch* [14] to perform real-time search operations.

3.7.4.1 Specify Search Sources

There are three possibilities to search over the index:

1. A Search over metadata and full text search. This is the default case and requires no query parameter
2. A Search over metadata only. This has to be specified by the query parameter *indexselection=metadata*.
3. A Search over the full text only. This has to be specified by the query parameter *indexselection=fulltext*.

The query parameters *limit* and *offset* are used to browse through the search result and to limit search depth. Limit sets the number of search result. The parameter offset allows to skip entries the in search. By default, the offset is set to 0 in order to return the search result from the beginning. With the offset parameter, paged browsing of search results can be realized e.g. in an AJAX application.

NOTE: By default limits is set to 15, meaning that the search only return 15 result in order to perform fast. If you need more search results, increment the limit query parameter.

3.7.4.2 Formulate Search Queries

The query language is based on Apache Lucene and is provided from elasticsearch [15]. The full possibilities can be found on http://www.elasticsearch.org/guide/reference/query-dsl/

A simple full text query can be done via query_string:

```
{
    "query_string" : {
        "query" : "foo"
     }
}
```

Listing 2: Simple query for full text search

To access metadata, Boolean and text queries are also useful. Dependent on the metadata fields a Boolean and text query can be done via:

```
{
    "bool": {
        "must": [
        {
            "text": {
            "title": "dolorem"
            }
        },
        {
            "text": {
            "titleValue": "Alienus"
            }
        }
        ]
    }
}
```

Listing 3: Query covering Boolean search expression and full text

3.7.4.3 Search on full text index and metadata

To search over the full text index and the metadata index perform following command.

Request	
Service route	/search/
Query parameter	Indexselection (optional)
Query parameter	limit (optional)
Query parameter	offset (optional)
HTTP verb	POST
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)
Content-Type	Application/JSON
Body	<search query>

The operation was successful. The object has been created. For the new object a UID has been generated by the system.

Response	
Status	200 OK
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	<search results> (see below)

If the search has been successful and search results exist, the search results are delivered back as JSON-formatted string (see next page). If no matching results have been found, the result is:

```
{"hits":[],"totalhits":0,"maxscore":"0.0"}
```

Listing 4: JSON-formatted search results with no hits

```
{
    "hitcount": 4,
    "maxscore": 1.4662267,
    "hits": [
        {
            "source": "production",
            "type": "metadata",
            "score": 0.7296878,
            "uid": "EAEA0-23A9-6D2C-7567-0"
        },
        {
            "source": "production",
            "type": "metadata",
            "score": 0.7296878,
            "uid": "EAEA0-7B37-D9E3-E627-0"
        },
        {
            "source": "production",
            "type": "metadata",
            "score": 0.8756254,
            "uid": "EAEA0-C8DB-788C-7E09-0"
        },
        {
            "source": "production",
            "type": "metadata",
            "score": 1.4662267,
            "uid": "EAEA0-9958-B32F-4CA3-0"
        }
    ]
}
```


Listing 5: JSON-formatted search result with hits

The search result produces following results.

JSON-Fields (search result)	
hitcount	Number of found results (hits)
maxscore	Score of the result with the highest relevancy
hits	Array of JSON hit-objects

JSON-Fields (hit object)	
source	Name of the search index, where the result has been found. By default, there is only one index.
type	Indication of the search result type (metadata or fulltext)
score	Relevancy of the search result in relation to the search
uid	UID of the object where the result has been found

A problem fulfilling the request has been occurred. The client should try to access the object once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	<Error message>

3.7.4.4 Search on metadata only

To search over the metadata index only perform following command.

Request	
Service route	/search/metadata/
HTTP verb	POST
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)
Content-Type	Application/JSON

The success and error messages are identical to the results shown above.

3.7.5 Collections

3.7.5.1 Introduction

In GWDG CDSTAR every object has a unique identifier, called UID. GWDG CDTSTAR supports a semantic connection of multiple objects through the mechanism of collections. Collections are special objects that contain instead of files/bitstreams a list of UIDs. This link list can point to CDSTAR-objects and even to other CDSTAR collection-objects. With this mechanism, concepts like folders, object trees or networks can be modeled and used within GWDG STAR. As collection also store metadata these collections can be filled with semantically information. Hence users can create views or link lists on entire sets of object. This is very useful to implement many scientific use cases with GWDG CDSTAR.

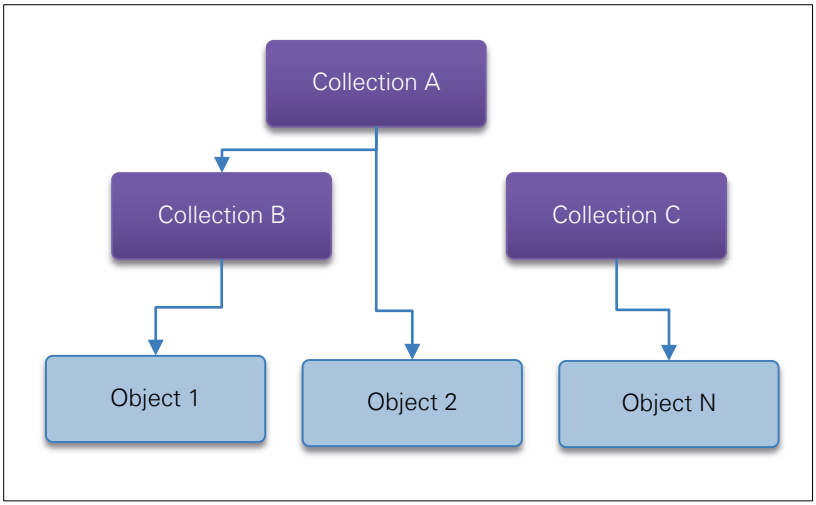


Figure 9: Structure example of GWDG CDSTAR collections

3.7.5.2 CREATE

Creation of objects is done by posting an empty data set to the resource /objects. To create collection-objects with the ability to hold collections the query parameter type=collection has to be appended to the service route.

Request	
Service route	/objects/
Query-parameter for creating a collection-object	type=collection (optional)
HTTP verb	POST
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)

The operation was successful. The object has been created. For the new object a UID has been generated by the system.

Response	
Status	201 Created
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	{“uid”:<UID>, “ok”:”true”}

A problem fulfilling the request has been occurred. The client should try to access the object once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	<Error message>

3.7.5.3 READ Collection Object

Like normal GWDG CDSTAR objects, collections can also be read and displayed by using the service route /objects. Please refer to the section objects READ at the chapter REST operations.

3.7.5.4 READ Collection Content

The return value of collection content is a JSON-formatted array of UIDs. When a collection-object has been created, the collection array is initially empty.

Request	
Service Route	/collections/<UID>
HTTP verb	GET
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)

The action is successful and the array of UIDs is delivered to the client.

Response	
Status	200 OK
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{[<UID_1>,<UID_2>, ...]}

The action failed due to insufficient permissions. The permissions on reading the object is needed to fulfill the action. Update the permissions of the object in order to get access to the resource or choose a different user with permissions on reading this object.

Errors	
Status	403 forbidden
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”forbidden”, “reason”:”permission_denied”}

The action failed. The collection-object was not found by GWDG CDSTAR. The object might be deleted in the meantime by concurrent accesses or a wrong UID has been submitted.

Errors	
Status	404 Object not found
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”not_found”, “reason”:”missing”}

The collection-object or associated resources (bitstreams, metadata or access rights) is currently updated by a concurrent access. If the error 409 is received by client, the client should check the object later.

Errors	
Status	409 Conflict
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"conflict", "reason":"object update in progress"}

A problem fulfilling the request has been occurred. The client should try to access the object once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	<Error message>

3.7.5.5 UPDATE

To update the collection, an entire array of UIDs has to be put to the collection-object. It is not possible to remove a single UID with an API call. To remove one or multiple UIDs from a collection, the client has to read the collection, then remove the respective UIDs from the array and then update the collection by putting the updated array back to the collection-object. This paradigm of full-updates for data sets follows the same principle in CDSTAR that applies on metadata and bitstreams. Hence, to remove all UID links to other object stored in a collection-object, simply put an empty array to the collection service route by using the collection-object UID.

Request	
Service route	/collections/<UID>
HTTP verb	PUT
Data	{[<UID_1>,<UID_2>, ...]}
Content-Type	application/JSON
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)

The operation was successful. The collection set has been update in the collection-object.

Response	
Status	201 Created
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	{“uid”:<UID>, “ok”:”true”}

The action failed due to insufficient permissions. The permissions on writing the collection-object is needed to fulfill the action. Update the permissions of the object in order to get access to the resource or choose a different user with permissions on writing this object.

Errors	
Status	403 forbidden
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”forbidden”, “reason”:”permission_denied”}

The action failed. The object was not found by GWDG CDSTAR. The object might be deleted in the meantime by concurrent accesses or a wrong UID has been submitted.

Errors	
Status	404 Object not found
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”not_found”, “reason”:”missing”}

The object or associated resources (collection, metadata or access rights) is currently updated by a concurrent access. If the error 409 is received by client, the client has to try the upload again later.

Errors	
Status	409 Conflict
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"conflict", "reason":"object update in progress"}

A problem fulfilling the request has been occurred. The client should try to upload the data once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	<Error message>

3.7.5.6 DELETE

Like normal GWDG CDSTAR objects, collections can also be deleted by using the service route /objects. Please refer to the section objects DELETE at the chapter REST operations.

3.7.6 Object Permissions

An introduction into the default role and permissions model is given in the Feature chapter in the section secure access. As GWDG CDSTAR can be equipped with custom modes, this section only describes the interaction with the default role bases model that comes with every standard instance of GWDG CDSTAR. With this default model users can act in different functions such as researchers, senior or student researcher, team leader, content admins, data curators, operators etc. Hierarchical role models can be transferred directly to CDSTAR. Also by default, the permissions are enforced on object level. This means that the permissions on CDSTAR objects cover all attached resources such as metadata, bitstreams or collections. Therefore no separate permissions can be set for bitstreams and metadata in the same object. However, this feature can be archived by separating metadata and bitstreams into single objects and using collections to tie the up.

In the default role model assigns to every user a default group that is identical to his user id.[3] This group should not be shared with other users. Also by default every object has default permissions and a default owner group. By default, the owner is the creator of an object.

Every CDSTAR object and collection has four attributes to control the access on reading, writing and maintenance.

1. READ – 0-n roles that can read the object and all metadata, collections or bitstreams
2. WRITE – 0-n roles that can write the object and all metadata, collections or bitstreams
3. OWNER – 1 role that owns the object - can change the permissions of the object
4. MANAGE – 0-n roles can change the permissions of the object and the owner of the object

To manage the permissions in an object, GWDG CDSTAR uses a JSON-formatted permission structure that is available under the key permissions in every object. The structure is returned, when the permission of an object are read or set.

[3] The default settings depend on the configuration of the GWDG CDSTAR instance.

```
{
    "owner" : "user",
    "manage" : ["admin", "user"],
    "read" : ["admin", "guest", "user"],
    "write": ["user", "portaluser"]
}
```

Listing 6: Object permissions for CDSTAR objects

3.7.6.1 READ

There are two ways how to read the permissions of an object. First, the permissions are return as part of the object attributes, when using a GET request on the /object route by using the UID. Please have a look at the REST operation chapter with the section on reading objects.

The recommended and second way is to use the /accesscontrol/ service route to read and set object permissions.

Request	
Service Route	/accesscontrol/<UID>
HTTP verb	GET
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)

The action is successful and the object attributes are delivered to the client.

Response	
Status	200 OK
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	<object permissions>

The action failed due to insufficient permissions. The permissions on reading the object is needed to fulfill the action. Update the permissions of the object in order to get access to the resource or choose a different user with permissions on reading this object.

Errors	
Status	403 forbidden
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”forbidden”, “reason”:”permission_denied”}

The action failed. The object was not found by GWDG CDSTAR. The object might be deleted in the meantime by concurrent accesses or a wrong UID has been submitted.

Errors	
Status	404 Object not found
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”not_found”, “reason”:”missing”}

The object or associated resources (bitstreams, metadata or access rights) is currently updated by a concurrent access. If the error 409 is received by client, the client should check the object later.

Errors	
Status	409 Conflict
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"conflict", "reason":"object update in progress"}

A problem fulfilling the request has been occurred. The client should try to access the object once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	<Error message>

3.7.6.2 SET

Request	
Service route	/accesscontrol/<UID>
HTTP verb	PUT
Data	<binary content>
Content-Type	Application/JSON
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)

The operation was successful. The metadata set has been stored.

Response	
Status	201 Created
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	<object permissions>

The action failed due to insufficient permissions. The permissions on writing the object is needed to fulfill the action. Update the permissions of the object in order to get access to the resource or choose a different user with permissions on writing this object.

Errors	
Status	403 forbidden
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"forbidden", "reason":"permission_denied"}

The action failed. The object was not found by GWDG CDSTAR. The object might be deleted in the meantime by concurrent accesses or a wrong UID has been submitted.

Errors	
Status	404 Object not found
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"not_found", "reason":"missing"}

The object or associated resources (bitstreams, metadata or access rights) is currently updated by a concurrent access. If the error 409 is received by client, the client has to try the upload again later.

Errors	
Status	409 Conflict
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"conflict", "reason":"object update in progress"}

A problem fulfilling the request has been occurred. The client should try to upload the data once again.

Errors	
Status	503 Service Temporarily Unavailable
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	application/JSON
Body	<Error message>

3.7.7 DARIAH Storage API

GWDG CDSTAR also supports access to bitstreams with the DARIAH Storage API – a specialized API for digital Humanities maintained in the DARIAH project [16]. Currently support for Version 1.0 of the API is offered. All mandatory and optional fields are supported. The support for Shibboleth and the associated PAOS-Header support will follow in late summer 2013.

To obtain a copy of the DARIAH storage specification as DARIAH-involved person can either consult the DARIA wiki (https://dev2.dariah.eu/wiki/) or other persons may address their request to the DARIAH-DE project (https://portal-de.dariah.eu/).

3.7.8 Landing Pages

Landing pages are the HTML-representation of an object or collection with the rendered JSON-metadata, collections and hyperlinks to associated bitstreams. Landing pages can be retrieved through the web browser by using the service route /landing/<UID>.

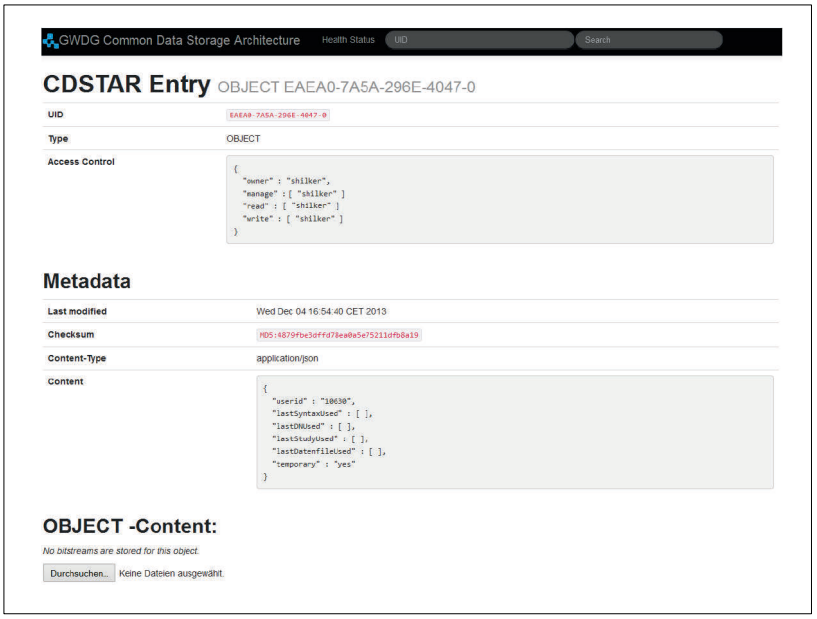


Figure 10: Object landing page in web browser

Reading objects shows the metadata set stored for an object.

Request	
Service Route	/landing/<UID>
HTTP verb	GET
Authorization	Basic (optional)
Authorization	PAOS-Headers (optional)
Cookie	Session-ID (optional)
Version	API-Version-String (optional)

The action is successful and the metadata set is delivered to the client.

Response	
Status	200 OK
Last Modified	DateTime
Cache-Control	no-store, no-cache
Location	UID of the created object
Content-Type	Text/html
Body	Landing page content

The action failed due to insufficient permissions. The permissions on reading the object is needed to fulfill the action. Update the permissions of the object in order to get access to the resource or choose a different user with permissions on reading this object.

Errors	
Status	403 forbidden
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”forbidden”, “reason”:”permission_denied”}

The action failed. The object was not found by GWDG CDSTAR. The object might be deleted in the meantime by concurrent accesses or a wrong UID has been submitted.

Errors	
Status	404 Object not found
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{“error”:”not_found”, “reason”:”missing”}

The object or associated resources (bitstreams, metadata or access rights) is currently updated by a concurrent access. If the error 409 is received by client, the client should check the object later.

Errors	
Status	409 Conflict
Last Modified	DateTime
Cache-Control	no-store, no-cache
Content-Type	application/JSON
Body	{"error":"conflict", "reason":"object update in progress"}

A problem fulfilling the request has been occurred. The web browser should try to access the object once again.

4 Conclusion

After going through all features, requirements and technical documentations, it's time to sum up all features. When using GWDG CDSTAR, scientists, researchers, information system designers and software developers get a powerful data management solution that is research data management enabled by design. GWDG CDSTAR is stable, easy to integrate into existing and new software and boosts the development results. It can be used as man data management software in project, integration middle ware or archiving solution for long-term access. For more information and question about GWDG CDSTAR please contact the GWDG CDSTAR team (see contact section).

5 Contact

Consulting about GWDG CDSTAR, research data management, long-term archiving and further GWDG services supporting your research data efforts can be provided at any time. Also a demonstration setup for GWDG CDSTAR can be installed within days. For further information and questions please contact following responsible persons at the GWDG.

Technical Consulting & Lead Developer GWDG CDSTAR
Oliver Schmitt
Phone: + 49 551 39-20512
E-Mail: oliver.schmitt@gwdg.de

Data management and long-term archiving
Dr. Ulrich Schwardmann
Phone: + 49 551 201-1542
E-Mail: ulrich.schwardmann@gwdg.de

6 Acknowledgements

Regarding the creation of GWDG CDSTAR we like to acknowledge the ideas and additions of following persons. Christof Pohl, Stephan Hilker and Dr. Christian Boehme from the GWDG for the provision of use cases in the project *socioeconomic reporting*. Also the GWDG data management team for discussing the concepts and ideas. Zeljko Carevic from the GESIS - Leibniz-Institute for the Social Sciences for testing extensively the REST-Interface. Bartłomiej Marzec, Bachelor student at the Department of Medical Informatics Göttingen for testing and regularly bug reporting. Danah Tonne and Francesca Rindone from SWM, KIT for providing their ideas on DARIAH storage API support. Maik Srba from the Cloud4E project at the GWDG for providing the necessary concepts and implementations on token-based authentication and finally Dr. Tim A. Majchrzak, Practical Computer Science Group, Department of Information System, University Münster for sharing ideas and feedback on handling JSON-based metadata schemas and map-reduce-based data retrieval operations.

7 Literature

[1] Amazon Web Services, Inc. (2013, June) Amazon S3, Cloud Computing Storage for Files, Images, Videos. [Online]. http://aws.amazon.com/s3/

[2] Microsoft Corporation. (2013, June) Blob Service REST API. [Online]. http://msdn.microsoft.com/en-us/library/windowsazure/dd135733.aspx

[3] Deutsche Forschungsgemeinschaft. (1998) Vorschläge zur Sicherung gute Wissenschaftlicher Praxis. [Online]. http://www.dfg.de/download/pdf/dfg_im_profil/reden_stellungnahmen/download/empfehlung_wiss_praxis_0198.pdf

[4] The Apache Software Foundation. (2013, June) Apache Tika - Supported Document Formats. [Online]. http://tika.apache.org/0.10/formats.html

[5] R. Taylor and R. Fielding, "Principled design of the modern Web architecture," *ACM Trans. Internet Technol.*, pp. 115-150, May 2002.

[6] European Persistent Identifier Consortium. (2013, June) EPIC - European Persistent Identifier Consortium. [Online]. http://pidconsortium.eu/

[7] Network Working Group. (2006, June) Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map. [Online]. http://tools.ietf.org/html/rfc4510

[8] J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee and R. Fielding (1999, April) Hypertext Transfer Protocol -- HTTP/1.1. [Online]. http://www.w3.org/Protocols/rfc2616/rfc2616.html

[9] D. Crockford. (2006, July) The application/json Media Type for JavaScript Object Notation (JSON). [Online]. http://www.ietf.org/rfc/rfc4627.txt

[10] University of Essex. (2013, June) UK Data Archive - RESEARCH DATA LIFECYCLE. [Online]. http://data-archive.ac.uk/create-manage/life-cycle

[11] iRODS Consortium. (2013, June) IRODS: Data Grids, Digital Libraries, Persistent Archives, and Real-time Data Systems. [Online]. http://www.irods.org

[12] Shibboleth Consortium. (2013, June) Shibboleth. [Online]. http://shibboleth.net/

[13] P. Resnick. (2001, April) Internet Message Format, Network Working Group RFC 2822. [Online]. http://www.ietf.org/rfc/rfc2822.txt

[14] Elasticsearch Global BV. (2013, June) Open Source Distributed Real Time Search & Analytics | Elasticsearch. [Online]. http://www.elasticsearch.org/

[15] Elasticsearch Global BV. (2013, June) Query Dsl | Reference Guide | Elasticsearch. [Online]. http://www.elasticsearch.org/guide/reference/query-dsl/

[16] DARIAH-EU Coordination Office. (2013, June) DARIAH - Digital Research Infrastructure for the Arts and Humanities. [Online]. http://www.dariah.eu/