GNDMS HTTP REST Service API

(Specification DRAFT, Version 0.4.1)

Stefan Plantikow, Maik Jorra, Ulrike Golas

Table of Contents

Introduction

This is a semiformal specification draft of the proposed HTTP REST interface of the next version of GNDMS (>= 0.6.X).

Terms and Definitions

Def. **“foo” specifier**
: URL to “foo” resource previously returned by GNDMS, plus a map which maps the URL path variables to real values, and possibly some payload.

Def. **Confirmation**
: Some actions respond with a confirmation, this is usually the HTTP status of the response which is mainly OK (200) or CREATED (201).

In resource descriptions, we do not write complete URLs but instead start after the leading prefix to the GNDMS installation, e.g. instead of **http://tralalala.de/gndms** we just write **/gndms**. To specify parameters encoded into the URL scheme for addressing resources, we consistently use the syntax **_{parameter}.**

Representative Usage Scenarios

Find Grid Service

1. GET /gndms RET List of grids
2. GET /gndms/_{grid id} RET List of services for grid _{grid id}
3. Success
4. Continue querying returned services (DSpace, GORFX, TaskExecution)

Create Slice and Upload File

All URLs are prefixed with /gndms/_{grid id}

1. GET /dspace RET List of subspace specifiers
2. Select subspace to be used
3. GET /dspace/_{subspace id} RET List of facet specifiers
4. GET /dspace/_{subspace id}/slicekinds RET List of slice kinds
5. Select slice kind to be used
6. POST /dspace/_{subspace id}/_{slice kind id} RET Slice specifier
7. PUT /dspace/_{subspace id}/_{slice kind id}/_{slice id}/_{file name}

RET Confirmation

8. Success

Create Slice and Upload File via GridFTP

All URLs are prefixed with /gndms/_{grid id}

1. GET /dspace RET List of subspace specifiers
2. Select subspace to be used
3. GET /dspace/_{subspace id} RET List of facet specifiers
4. GET /dspace/_{subspace id}/slicekinds RET List of slice kinds
5. Select slice kind to be used
6. POST /dspace/_{subspace id}/_{slice kind id} RET Slice specifier
7. GET /dspace/_{subspace id}/_{slice kind id}/_{slice id}/gsiftp
   RET GridFTP-Space URL
8. Upload file {file name} using a GridFTP-Client to <GridFTP-Space URL>/{file name}
9. Optionally verify succesful Upload
   GET /dspace/_{subspace id}/_{slice kind id}/_{slice id}/files
   RET List of files in slice
10. Success

Common Slice Operations

- List files: GET <Slice URL>/files
- Find associated GSI-FTP-Space: GET <Slice URL>/gsiftp
- Delete slice: DELETE <Slice URL>
- Retrieve file: GET <Slice URL>/_{file name}
- Delete file: DELETE <Slice URL>/_{file name}

Execute Task Flow

All URLs are prefixed with /gndms/_{grid id}

1. GET /gorfx RET List of supported facets
2. GET /gorfx/taskflows RET List of supported task flow types
3. Select task flow type to instantiate
4. POST /gorfx/_{task flow type id} RET Task flow specifier
5. Optionally change order and select quote
6. PUT /gorfx/_{task flow type id}/_{task flow id}/task
   RET Task specifier
7. GET <Task URL> RET List of facets
8. Retrieve status and results via returned facets
   a. GET <Task URL>/status RET Status

    b. GET <Task URL>/result RET Result
    c. GET <Task URL>/errors RET List of errors
9. Success

Execute Config Action

All URLs are prefixed with /gndms/_{grid id}

1. GET /gorfx/config RET List of available actions
2. Select action to execute
3. GET /gorfx/config/_{action id} RET Description, possibly Web-Form
4. POST /gorfx/config/_{action id} RET Result
5. Success

Execute Batch Action

All URLs are prefixed with /gndms/_{grid id}

1. GET /gorfx/batch RET List of available actions
2. Select action to execute
3. GET /gorfx/batch/_{action id} RET Description, possibly Web-Form
4. POST /gorfx/batch/_{action id} RET Batch action specifier
5. GET /gorfx/batch/_{action id}/_{batch action id}
   RET Task specifier
6. GET <Task URL> RET List of facets
7. Retrieve status results via returned facets
    a. GET <Task URL>/status RET Status
    b. GET <Task URL>/result RET Result
    c. GET <Task URL>/errors RET List of errors
8. Success

Resources

GNDMS is accessed exclusively via an HTTP REST interface. Below all relevant services are specified in detail. Exact serialization formats are not provided here, as conforming clients are expected to use the Java based API.

HTTP-Header

Per Resource:

- HTTP Request URL used to access Resource, e.g. http://hostname/resource/
- **GNDMS-Facet-URL**: Facet URL, e.g. http://hostname/resource/<facet>
  A facet is a part of a resource (i.e. configuration, access rights, etc.).
  Faceted resources return a list of URLs to all available facets upon http GET to the

main resource.
- **GNDMS-Parent-URL:** (If existing)
  URL to resource containing this resource (Resource that created this resource). Facet URLs use the parent of their associated main resource.
- **GNDMS-Main-Resource-URL**: (if faceted)
- DN / OpenID
  (Depending on security architecture)
- [OPTIONAL] **GNDMS-Workflow-Id**:
  For tracking distributed workflows over multiple log files.

Content Types

- text/application-json
  - for request bodies and answer bodies when communicating with GNDMS
- application/octet-stream
  - for file transfer from and to the GNDMS
- [OPTIONAL] text/html
  - for web-based access to GNDMS
  - [PENDING] for as long as HTTP access using the proposed security architecture is not defined properly

Standard Errors and Return Values

- Resource doesn't exist
- File transfer may use redirects; this is handled by the client

List of Top-Level Resources

All top-level resources primarily provide service discovery of deployed grids and available functionality ("subservices").

/gndms

GET
-> List of grids

/gndms/_{grid id}

GET
-> List of service specifiers

List of DSpace Resources

DSpace is the storage layer of GNDMS. It manages and grants access to flat file containers

("slices") inside larger storage volumes ("subspaces"). Slices have a limited lifetime after which they are deleted. Slices are typed with a role ("slice kind") that may be used for authorization, authentication, and resource assignment decisions. Slices may be transformed to a new slice with a different slice kind as part of task and work flow execution.

/gndms/_{grid id}/dspace

GET
-> List of subspace specifiers

/gndms/_{grid id}/dspace/_{subspace id}

GET
-> List of facet specifiers

PUT (CREATE only)
<- Subspace configuration
-> Result of subspace GET
!! Invalid configuration
!! Overwrite

DELETE
-> Task specifier

/gndms/_{grid id}/dspace/_{subspace id}/config

GET
-> Subspace configuration

PUT
<- New subspace configuration
-> Confirmation
!! Invalid Configuration

/gndms/_{grid id}/dspace/_{subspace id}/slicekinds

GET
-> List of slice kind specifiers

/gndms/_{grid id}/dspace/_{subspace id}/_{slice kind id}/

GET
-> slice kind representation, containing
- Type URL
- Descriptive Commentary

- Statistics

POST
<- Optional slice parameters (Lifetime, Size, Permissions)
<- Optional new type URL
<- Optionally reset statistics
-> Slice specifier

PUT
<- Optional new default slice parameters (Lifetime, Size, Permissions)
-> Confirmation

DELETE
-> Confirmation

/gndms/_{grid id}/dspace/_{subspace id}/_{slice kind id}/_{slice id}

GET
<- Slice parameters
<- List of facets

PUT
<- New slice parameters
-> Confirmation

POST /* Transform slice */
<- Slice kind specifier
-> Slice specifier

DELETE (Deletes whole slice)
<- Confirmation

/gndms/_{grid id}/dspace/_{subspace id}/_{slice kind id}/_{slice id}/files

GET[?attrs=ListOfAttrs]
-> Slice parameters
-> List of files with requested attributes

DELETE (Deletes all files)
<- Confirmation

/gndms/_{grid id}/dspace/_{subspace id}/_{slice kind id}/_{slice id}/gsiftp

GET
-> GridFTP URL to Slice

!! Not Supported

/gndms/_{grid id}/dspace/_{subspace id}/_{slice kind id}/_{slice id}/_{file name}

GET[?attrs=ListOfAttrs]
-> File attributes
-> Contents

PUT
<- Contents
-> Confirmation

DELETE
-> Confirmation

List of GORFX Resources

GORFX denotes a bundle of services for the execution of long running domain specific data management activities (“task flows”) and for configuration of the GNDMS installation.

/gndms/_{grid id}/gorfx

GET
-> List of available facets

List of GORFX Configuration Resources

These resources are used to configure a local GNDMS installation by the system administrator and possibly by grid administrators, too.

/gndms/_{grid id}/gorfx/config
Configration endpoint for the whole GNDMS system.

GET
-> Lists available config actions

/gndms/_{grid id}/gorfx/config/_{action id}

GET
-> Description and Help,
(possibly containing form for submitting input parameters directly)

POST
<- Input Parameters
-> Result (page)

/gndms/_{grid id}/gorfx/batch

Endpoint for one-shot job submission

GET

-> Lists available batch actions

/gndms/_{grid id}/gorfx/batch/_{action id}

Fires a well know action

GET

-> Description and Help,
(possibly containing form for submitting input parameters directly)

POST

<- Input Parameters
-> Batch action specifier
-> Task specifier (Maybe a redirect)

/gndms/_{grid id}/gorfx/batch/_{action id}/_{batch action id}

GET

-> Task specifier (Maybe a redirect)

List of GORFX Task Flow Resources

GORFX task flows allow the negotiation and execution of complex, possibly domain-specific data management activities.

/gndms/_{grid id}/gorfx/taskflows

Endpoint for GORFX task flows (ORQs in the WS-GNDMS)

GET

-> List of supported task flow types

/gndms/_{grid id}/gorfx/_{task flow type id}

Addresses a well-known task flow

GET

-> Statistics
-> Description

POST

<- Task flow type
<- Order
-> Task flow specifier

/gndms/_{grid id}/gorfx/_{task flow type id}/_{task flow id}
Addresses an instance of a task flow

GET
-> List of supported facets

DELETE
-> Confirmation

/gndms/_{grid id}/gorfx/_{task flow type id}/_{task flow id}/order
The order facet of a task flow instance, used to provide task flow parameters

GET
-> Submitted order

PUT
<- New order
-> Confirmation if the new order was accepted

/gndms/_{grid id}/gorfx/_{task flow type id}/_{task flow id}/quote
The quote facet, used to acquire and submit quote (Offer in WS-GNDMS)

GET
-> (Optionally) User-requested quote
-> List of quote specifiers

POST
<- New user-requested quote
-> Confirmation
-> List of quote specifiers

/gndms/_{grid id}/gorfx/_{task flow type id}/_{task flow id}/quote/_{quote id}
Select a specific quote

GET
-> Quote

DELETE
-> Confirmation

/gndms/_{grid id}/gorfx/_{task flow type id}/_{task flow id}/task
Addresses the task facet of a task flow instance

GET
-> Actual task specifier (Maybe as redirect)

PUT[?quote=Quote URL or id]
<- (Optional) Quote specifier
-> New task specifier (Maybe as redirect)
!! Already exists

/gndms/_{grid id}/gorfx/_{task flow type id}/_{task flow id}/status
Requests the status of a task flow instance

GET
-> Actual taskflow status specifier, (might box task status if a task available)

/gndms/_{grid id}/gorfx/_{task flow type id}/_{task flow id}/result
Addresses the result of a task flow if available

GET
-> Actual task result specifier (Maybe as redirect)

/gndms/_{grid id}/gorfx/_{task flow type id}/_{task flow id}/errors

Queries errors of the task flow if available

GET
-> Actual task error specifier (may contain taskflow errors)

List of Task Resources

/gndms/_{grid id}/gorfx/tasks
Endpoint of the GORFX task service, used to control running tasks

GET
-> Statistics
-> List of tasks specifiers

/gndms/_{grid id}/gorfx/tasks/config
Addresses the service configuration

GET
-> TaskExecutionService configuration

POST
<- New TaskExecutionService configuration

-> Confirmation

/gndms/_{grid id}/gorfx/tasks/_{task id}
Addresses a single task instance

GET
-> List of task facets

DELETE
-> Confirmation

/gndms/_{grid id}/gorfx/tasks/_{task id}/status
Queries the status of a concrete task

GET
-> Task status
[-> RSS with
-> Task description
-> Task status]
POST
<- Status change request
-> Confirmation

/gndms/_{grid id}/gorfx/tasks/_{task id}/result
Collects the result of a concrete task, if available
GET
-> Result, if available

/gndms/_{grid id}/gorfx/tasks/_{task id}/errors
Collects the errors of a concrete task, if available

GET
-> List of errors, possibly empty

User-Centric View on GNDMS

Provides per user listings about task flow and data (“DSpace”) entities owned by a specific user, possibly enhanced with authorization information.

/gndms/_{grid_id}/home/_{user id}
Entry point for the per user view

GET

-> List of facets

/gndms/_{grid_id}/home/_{user id}/slices
Queries all slices of a specific user

GET
-> List of slice specifiers

/gndms/_{grid_id}/home/_{user id}/taskflows
Queries all task flows of a specific user

GET
-> List of task flow specifiers

/gndms/_{grid_id}/home/_{user id}/tasks
Queries all task of a specific user

GET
-> List of task specifiers

/gndms/_{grid_id}/home/_{user id}/config
Queries allowed config actions

GET
-> List of allowed config actions for this user

/gndms/_{grid_id}/home/_{user id}/batch
Queries allowed batch actions

GET
-> List of allowed batch actions for this user

/gndms/_{grid_id}/home/_{user id}/summary
Delivers a "summary" for a user

GET
-> Aggregated information about all resources of {user id} managed by this installation of GNDMS

Authorization and Role Management

This section is a non-normative DRAFT and work-in-progress of non-binding character.

Underlying Security Model Assumptions

Users are globally uniquely identified (e.g. with a DN or Open ID). This user identifier is part of each HTTP request header and denotes the user on which behalf the request is executed. The user identifier is not necessarily identical with the credential used to authenticate the request's connection. This may happen for example when trusted services need to communicate using server or service credentials. However it is always guaranteed that the included user identifier is valid and was authenticated at some prior point in a preceding chain of trusted intermediaries.

Each user is assigned multiple roles. Roles are identified by unique role identifiers (URIs). The technical details of how a information about roles is transported is left unspecified here. Possible solutions include additional headers, lookup services, or embedded, signed SAML assertions.

Connecting Resource Access Rights with User Roles

Each restricted resource defines a set of URI-identified permissions and permission management resources required for assigning it to roles. The list of permission specifiers that are in use by a concrete resource is obtained via an HTTP GET to a special permission information resource. A link to this permission information resource is returned as part of the answer to an HTTP GET to the restricted resource. This link may be part of the HTTP answer header. This permission information resource may very well be a facet of it's restricted resource. We reserve the facet name **/perms** for this purpose.

Behaviour of Permission Management Resources

GET

-> Role expression requirement for using this permission

PUT

<- Role expression requirement for using this permission

[/_[{user}]] GET

-> Confirmation if user has this permission (Current user if not specified)

!! Forbidden

Alternatively permission management resources may be named capabilities. We reserve **/gndms/caps** and **/gndms/_{grid specifier}/caps** and **/gndms/_{grid specifier}/{service} /caps** as possible REST URI space locations for such resources. These top-level resources may return a list of subcapabilities on HTTP GET.

Comments on Various Technical Problems

Redirecting HTTP Posts

It may be necessary to provide an HTTP-based file upload from the portal to a GNDMS site. Below is a description of how it should work but doesn't due to technical limitations:

- An authenticated user performs a file upload (HTTP POST) to an upload URL at the portal. This upload URL is associated with an actual target slice address and optional request information.
- The portal informs the GNDMS target site **out of band** about the future incoming file upload, and the associated target slice address, optional request information, and a special upload cookie used to identify this file upload. The GNDMS site will reply with a target URL and store this data if the request was valid. If the request was invalid, the target site will reply with an error.
- In case of having received an error, this error is returned directly to the user as a reply to the HTTP post by the portal. The protocol ends here.
- Otherwise, in case of having received a target URL, the file upload is redirected to this target URL and the user's browser is told to set the upload cookie as part of the reply that contains the redirect.
- Finally, the GNDMS target URL verifies the HTTP request using the agreed upload cookie and processes the uploaded data.

Sadly, this only works in theory. In practice HTTP does not support redirection of POST requests. A possible alternative based on HTTP GET is outlined below:

- An authenticated user wishes to perform a file upload. It performs an HTTP GET to a file upload preparation URL at the portal. This preparation URL is associated with an actual target slice address and optional request information.
- The portal informs the GNDMS target site **out of band** about the future incoming file upload, and the associated target slice address, optional request information and a special upload cookie used to identify this file upload. The GNDMS site will reply with a target URL and store this data if the request was valid, If the request was invalid, the target site will reply with an error.
- In case of having received an error, this error is returned directly to the user as a reply to the HTTP GET by the portal. The protocol ends here.
- Otherwise, in having received a target URL, the portal replies with a redirect to the target URL and the user's browser is told to set the upload cookie as part of the reply that contains the redirect.
- The GNDMS target URL is accessed by the user with an HTTP GET. The GNDMS target site verifies the request using the agreed upload cookie and replies with a HTTP body document (XML, HTML) containing a drop-off URL at the target site.
- The user's client sends an HTTP POST to this drop-off URL containing the file data to be uploaded. The GNDMS target site again verifies the request using the upload

cookie and proceeds with the upload. The protocol ends here.

Alternatively, HTTP HEAD requests may be used to achieve a similar effect.

Or even better, just use 307, it’s meant to exactly solve this problem.

The exact specification of REST APIs for the creation of target URLs, drop-off URLs, and target URL body formats is left open until needed.