1. Introducing DSSWeb

As a data curator of one or more data projects, you can manage your DSS storage via our DSSWeb self-service system. The system provides you with a Web Interface (GUI), a Command Line Interface (CLI) and a HTTP REST API (API).

You can think of DSSWeb as your virtual storage system administrator, which accepts your request, such as:

- Create a 10TB DSS container for project XYZ and back it up every day
- Give user Max access to DSS container XYZ
- Export DSS Container XYZ to my virtual machine with IP X.X.X.X for 10 days

and then automatically executes the necessary steps in order to fulfill your request.

1.1. Introducing the DSSWeb Management Portal

Currently the Web Interface is still in development and not available for production use. We plan general availability of the UI for Q2/Q3 2018.

1.2. Introducing the DSSWeb Management CLI

The DSSWeb Command Line Interface is currently available as Software as a Service (SaaS) in the LRZ cloud (which we recommend to use) and as an on-premise software solution. In the following, we will describe how to get started with both versions of the CLI.

1.2.1. Installing the CLI on premise

In order to install the CLI on premise, the installation of git and python3.4 (or higher) is required. Use the following steps to install the CLI on a *NIX system:

```bash
  git clone https://gitlab.lrz.de/a2822bp/dssweb-client.git
  cd dssclient
  python3 setup.py install
```

After successful installation you should be able to start the CLI in interactive mode using the `dsscli` command.
$ dsscli
(dsscli) help

Shell commands (type help <topic>):
=====================================
cmdenvironment  help     load  pyscript  run   set    shortcutsedit              history  py     quit     save  shell  show

Application commands (type help <topic>):
==========================================
complete               dss container update   login    project show
dss container create   dss nfsexport create  pool list
dss container list     dss nfsexport list     pool show
dss container show     dss nfsexport show id  pool show id
dss container show id   help                   project list
...

You can also use the CLI in non-interactive mode, like a regular shell command by calling the dsscli command directly with the particular arguments for the action you want to perform.

$ dsscli -h

DSSWeb Client

optional arguments:
--version            show program's version number and exit
-v, --verbose        Increase verbosity of output. Can be repeated.
-q, --quiet          Suppress output except warnings and errors.
--log-file LOG_FILE  Specify a file to log output. Disabled by
default.
-h, --help           Show help message and exit.
--debug              Show tracebacks on errors.

Commands:
...
help                  print detailed help for another command (cliff)
login                 Request username and password and obtain a token from
DSSWeb API.
...

1.2.2. Updating the CLI

As new features become available, we will update the code in the gitlab repository, so make sure to update the software from time to time. In order to update the software, just follow the installation procedure again. It will overwrite the old version of the software.

1.2.3. Accessing the CLI in the cloud

In order to access the interactive CLI, which is offered as Software as a Service in the LRZ cloud, just use SSH to login to dsscli.dss.lrz.de. In order to be able to log in, your user will need to have data curator rights for one or more data projects.
$ ssh XXXXXX@dsscli.dss.lrz.de

For your convenience, you can provide one or more SSH public keys in order to be able to login in without having to provide a password. Just use the *edit* command, which will allow you to edit the *authorized_keys* file of your user.

Another advantage of the SaaS version is that we will provide you with updated versions of the CLI automatically.

1.2.4. General usage of the CLI

The CLI is structured in several subcommands, that allow you to list, show, create and change various items of a data project like containers, access invitations, NFS exports, etc. To get a list of available subcommands, just call the *help* command in interactive mode or use the *-h* switch in non-interactive mode.

In order to get help on a particular command, you can just call the subcommand, followed by the *-h* switch.
(rdsscli) dss container list -h                <---- You can also use
dsscli dss container list -h for non-interactive version
usage: dss container list [-h] [-f {csv, json, table, value, yaml}] [-c
COLUMN]
        [--max-width <integer>] [--fit-width]
        [--print-empty] [--noindent]
        [--quote {all, minimal, none, nonnumeric}]
        [--poolname [POOLNAME]]
        [projectname]

List all available DSS data containers.

positional arguments:
  projectname

optional arguments:
  -h, --help            show this help message and exit
  --poolname [POOLNAME]

output formatters:
  output formatter options
  -f {csv, json, table, value, yaml}, --format {csv, json, table, value, yaml}
      the output format, defaults to table
  -c COLUMN, --column COLUMN
      specify the column(s) to include, can be
      repeated

table formatter:
  --max-width <integer>
      Maximum display width, <1 to disable. You can
      also use
      the CLIFF_MAX_TERM_WIDTH environment variable,
      but the
      parameter takes precedence.
  --fit-width
      Fit the table to the display width. Implied if
      --max-
      width greater than 0. Set the environment
      variable
  --print-empty
      Print empty table if there is no data to show.

json formatter:
  --noindent
      whether to disable indenting the JSON

CSV Formatter:
  --quote {all, minimal, none, nonnumeric}
      when to include quotes, defaults to nonnumeric

Note that the output of the commands is usually optimised for human readability. However, if you want to script certain things, you might be
interested using the --f switch to change the output formatting to something, which is easier to parse by a script.

1.3. Introducing the DSSWeb Management API

Currently the API still lacks some important documentation information. Though, we are working hard on extending this documentation in the
future. You can access the current documentation via https://dssweb.dss.lrz.de/docs/.
2. Understanding Data Projects and Data Pools

In the LRZ Data Science Management Concept, we use Data Projects as organizational envelopes around the data, which is stored in the context of this data project. This is particularly useful as it implements a link between the data and the people who are formally responsible for the data and therefore protects us from orphaned data.

In order to keep this link alive, as master user of such a data project, LRZ will request you to confirm prolongation of the data project, every year. If you fail to do so, we will shut down the project and delete the data that is stored in the context of it.

A particular data project is managed by a group of users, which is called the data curator group of the project. By default only the master users of the project belong to this group, but upon request via the Servicedesk, LRZ can also add arbitrary other users to the list of data curators for a data project.

In order to be able to store data in the context of a data project, the project also needs grants on one or more data pools. Data pools are physical storage systems which are integrated into the LRZ Data Science Management Ecosystem. Currently only Data Science Storage systems are supported, but it is planned to add other systems like an Archive for example in the future. Grants specify which fraction of a particular data pool (a quota), a particular data project is allowed to use. These grants are managed by LRZ.

Note that data pools are usually optimized for a particular data pattern or workload. This information is provided you via the description and hints fields of the data pool to give you some guidance on container placement.

2.1. Getting information about the data projects of a data curator

2.1.1. Using the GUI

2.1.2. Using the CLI

In order to list the available data projects, for which you are assigned the data curator role, use the project list command.
In order to get the details about a particular data project, use the `project show` command.

```
(dsscli) project show pr74qo
```

---

2.2. Getting information about the grants for a particular data project

2.2.1. Using the GUI

Click here to see how it works in the GUI

2.2.2. Using the CLI

In order to list the available grants for a particular data project, use the `pool list` command.

```
(dsscli) pool list pr74cu
```

---

In order to get the details about a particular data project, use the `pool show` command.
3. Understanding Containers

Containers are the organizational elements, in which your actual data will live. Depending on the type of the underlying storage system, containers could be implemented in very different ways. For the Data Science Storage (DSS) systems, a container is basically a directory in a POSIX file system, with some additional magic attached to it. For a future archive system or an object storage system, a container will most likely be implemented in a different way. However, what is important to note is the basic concept of a container, which will be equal across all potential underlying storage systems:

1. A container is some kind of storage element with a certain size, which lets you store data in it.
2. A container should be used to encapsulate data, which semantically belongs together.
3. You can invite arbitrary users, which exist in the LRZ Identity Management System to access the data in the container.

3.1. Understanding Data Science Storage Containers

As already discussed, data containers on a Data Science Storage system are implemented as a directory on a POSIX file system, which is owned by a POSIX group, dedicated to the container. Each container has a certain limitation on the capacity and number of files, which can be stored in the container. The members of the container group are managed by the data curators via an invitation process. The access to the container is managed via POSIX rights and NFSv4 type ACLs. Also each container is associated with a so called enforcement policy, which defines how hard the system will try to make sure that access rights and ACLs stay in alignment with the access rights concept. Last but not least, each DSS type container is associated with a so called protection policy, which defines if and how the data in a container is protected by regular backups to the LRZ Backup- and Archive system.
3.1.1. Understanding Data Science Storage Container Enforcement Policies

In contrast to other file system types, such as NTFS, you cannot revoke certain POSIX mode bit and ACL change rights of the owner of a file. Therefore you cannot guarantee, that a user may accidentally or on purpose do something unexpected, that may break the right enforcement methods in place. However, there are certain measurements that can be taken, which increasingly deviate from the POSIX standard and also are increasing expensive in terms of performance. Therefore, we offer several levels of enforcement policy, so you can pick the right one for your use-case.

- The most basic enforcement policy is called **NONE**. Using this policy, the container directory will be created with owning user root and owning group <container group> and sets the POSIX rights to **rw-</**. This will make sure that directories and files created within the container will only be accessible by the user who created the directory and files created within the container. However, every user can change the permissions of their files and directories using chmode and therefore overwrite the default settings easily. So the use case for this policy is usually that you want to let your users manage the access rights within the container by themselves and we only make sure that access at the container level is restricted to the invited container users.
- A more restrictive but still almost POSIX compliant enforcement policy is called **NORMAL**. This is also the default mode, you should choose if you are in doubt. This implements the same POSIX mode bits as the **NONE** mode, but in addition sets a set of NFSv4 style ACLs on the container directory, that will automatically be inherited by all files and directories, created in the container, which makes sure that all users belonging to the container group have **rwx** rights on files and **rwx** rights on directories. The point where this setting deviates from the POSIX or NFSv4 semantic is that if a user changes the POSIX mode bits using chmod, we do not replace the ACL by the new mode bits, but instead only update the ACL and therefore keep our access right settings for the container group intact. So basically the only way for a user to overwrite the permissions on a file he owns, is by explicitly editing the NFSv4 style ACL.
- A even more restrictive and barely POSIX compliant enforcement policy is called **STRICT**. This implements the same techniques described for the **NORMAL** mode, but disallows the use of chmod completely. As this happens on the file system level, this also holds true for each program library call, that uses a chown POSIX API call. So users should not be able to remove the setgid bit. However, by editing the ACLs, they can still overwrite the permissions.

3.1.2. Understanding Container Access Rights Management

For managing container access rights, we follow an **invitation approach** as you may already be familiar from cloud storage services like the LRZ Sync+Share service or Dropbox, Google Drive, etc. This means, for example, if data curator Alice wants to give user Bob access to the data container Cont-A, Alice will tell DSSWeb to invite Bobs username(s) to access container Cont-A. DSSWeb will then send an E-Mail, containing a unique invitation link to the E-Mail address assigned to Bobs username(s) in the LRZ Identity Management System. Bob can then accept the invitation by clicking on the invitation link and accepting our data privacy and usage terms. After that, Bobs username will be automatically placed in the container POSIX group of Cont-A and Bob is ready to access the data in Cont-A. However, if Bob chooses not to accept the invitation for whatever reason, the pending invitation will automatically expire after 30 days in DSSWeb.

3.1.3. Understanding Container Access Revocation and User Deletion Actions

As you may have noticed, the semantics of Data Containers are designed as collaborative space, in which every user can access all data. So semantically the data is not owned by an individual, but the whole group. However, as we operate on a POSIX file system, the POSIX semantics dictate that for each file there has to be a single user who is the formal owner of the file. Technically this is implemented by storing the UID of the owning user for each file in the metadata of the file.

Now let's look at the case when a data curator wants to revoke access of a certain user to a container. Therefore, we remove the user from the container group and because of the permissions set on the container level, the user cannot access any data in the container anymore. However, there may now still be files in the container which are formally owned by the revoked user. While this is technically insignificant - since the user cannot access the data in the container anymore - it may be misleading and ambiguous to the remaining users. The same holds true if a user, invited to a container gets completely deleted. In this case, even resolving the UID to a user name will not work anymore and you will be presented with a cryptic number as owner of the files in your container.

In order to avoid these kinds of problems, whenever you revoke access for a user or a user get's deleted in the Identity Management System, we automatically start a batch job, that changes ownership of all files, which belong to the revoked or deleted user. Please note that this will not happen if the container enforcement policy is set to **NONE**. Currently the new owner is automatically determined from the list of data curators, whereby we choose that user, which is the first one in an alphabetical order. Note that this happens asynchronously so there may be a noticeable delay between revocation/deletion and ownership change.

We plan to extend this functionality in future releases, so that you may be able to specify a new owner, when you revoke a particular user, or put all data owned by that user to a tape archive or even delete all data owned by that user. If you have a particular use case, which you wish to see implemented, please open a ticket at the LRZ Servicedesk so that we can discuss it and file a user story in our product backlog.

3.1.4. Understanding Data Science Storage Container Protection Policies

In contrast to highly available and therefore more cost intensive LRZ storage offerings like the LRZ NAS Cloud or LRZ Sync+Share, where data is replicated between two independent disk storage systems, so that in case a whole storage system fails we can switch the service to the second system within hours, Data Science Storage is cost and performance optimized. Therefore
The most simple data protection policy is called **NONE**. Containers which are assigned this protection policy are not protected at all. Use this policy only for data, which can easily be regenerated or recaptured.

For automatic backups, which are performed once a week, use the policy **BACKUP_WEEKLY**. This will perform an incremental backup run of your container once a week (usually during the weekend). The retention policies of the backup are 180 days and a maximum of 3 versions per file. For more information about DSS backup retention policies, see **DSS Understanding Data Science Storage Container Backup Retention**.

For automatic backups, which are performed once a day, use the policy **BACKUP_DAILY**. This will perform an incremental backup run of your container every day (usually during night). The retention policies of the backup are 180 days and a maximum of 3 versions per file. For more information about DSS backup retention policies, see **DSS Understanding Data Science Storage Container Backup Retention**.

For containers which exclusively store static, read-only content, like simulation results, output data from instruments like genomic sequencers, microscopes, NMR spectroscopes, etc., we also provide special **ARCHIVE** policies. These policies help you to be compliant with the DFG rules of good scientific practice, which require you to retain scientific primary data for 10 years. This is implemented by assigning the backed up data of your container a special retention policy, that keeps backups for 10 years and also keeps two copies of the backed up data in two different locations. Please note the restrictions that apply when using this data protection policy: **DSS Understanding Data Science Storage Container Backup Retention**.

- For automatic incremental archives of static data, which are performed once a week, use the policy **ARCHIVE_WEEKLY**. This will perform an incremental archive run of your container once a week (usually during weekend).
- For automatic incremental archives of static data, which are performed once a day, use the policy **ARCHIVE_DAILY**. This will perform an incremental archive run of your container once a day (usually during night).

Data Science Storage Container data protection mechanisms have several known limitations. Make sure that you fully understand them, before implementing a solution that relies on them.

For information on how to restore backed up files, please check this Knowledge Base article.

**4. Understanding the Asynchronous Task Model**

As we already described earlier, DSSWeb takes your high level requests (like create a new container) and automatically executes the necessary steps to fulfill your request. Every time you tell DSSWeb to create or change something, it must talk to one or more external systems, like LRZ SIM, LRZ’s Backup and Archive System, the underlying filesystems of DSS and so on and ask them to execute one or more tasks. So under the covers, DSSWeb is a fairly complex, highly distributed system which depends on multiple systems working seamlessly together. However, in such highly distributed systems, we have to accept that things may fail from time to time. Fortunately, these failures often have a temporary and very seldom a permanent nature. For example, think of the situation in which one of the subsystems is not available for whatever reason. This situation is usually fixed very fast and then everything works just fine again.

In order to hide all this complexity and not to bother you with temporary errors, thereby giving you the best possible user experience, we work with an asynchronous task model. That means, whenever you tell DSSWeb to create or change something, it will just store the necessary information in its database and make sure that an asynchronous task is started in the background. So while DSSWeb may still be busy, fulfilling your request, you can already go on and do other stuff and then check back later to see if your request was carried out successfully.

You can check the status of an object (like a container), by looking at its **Status** and **Active Task** fields. Typically when you first create an object it goes to a **CREATE_PENDING** state, meaning the object is currently being setup. When an object reaches a "steady state", meaning it was successfully created, its status is usually **ACTIVE**. When you change an object, it will go to a **CHANGE_PENDING** state and as soon as the changes have been carried out it will go back to **ACTIVE** again. The same semantics often are also true for deletion of objects, meaning the object will get to a **DELETE_PENDING** state and only after it has been successfully cleaned up on the subsystems, it is removed from the DSSWeb database.

Please note that currently, when an object is in transition, we do not accept any further changes to the object. So whenever an object is in a **PENDING** state, you have to wait until it is **ACTIVE** again to apply the next operation to this object.

**4.1. Understanding Task Failures**

Usually, when an asynchronous task hits a failure in a subsystem, the only thing you should notice is that the task may take longer than normal to execute (there is no special state indicating a problem for an object, it will simply stay in the **PENDING** state). The implemented semantics are that whenever a task encounters a failure, it will retry after some minutes until the task can finally be finished. DSSWeb monitors the execution times and number of retries of tasks and in case the retry count of a task gets unusually high, it will notify LRZ...
5. Understanding our Software Development Process

Our software development process is loosely based on an agile approach and is highly driven by the demands of our customers. So, if you are missing a particular feature or would like to see something particular to be implemented in DSSWeb, please don’t hesitate and reach out to us by opening a request on the LRZ Servicedesk so we can discuss your requirements and record it in our product backlog.

6. Managing Containers

In the following we describe how you can create a new DSS Container for your data project.

6.1. Creating Containers

Please note that in order to create a new DSS container, the following requirements must be met:

- You must be assigned the data curator role for the data project
- Your data project must own a grant for the specific data pool in which you want to create the container
- The size and file limits of the container must be less or equal to the available size and files in your pools grant

Please note that currently you cannot change the data protection mode or enforcement policy of a container. Since these changes may require substantially resources to perform a transition or may have unwanted side effects on your data, currently these properties can only be set at container creation time. So choose carefully at container creation time.

If you really require such a transition for an existing container, please contact us via the LRZ Servicedesk.

Click here to see how it works in the GUI

6.1.1. Using the GUI

Click here to see how it works in the CLI

6.1.2. Using the CLI

In order to create a new container for a particular data project in a particular pool, use the `dss container create` command.

This command takes the following mandatory arguments:

- The name of the data project
- The name of the data pool
- The maximum allowed amount of data, which can be stored in the container in Gigabytes
- The maximum allowed number of files, which can be created in the container

Additionally, you can specify the following optional arguments:

- A description for the container.
- The data protection mode as described in the data protection section. (Default is NONE)
- The enforcement policy as describe in the enforcement policy section. (Default is NORMAL)

In the following example, we are going to create a new container

- for the data project `pr74cu`
- on the pool `DSS Testsystem`
- with a size of `1000GB`
- and a maximal number of `10,000 files`
- which will be backed up daily
- and has assigned a strict enforcement policy
6.2. Showing Containers

Please note that all the data curators of a data project will automatically get access to the containers, created for their project. If that is not what you want, you have to revoke access for them manually.

6.2.1. Using the GUI

Click here to see how it works in the GUI

6.2.2. Using the CLI

In order to get an overview about all containers you are allowed to manage via DSSWeb, you can use the `dss container list` command. If you are data curator for multiple projects, you can limit the containers to the ones of a particular project. Additionally you can also limit the listing to the containers of a particular pool.

```bash
(dsscli) dss container list
+----+---------+----------------+-----------------+----------------+
| ID | Project | Pool           | Name            | Status         |
+----+---------+----------------+-----------------+----------------+
| 46 | pr74cu  | DSS Testsystem | pr74cu-dss-0008 | ACTIVE         |
| 48 | pr74cu  | DSS Testsystem | pr74cu-dss-0009 | ACTIVE         |
| 49 | pr74cu  | DSS Testsystem | pr74cu-dss-0010 | ACTIVE         |
| 50 | pr74cu  | DSS Testsystem | pr74cu-dss-0011 | ACTIVE         |
| 51 | pr74cu  | DSS Testsystem | pr74cu-dss-0012 | ACTIVE         |
| 55 | pr98xx  | DSS Testsystem | pr74cu-dss-0013 | ACTIVE         |
| 56 | pr98xx  | SFA12KX Large  | pr74cu-dss-0014 | CHANGE PENDING |
| 57 | pr74cu  | DSS Testsystem | pr74cu-dss-0015 | CREATE PENDING |
| 58 | pr74cu  | DSS Testsystem | pr74cu-dss-0016 | CREATE PENDING |
+----+---------+----------------+-----------------+----------------+
In order to view the details of a specific container, you can use the `dss container show` or `dss container show id` command. The first one takes the container name, the later one the container id of the container to show as mandatory argument:

```
(dsscli) dss container list --poolname 'DSS Testsystem' pr74cu
+----+---------+----------------+-----------------+----------------+
| ID | Project | Pool           | Name            | Status         |
+----+---------+----------------+-----------------+----------------+
| 46 | pr74cu  | DSS Testsystem | pr74cu-dss-0008 | ACTIVE         |
| 48 | pr74cu  | DSS Testsystem | pr74cu-dss-0009 | ACTIVE         |
| 49 | pr74cu  | DSS Testsystem | pr74cu-dss-0010 | ACTIVE         |
| 50 | pr74cu  | DSS Testsystem | pr74cu-dss-0011 | ACTIVE         |
| 51 | pr74cu  | DSS Testsystem | pr74cu-dss-0012 | ACTIVE         |
| 57 | pr74cu  | DSS Testsystem | pr74cu-dss-0015 | CREATE PENDING |
| 58 | pr74cu  | DSS Testsystem | pr74cu-dss-0016 | CREATE PENDING |
+----+---------+----------------+-----------------+----------------+
```

```
In order to view the details of a specific container, you can use the `dss container show pr74cu-dss-0002` command:

```
(dsscli) dss container show pr74cu-dss-0002
+----------------------+----------------------------------+
| Field                | Value                            |
+----------------------+----------------------------------+
| ID                   | 31                               |
| Project              | pr74cu                           |
| Pool                 | DSS Testsystem                   |
| Name                 | pr74cu-dss-0002                  |
| Path                 | /dss/dsstestfs01/pr74cu-dss-0002 |
| Description          | Describe me.                     |
| Status               | ACTIVE                           |
| Active Task          | None                             |
| Read/Write Group     | pr74cu-dss-0002                  |
| Read Only Group      | pr74cu-dss-0002-ro               |
| Quota GB             | 99                               |
| Quota Files          | 10000                            |
| ID Streamline Mode   | NONE                             |
| Data Protection Mode | NONE                             |
+----------------------+----------------------------------+
```
6.3. Modifying Containers

In the following we show how certain properties of containers can be changed. Currently you can change the following properties of an existing container:

- The maximal size of the container (Quota GB)
- The maximal number of files of the container (Quota Files)
- The description

Click here to see how it works in the GUI

6.3.1. Using the GUI

Click here to see how it works in the CLI

6.3.2. Using the CLI

In order to modify an existing container, use the `dss container update` command.

This command takes the following mandatory arguments:

- The name of the data container

Additionally, you have to specify at least one of the following arguments:

- The new maximum usable size.
- The new maximum number of files.
- The new description

In the following example, we are going to update the container `pr74cu-dss-0000`:

- New maximum size: 1,000  GB
- New maximum files: 1,000,000
- New description: Updated description
(dsscli) dss container update pr74qo-dss-0002 --size 2048 --files 200000 --description 'Testcontainer updated'

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>99</td>
</tr>
<tr>
<td>Name</td>
<td>pr74qo-dss-0002</td>
</tr>
<tr>
<td>Description</td>
<td>Testcontainer updated</td>
</tr>
<tr>
<td>Status</td>
<td>CHANGE PENDING</td>
</tr>
<tr>
<td>Active Task</td>
<td>1bd90469-adb7-491f-8107-5c6d8f9ee034</td>
</tr>
<tr>
<td>Read/Write Group</td>
<td>pr74qo-dss-0002</td>
</tr>
<tr>
<td>Read Only Group</td>
<td>pr74qo-dss-0002-ro</td>
</tr>
<tr>
<td>Quota GB</td>
<td>2048</td>
</tr>
<tr>
<td>Quota Files</td>
<td>200000</td>
</tr>
<tr>
<td>ID Streamline Mode</td>
<td>STRICT</td>
</tr>
<tr>
<td>Data Protection Mode</td>
<td>ARCHIVE WEEKLY</td>
</tr>
</tbody>
</table>

6.4. Delete Containers

Please note that we have not yet implemented a function for automatically deleting a data container. This feature will be released in a future version of DSSWeb. In the mean time, if you require the deletion of a data container, please contact the LRZ Service Desk.

7. Managing Container Access Rights

As discussed, Data Science Containers are collaborative spaces. As such, you can grant virtually any user, which is known in the LRZ Identity Management, access to a data container. As the Identity Management Systems of LRZ, TUM, LMU are kind of federated, this also means that you can grant any user, managed by one of these IdMs access to your data containers.

7.1. Granting Access Rights

In order to allow an user to access a particular data container, you must invite him or her to the container group.

- Click here to see how it works in the GUI

7.1.1. Using the GUI

- Click here to see how it works in the CLI

7.1.2. Using the CLI

In order to invite a user to access a particular data container, use the `dss invitation create` command.

This command takes the following mandatory arguments:

- The name of the data container
- The username to invite

Additionally, you can specify the following optional arguments:

- The access mode to grant. This can either be `READ_WRITE` (the default) or `READ_ONLY`. However, the later can only be used with containers that implement an `enforcement policy` of `STRICT`.
- The maximum amount of GB, the user is allowed to consume within the container. (Default: unlimited)
- The maximum number of files, the user is allowed to create within the container. (Default: unlimited)

Please note that the various ways in which users can access the data within a container is documented in the DSS documentation for users document.
In the following example, we are going to invite user `a28ditix` to container `pr74cu-dss-0000`, using:

- Maximum granted size: 1,000 GB
- Maximum granted files: 1,000,000
- Access mode: READ_WRITE

```
(dsscli) dss invitation create --containername pr74qo-dss-0001 --username di57gix --accessmode READ_WRITE --maxgb 1000 --maxfiles 1000000
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>83</td>
</tr>
<tr>
<td>Container</td>
<td>pr74qo-dss-0001</td>
</tr>
<tr>
<td>User</td>
<td>di57gix</td>
</tr>
<tr>
<td>Quota GB</td>
<td>1000</td>
</tr>
<tr>
<td>Quota Files</td>
<td>1000000</td>
</tr>
<tr>
<td>Access Mode</td>
<td>RW</td>
</tr>
<tr>
<td>Status</td>
<td>CREATE PENDING</td>
</tr>
<tr>
<td>Active Task</td>
<td>46d9dc61-7135-4523-baa4-d950ecce2ba3</td>
</tr>
<tr>
<td>Expires At</td>
<td>2018-01-04</td>
</tr>
<tr>
<td>Inviter</td>
<td>a2828yy</td>
</tr>
</tbody>
</table>

7.2. Showing Access Rights

- Click here to see how it works in the GUI

7.2.1. Using the GUI

- Click here to see how it works in the CLI

7.2.2. Using the CLI

In order to get an overview about all invitations to containers of your project via DSSWeb, you can use the `dss invitation list` command. Additionally you can limit the output to the invitations of a particular container and/or username.

In the following examples, we will examine invitations for the example project `pr74cu`. Note that size and file quotas are only accepted for invitations with `READ_WRITE` access. If you specify quotas for an invitation with `READ_ONLY` access, these will be ignored.
In order to view the details of a specific invitation, you can use the `dss invitation show` or `dss invitation show id` command. The first one takes the container and username, the later one the invitation id of the invitation to show as mandatory argument:
7.3. Modify Access Rights

In the following we show how certain properties of invitations can be changed. Currently you can change the following properties of an existing invitation:

- For invitations that have not yet been accepted:
  - Prolong expiration date
  - The maximum amount of GB, the user is allowed to consume within the container.
  - The maximum number of files, the user is allowed to create within the container.
- For invitations that have already been accepted:
  - The access mode to grant. This can either be `READ_WRITE` or `READ_ONLY`. However, the later can only be used with containers that implement an `enforcement policy` of `STRICT`.
  - The maximum amount of GB, the user is allowed to consume within the container.
  - The maximum number of files, the user is allowed to create within the container.
7.3.1. Using the GUI

7.3.2. Using the CLI

In order to modify an existing invitation, use the `dss invitation update` command.

This command takes the following mandatory arguments:

- The ID of the invitation to update

Additionally, you have to specify at least one of the following arguments:

- The new access mode.
- The new maximum usable size.
- The new maximum number of files.
- The prolong flag

In the following example, we are going to prolong the invitation 83 to be valid for another 30 days:

```plaintext
(dsscli) dss invitation update 83 --prolong
+-------------+---------------------+
| Field       | Value               |
+-------------+---------------------+
| ID          | 83                  |
| Container   | pr74qo-dss-0001     |
| User        | di57gix             |
| Quota GB    | None                |
| Quota Files | None                |
| Access Mode | RO                  |
| Status      | USER ACCEPT PENDING |
| Active Task | None                |
| Expires At  | 2018-01-04          |
| Inviter     | a2828yy             |
+-------------+---------------------+
```

In the following example, we are going to update the invitation 83:

- New maximum size: 1,000 GB
- New maximum files: 1,000,000
- New access mode: READ_WRITE

```plaintext
(dsscli) dss invitation update 83 --accessmode READ_WRITE --maxgb 1000 --maxfiles 10000000
+-------------+---------------------+
| Field       | Value               |
+-------------+---------------------+
| ID          | 83                  |
| Container   | pr74qo-dss-0001     |
| User        | di57gix             |
| Quota GB    | 1000                |
| Quota Files | 10000000            |
| Access Mode | RW                  |
| Status      | CHANGE PENDING      |
| Active Task | 84a85e4a-933f-464c-aa70-273efe32ccc6 |
| Expires At  | 2017-12-07          |
| Inviter     | a2828yy             |
+-------------+---------------------+
```
7.4. Revoking Access Rights

In the following, we show how existing invitations can be revoked/deleted. You can revoke accepted as well as pending invitations. If you revoke a pending invitation, the invitation link, sent to the person you invited will be invalidated, so accepting the invitation is no longer possible. If you revoke an already accepted invitation, the user will be removed from the container access group and file ownership of each file, technically owned by the user, will be transferred to a data curator. See Understanding Container Access Revocation and User Deletion Actions for more information on the topic.

---

7.4.1. Using the GUI

---

7.4.2. Using the CLI

In order to revoke access to a container for a particular user, use the `dss invocation delete` command.

Whenever a user, invited to a container will be deleted in the Identity Management System, we will automatically remove the user from the container access group and transfer file ownership of the files, technically owned by that user, to a data curator. (See Understanding Container Access Revocation and User Deletion Actions). As we add other possible actions in the future, we will make this configurable per container.

In the following we are going to delete the invitation for user `di57gix` on container `pr74qo-dss-0001`

```
(dsscli) dss invocation delete pr74qo-dss-0001 di57gix
Successfully initiated deletion of invitation 83!
```

After this command was issued, the invitation will transition into the DELETE PENDING state and will disappear after the file ownership transfer job discussed above, was successfully launched.

---

8. Managing DSS Container Exports

Dynamically managing container exports is unfortunately not yet supported because of a technical limitation in the underlying NFS server used. This problem will be addressed and fixed in a future release of DSS. If you need your container exported via NFS to LRZ Cloud Machines or Virtual Machines for example, please open a ticket at the LRZ Service Desk for it. We will setup the exports as soon as possible for you.

---

8.1. Limitations

Please note that there are several limitations for exporting DSS containers via NFS.

You can only export DSS containers to IPs that are located in the LRZ data center. Exporting to IPs external to LRZ is not supported.

All DSS container exports are configured with `root_squash` enabled. Therefore it is not possible to have "root access" on the data via an NFS client.

Also technically not forbidden, you should only export DSS containers to IPs that are statically assigned to and trusted by you. NFS exports follow a "host based trust" semantic, which means the DSS NFS server will trust any IP/system to which a DSS container is exported. There is no additional user authentication between NFS server and client enforced. This is especially important if you want to export DSS containers to cloud machines, as these - by default - use a dynamically allocated IP, which may be reused by other machines as soon as you shut down your VM.

---

8.2. Creating exports
8.3. Showing exports

8.3.1. Using the GUI

Click here to see how it works in the GUI

8.3.2. Using the CLI

Click here to see how it works in the CLI

In order to get an overview about all NFS exports of containers of your project via DSSWeb, you can use the `dss nfsexport list` command. Additionally you can limit the output to the exports of a particular container and/or pool name.

In the following examples, we will examine NFS exports for the example project `pr74qo`.

```
(dsscli) dss nfsexport list pr74qo
+----+-----------------+--------------+-------------+--------+
| ID | Container       | IP           | Access Mode | Status |
+----+-----------------+--------------+-------------+--------+
| 22 | pr74qo-dss-0002 | 10.156.29.72 | RW          | ACTIVE |
| 23 | pr74qo-dss-0002 | 10.156.29.73 | RW          | ACTIVE |
| 24 | pr74qo-dss-0001 | 10.156.29.74 | RW          | ACTIVE |
+----+-----------------+--------------+-------------+--------+
```

```
(dsscli) dss nfsexport list pr74qo --containername pr74qo-dss-0002
+----+-----------------+--------------+-------------+--------+
| ID | Container       | IP           | Access Mode | Status |
+----+-----------------+--------------+-------------+--------+
| 22 | pr74qo-dss-0002 | 10.156.29.72 | RW          | ACTIVE |
| 23 | pr74qo-dss-0002 | 10.156.29.73 | RW          | ACTIVE |
+----+-----------------+--------------+-------------+--------+
```

```
(dsscli) dss nfsexport list pr74qo --poolname 'DSS Testpool'
+----+-----------------+--------------+-------------+--------+
| ID | Container       | IP           | Access Mode | Status |
+----+-----------------+--------------+-------------+--------+
| 22 | pr74qo-dss-0002 | 10.156.29.72 | RW          | ACTIVE |
| 23 | pr74qo-dss-0002 | 10.156.29.73 | RW          | ACTIVE |
+----+-----------------+--------------+-------------+--------+
```

In order to view the details of a specific NFS export, you can use the `dss nfsexport show` command. The command takes the NFS export id of the export to show as mandatory argument:
(dsscli) dss nfsexport show 22

+-------------+----------------------------------------------------------+
| Field       | Value                                                   |
+-------------+----------------------------------------------------------+
| ID          | 22                                                      |
| Container   | pr74qo-dss-0002                                         |
| IP          | 10.156.29.72                                            |
| Access Mode | RW                                                      |
| Status      | ACTIVE                                                  |
| Mount Path  | datdsscest102.dss.lrz.de:/dss/dsstestfs01/pr74qo/pr74qo- |
| Expires     | None                                                    |
| Active Task | None                                                    |
| Created     | 2017-12-06T08:07:40.997646+01:00                         |
| Last Update | 2017-12-06T08:07:51.507205+01:00                         |
+-------------+----------------------------------------------------------+

8.4. Deleting exports

Dynamically managing container exports is unfortunately not yet supported because of a technical limitation in the underlying NFS server used. This problem will be addressed and fixed in a future release of DSS. If you need your container unexported via NFS to LRZ Cloud Machines or Virtual Machines for example, please open a ticket at the LRZ Service Desk for it. We will setup the exports as soon as possible for you.

9. Hints and possible pitfalls

9.1. Known Limitations

9.2. Do's and Don't's