Building the Model Research Data Portal

Ian Foster
foster@uchicago.edu
with Kyle Chard, Eli Dart, Steve Tuecke, Jason Williams
The Modern Research Data Portal: A Design Pattern for Networked, Data-Intensive Science

https://docs.globus.org/mrdp

The Modern Research Data Portal is a new design pattern for providing secure, scalable, and high performance access to research data.

GitHub Repo
provides code for the simple data portal that you can experiment with online
LEARN MORE

Example Data Portal
allows you to experiment with an example implementation of the design pattern
LEARN MORE

Code Walkthrough
provides a narrative description of the simple data portal code
LEARN MORE

Jupyter Notebook
demonstrates some Globus features described in the technical article
LEARN MORE

A technical article describes this design pattern, reviews representative examples at research laboratories and universities (see below), and uses coding examples to demonstrate how Globus APIs can be used to implement a range of research data portal capabilities.

https://doi.org/10.7287/peerj.preprints.3194v1
LEARN MORE
The legacy research data portal

Portal server applications:
- web server
- search
- database
- authentication
- data service

Browsing path
Query path
Data path

Filesystem
(data store)

Border Router
WAN
Firewall
Enterprise

Portal
Server
The modern research data portal

Data movement
Data sharing
Authentication

Portal server applications:
- web server
- search
- database

Data access

API DTNs
(data access governed by portal)

Portal Server

Filesystem (data store)
A key message: Outsource all that you can

- Outsource responsibility for determining user identities
- Outsource control over who can access different data and services within the portal
- Outsource responsibility for managing data uploads and downloads between various locations and storage systems
- Leverage standard web user interfaces for common user actions
The modern research data portal

WAN

Border Router

Firewall

Enterprise

Data Path

Data access

Portal server applications:
- web server
- search
- database

Portal Query/Browse Path

API DTNs
(data access governed by portal)

Filesystem (data store)

Science DMZ
Switch/Router

perfSONAR

perfSONAR

perfSONAR

Border Router

Data Path

portal Query/Browse Path

Portal server applications:
- web server
- search
- database

Portal Query/Browse Path

API DTNs
(data access governed by portal)

Filesystem (data store)
A simple example of MDRDP logic

User

Authenticate
Identify data
Download data

Portal

1. Create “shared endpoint”
2. Copy data to shared endpoint
3. Sets permissions on shared endpoint for user; notify user
4. (Eventually) delete shared endpoint
1. Create “shared endpoint”

2. Copy data to shared endpoint

3. Set permissions on shared endpoint for user; notify user

... 

4. (Eventually) delete shared endpoint
1. Create “shared endpoint”

2. Copy data to shared endpoint

3. Set permissions on shared endpoint for user; notify user

4. (Eventually) delete shared endpoint

# (2) Copy data into the shared endpoint

tc.endpoint_autoactivate/share_id

tdata = TransferData(tc, host_id, share_id,
  label='RDP copy', sync_level='checksum')

tdata.add_item(source_path, '/', recursive=True)

r = tc.submit_transfer(tdata)

tc.task_wait(r['task_id'], timeout=1000,
  polling_interval=10)

# (3) Enable access by user

r = ac.get_identities(usernames=email)

user_id = r['identities'][0]['id']

rule_data = {
  'DATA_TYPE': 'access',
  'principal_type': 'identity', # Grantee is
  'principal': user_id, # a user.
  'path': '/', # Path is /
  'permissions': 'r', # Read-only
  'notify_email': email, # Email invite
  'notify_message': # Invite msg
    'Requested data are available.'
}

tc.add_endpoint_acl_rule(share_id, rule_data)

# (4) Ultimately, delete the shared endpoint

tc.delete_endpoint(share_id)
1. Create “shared endpoint”

2. Copy data to shared endpoint

3. Set permissions on shared endpoint for user; notify user

... 

4. (Eventually) delete shared endpoint

```python
# (2) Copy data into the shared endpoint
tc.endpoint_autoactivate(share_id)
tdata = TransferData(tc, host_id, share_id,
                     label='RDP copy', sync_level='checksum')
tdata.add_item(source_path, '/', recursive=True)
r = tc.submit_transfer(tdata)
tc.task_wait(r['task_id'], timeout=1000,
             polling_interval=10)

# (3) Enable access by user
r = ac.get.identities(usernames=email)
user_id = r['identities'][0]['id']
rule_data = {
    'DATA_TYPE': 'access',
    'principal_type': 'identity', # Grantee is
    'principal': user_id,        # a user.
    'path': '/',                 # Path is /
    'permissions': 'r',          # Read-only
    'notify_email': email,       # Email invite
    'notify_message':            # Invite msg
        'Requested data are available.'
}
tc.add_endpoint_acl_rule(share_id, rule_data)

# (4) Ultimately, delete the shared endpoint
tc.delete_endpoint(share_id)
```
1. Create “shared endpoint”

2. Copy data to shared endpoint

3. Set permissions on shared endpoint for user; notify user

4. (Eventually) delete shared endpoint

---

```python
# (2) Copy data into the shared endpoint
tc.endpoint_autoactivate(share_id)
tdata = TransferData(tc, host_id, share_id,
                     label='RDP copy', sync_level='checksum')
tdata.add_item(source_path, '/', recursive=True)
r = tc.submit_transfer(tdata)
tc.task_wait(r['task_id'], timeout=1000,
             polling_interval=10)

# (3) Enable access by user
r = ac.get_identities(usernames=email)
user_id = r['identities'][0]['id']
rule_data = {
    'DATA_TYPE': 'access',
    'principal_type': 'identity',  # Grantee is user
    'principal': user_id,  # a user.
    'path': '/',  # Path is /
    'permissions': 'r',  # Read-only
    'notify_email': email,  # Email invite
    'notify_message':  # Invite msg
        'Requested data are available.'
}
tc.add_endpoint_acl_rule(share_id, rule_data)

# (4) Ultimately, delete the shared endpoint
tc.delete_endpoint(share_id)
```
An example MRDP

https://docs.globus.org/mrdp

Repository

Select some dataset(s) to transfer or click on a dataset name to browse its files.

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Select</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlanta International Airport Climate Data</td>
<td></td>
</tr>
<tr>
<td>Boston Logan International Airport Climate Data</td>
<td></td>
</tr>
</tbody>
</table>
Many variants possible

- Manage access to data at multiple locations
- Manage access to data on cloud
- Upload data for analysis
- Data download from scientific instruments
- Data publication
- Transfer data to computer for analysis
The Modern Research Data Portal: A Design Pattern for Networked, Data-Intensive Science

https://docs.globus.org/mrdp

The Modern Research Data Portal is a new design pattern for providing secure, scalable, and high performance access to research data.

GitHub Repo
provides code for the simple data portal that you can experiment with online

Example Data Portal
allows you to experiment with an example implementation of the design pattern

Code Walkthrough
provides a narrative description of the simple data portal code

Jupyter Notebook
demonstrates some Globus features described in the technical article

A technical article describes this design pattern, reviews representative examples at research laboratories and universities (see below), and uses coding examples to demonstrate how Globus APIs can be used to implement a range of research data portal capabilities.


https://doi.org/10.7287/peerj.preprints.3194v1

foster@uchicago.edu