EGI-Engage

Second Data Accounting Prototype

D3.15

Date	02 March 2018
Activity	WP3
Lead Partner	STFC
Document Status	FINAL
Document Link	https://documents.egi.eu/document/3029

Abstract

This report documents the release of the second prototype for dataset accounting during EGI-Engage, focused on dataset usage, which will be run as a test bed by the EGI Accounting Repository team for further improvements during future projects. A dataset is defined as a logical set of files which may exist in several places at once and to which it is possible to assign some form of persistent unique identifier. This report looks at how the usage metrics and architecture of this prototype have developed since the first prototype and at the testing of it against the EGI DataHub. It uses software from the APEL project, modified so that it can pull usage metrics from an HTTP interface. Exploitation and dissemination plans are presented, and lastly potential work as part of forthcoming projects is discussed. External work relevant to this prototype, which the APEL team has become involved in, is shown in the appendix.

The EGI-Engage project is co-funded by the European Union (EU) Horizon 2020 program under Grant number 654142 http://go.egi.eu/eng

COPYRIGHT NOTICE

DELIVERY SLIP

	Name	***Partner/Activity***	***Date***
From:	A. Coveney, G. Corbett	STFC / JRA1	2017-07-05
Moderated by:	Malgorzata Krakowian	EGI Foundation/WP1	
Reviewed by	Peter Solagna Miroslav Ruda	EGI Foundation/WP5 CESNET	
Approved by:	AMB and PMB		

DOCUMENT LOG

Issue	***Date***	***Comment***	***Author / Partner***
v0.1	2017-07-05	Document creation	A. Coveney / STFC
v0.2	2017-07-19	Review	D. Scardaci / EGI F. - INFN
v0.3	2017-08-08	External review	A. Coveney / STFC
FINAL	2017-08-25	Final review	A. Coveney / STFC
FINAL	2018-03-02	Updated version following the reviewers' recommendations after the final review (repeated introductory text removed, and service architecture removed)	A. Coveney / STFC

TERMINOLOGY

A complete project glossary and acronyms are provided at the following pages:

- https://wiki.egi.eu/wiki/Glossary
- https://wiki.egi.eu/wiki/Acronyms

Contents

Executive summary

This report documents the release of the second prototype for dataset accounting during EGI-Engage, focused on dataset usage, which will be run as a test bed by the EGI Accounting Repository team. A dataset is defined as a logical set of files that may exist in several places at once and to which it is possible to assign some form of persistent unique identifier and to perform dataset accounting it is assumed that this unique identifier is available.

The EGI Open Data Platform (which will provide capabilities to publish, use and reuse openly accessible data identified by PID) was chosen as a source of dataset accounting data. The planned design of this platform included the integration of current EGI storage services into the platform backend, which is intended to hide the complexity caused by the wide variety of storage systems.

This second prototype has the same high-level architecture as the first, which is detailed in D3.8[1].

The software was tested by running it and pointing a modified instance of SSM, designed to pull data from a REST endpoint, at the EGI DataHub[2], the Data as a Service (DaaS) built on top of the Open Data Platform, for a number of days as well as summarising the data received. This demonstrated the prototypes is capable of extracting space metrics from a test space, parsing them into an OGF based message format, loading the data into a database, and finally aggregating the data into summaries.

It is intended that this prototype will be improved during future projects by using feedback following this release to ensure it will meet user requirements. Additionally, since the first prototype, there are two external projects, SeaDataCloud and AtlantOS, which have relevance to this prototype that the APEL team has either become involved in or come into contact with. These will provide opportunities for future collaboration.

[1] https://documents.egi.eu/document/2968
[2] https://datahub.egi.eu/

1 Introduction

This report documents the release of the second prototype for dataset usage accounting, which is a development of the first prototype presented in D3.8[3]. In this context, a dataset is defined as a logical set of files which may exist in several places at once and to which it is possible to assign some form of persistent unique identifier and to perform dataset accounting it is assumed that this unique identifier is available.

APEL is an accounting tool that collects accounting data from sites participating in the EGI infrastructure as well as from sites belonging to other Grid organisations that are collaborating with EGI, including OSG and NorduGrid.

Table 1 provides a summary of the tool covered in this release.

Table 1 - APEL tool summary

Tool name	APEL – Dataset accounting feature
Tool URL	http://apel.github.io/
Tool wiki page	https://wiki.egi.eu/wiki/Accounting_Repository
Description	EGI Core Service – The Accounting Repository collects and stores user accounting records from various services offered by EGI.
Value proposition	Support for dataset usage accounting can aid site and experiment administrators in making decisions about the location and storage of datasets to make more efficient use of the infrastructure, and to assist scientists in assessing the impact of their work.
Customer of the tool	EGI
User of the service	EGI Accounting Repository
User documentation	https://twiki.cern.ch/twiki/bin/view/EMI/EMI3APELClient
Technical documentation	https://twiki.cern.ch/twiki/bin/view/EMI/EMI3APELClient
Product team	STFC
License	Apache License, Version 2.0
Source code	https://github.com/gregcorbett/apel/tree/dataset_accounting_v2 https://github.com/gregcorbett/ssm/tree/dataset_accounting_v2

[3] https://documents.egi.eu/document/2968

The dataset accounting prototype has been developed using the EGI DataHub[4] as a storage service to integrate against. The reasons for this were covered in the previous report on dataset accounting[5]. This second release of the dataset accounting prototype further improves on the functionality added in the first prototype and increases the integration with the EGI DataHub.

The outline of this deliverable is as follows: first we provide a short introduction to the components provided by the APEL project as part of this prototype. Then the high-level architecture of the tool and its components are described, along with the integrations and dependencies it has. Then the supported storage systems are described and the updated record metrics presented, with integration guidance given for other data storage systems. Release notes and the results of testing for this release are provided, followed by a dissemination and exploitation plan. Finally, a selection of future developments is shown. In the appendix is an overview of related work outside of this project.

2 Service architecture

2.1 High-Level service architecture

Details and diagrams of the high-level service architecture can be found in the report on the first prototype presented in D3.8[6].

The benefit of this new architecture is that it allows the Accounting Repository itself to control the rate of flow of accounting data into the Repository. The current services, in contrast, rely on sites sending data at appropriate time intervals, but there is always the chance of misconfiguration leading to the central Accounting Repository to receive many more messages than normal. Limitations of this architecture include the fact that separate extensions to the SSM software will be needed to parse the raw response from the REST APIs of each supported storage system. For example, Onedata[7] based systems, like the EGI DataHub, will return raw usage data in a different format to Ceph based systems. Also, pulling the records directly will require the REST interface to be available at the same time as the SSM initiates the pull. This differs from sending records via the Message Broker, which acts as a buffer, storing records for multiple days, allowing asynchronous publishing and consuming. Once the records are on the APEL server, they will be loaded and summarised. The portal is then updated via the Message Broker as happens now.

[4] https://datahub.egi.eu/
[5] https://documents.egi.eu/document/3025
[6] https://documents.egi.eu/document/2968
[7] https://onedata.org/

Since the first dataset accounting prototype, the ideas on how the prototype will interact with the message brokers has been revised.

It is possible that the extended SSM could be installed on a node separate to the APEL server; in this case the SSM could then use the existing SSM functionality to send the extracted messages via the message broker. The extended SSM could even be installed on a node within the storage system, access usage information internally, and send it via the message broker. However this has not been tested or developed as the prototypes have focused on making use of the external REST application programming interface (API) provided by the EGI DataHub.

As exposing a REST interface can be more technically challenging for a storage service to implement than using a centrally managed message broker service, there is still the option for a storage service to push data via message brokers. It will depend on the specific storage service as to which is the better system to exploit.

Aware that the current message brokers will be replaced by the REST based ARGO Messaging Service[8] (AMS), the REST functionality of the prototype has been developed such that it could be integrated with the REST functionality developed for the prototype AMS enabled SSM, by making the SSM more modular and able to handle different protocols, although this work is still ongoing.

2.2 Integration and dependencies

For this dataset usage accounting prototype, the central APEL server uses an updated SSM with support for pulling data directly from REST HTTP interfaces so that it can interact with EGI DataHub. It is possible these usage records could be sent over a messaging broker; however the prototype has been developed against the external OneData REST API[9], with the SSM querying the storage system directly.

The central APEL server can use the EGI service registry (GOCDB[10]) to get a list of endpoints so that only data from endpoints correctly defined in GOCDB is processed. This feature is not currently used by this prototype, as it is not clear yet how multiple separate instances of the same underlying technology, i.e. OneData, would be handled. They could be handled by separate SSM instances or one instance querying different destinations. In either case, determining how the new feature would integrate with GOCDB is considered a mandatory requirement to move DataSet accounting into production in the future.

2.3 Supported storage solutions

2.3.1 Onedata

.

[8] http://argoeu-devel.github.io/messaging/v1/
[9] https://onedata.org/docs/doc/advanced/rest/index.html
[10] http://goc.egi.eu/

Onedata[11], the underlying technology powering the EGI Open Data platform and DataHub[12], has been already introduced in D3.14[13]. It provides a REST API which can be used to extract space and user metrics. Now, It can provide persistent identifiers (PIDs), such as DOIs or its own type of identifier, which can be used to find the correct usage information to retrieve.

Additionally, the Onedata REST API provides metrics in a format that does not currently provide all proposed metrics present in the next section so some compromise will need to be found between the two. Also, since a single dataset can be divided between several storage providers, consideration should be made about how the metrics for a dataset can be collated from the data retrieved from disparate providers, although this was outside the scope of this prototype as it focused on Onedata Spaces mapping to a single provider.

The APEL server software was modified to support the loading of dataset usage records into a specifically designed database schema, and the APEL Secure Stomp Messenger (SSM) component was modified to support fetching dataset usage records from a REST interface (as opposed to sending messages via the EGI Message Brokers), which is the method that Onedata provides access to accounting data. Currently, the prototype uses a simple REST puller process, similar to the receiver used to retrieve messages from the EGI Message Brokers. Using this puller process in production does mean that effort would need to be spent supporting an additional interface to the Accounting Repository, although the added flexibility may be beneficial.

The APEL software has been modified to support loading of this new format into a database by starting a separate loader process with its own configuration file. This means that the prototype is capable of extracting the space metrics of the test space, parsing them into the OGF message format then loading the data into the database.

A lot of the metrics proposed are available internally to Onedata, but not all of them are exposed by the REST API and the ones that are use different keys, and some are not yet implemented (mainly PIDs, ORCIDs, and specific metrics about transfers). Additional modifications to the software are thus required to convert the data retrieved from the Onedata REST API into a format suitable for ingestion by the Accounting Repository and further collaboration will be required between the Onedata and APEL developers to ensure all the right metrics are exposed.

2.4 Updated record metrics

Table 2 shows an outline of the metrics that were proposed for performing dataset usage accounting. They are intended as an extension to the Open Grid Forum (OGF) Usage Record version 2 (UR-2.0)[14], and there are currently no metrics that are mandatory during this prototyping stage. The final implementation for Dataset Usage Accounting is likely to change as experience is gained

[11] https://onedata.org/
[12] https://datahub.egi.eu/
[13] https://documents.egi.eu/document/3025
[14] https://www.ogf.org/documents/GFD.204.pdf

developing the prototype into a production service and through knowledge sharing as part of the Accounting Team's related work (see Appendix I). This prototype supports a subset of these metrics but can easily be extended.

Compared to the first prototype, the metrics have been altered in the following way:

- AccessEvents has been split into ReadAccessEvents and WriteAccessEvents as OneData provides this separation. If, in future, supported systems did not provide this, a TotalAccessEvents field could be added. Although it is not expected that a dataset associated with a DOI would change and so require the recording of write events, there are different use cases for dataset accounting that might require this information. Some of these are covered in the previous report on dataset accounting[15].
- DataSet was renamed DataSetID, and an additional DataSetIDType was added to allow for future support for other, non DOI, unique identifiers, such as PIDs.
- Infrastructure was added so that we could determine all DataSet records corresponding to the same system, e.g the EGI DataHub. This attribute is partly for problem solving purposes, but it will also enable accounting for datasets identified by IDs that are not global but that are only guaranteed to be unique within a particular system or community. As such, this field should not be used in an ad-hoc manner, but be a choice from a list of options that have been agreed with an infrastructure or community.

Table 2 – Updated dataset accounting metrics

	Key	***Type***	***Description***
Record Identity Block	Resource provider	string	Resource provider at which the resource is located (e.g. GOCDB sitename)
	Infrastructure	string	High level system that the provider is part of (e.g. https://datahub.egi.eu)
Subject Identity Block	GlobalUserID	string	e.g. X.509 certificate DN / EGI unique ID (from Checkin service)
	GlobalGroupId	string	e.g. VO
	GlobalGroupAttribute	string	e.g. VO Group and/or Role
	ORCID	string	ORCID iD of the user
Dataset Usage	DatasetID	string	Unique identifier such as a PID / DOI
	DataSetIDType	string	PID, DOI, etc.

[15] https://documents.egi.eu/document/3025

Block	ReadAccessEvents	integer	Number of read operations in period
	WriteAccessEvents	integer	Number of write operations in period
	Source	IP address / other	Source of transfer at resource provider
	Destination	IP address / other	Destination of transfer
	StartTime	ISO 8601 timestamp	Start time of transfer
	Duration	ISO 8601 duration	Duration of transfer
	EndTime	ISO 8601 timestamp	End time of transfer
	TransferSize	integer	Bytes transferred
	HostType	string	Storage system Type
	FileCount	integer	Number of files accessed
	Status	string	Success / failure / partial transfer

The association of the ORCID of a user with a dataset usage record was not covered in the initial survey and was suggested at a later stage of the development process with the aim to help linking datasets to research publications. However, additional personal information should only be collected if there is a clear need and there is agreement between stakeholders. It may be better to remove the ORCID identifier from the basic dataset usage record and to gather it from third party services (e.g. from the Check-in service or directly from ORCID) only when needed, but for the moment it remains an optional metric.

Depending on how detailed the accounting data is, a method for aggregating this information should be created so that the volume of accounting data does not become unmanageable. This applies especially to the fields that relate to transfer operations as getting information for these fields could require quite a fine-grained approach to the usage data. To prepare for this, an example summariser process, which would run on the data in the Accounting Repository, has been developed, as well as a corresponding summary schema and message field for the storage and sending of this summarised data from the Repository to the Portal.

2.5 Integration guidance

Currently, it seems likely any external services that wanted to interact with the dataset accounting data would do so via the Portal (as they currently do for Job, Cloud and Storage accounting), so only integration between the dataset accounting repository and the Accounting Portal is required for users to retrieve information.

Any system that can provide a list of PIDs hosted at a given storage system could easily be integrated with the current prototype, by outputting the PIDs to a file to be later read by the SSM puller process. To integrate with the current prototype, they would ideally provide:

- A REST endpoint that can be queried to discover all the PIDs that need their usage to be accounted for, and an endpoint that can then be queried to extract the usage information for those PIDs using the list of PIDs provided by the first endpoint; or an endpoint that can be queried directly to return usage metrics for all datasets in that storage system. This would then be used by the accounting service to find and retrieve usage metrics.
- Documentation for their API that can be used to extract the correct metrics from the endpoint.

3 Release notes

These are the changes included in this prototype release of the APEL software compared to the first prototype:

- Added a function for retrieving a list of DOIs from a file which will also prevent the puller process from starting if no DOIs are found.
- Added new methods for interacting with the OneData API so that DOIs can be resolved to a specific Share and so that the version of the API can be configured.
- Changed method of retrieving usage metrics so that only complete days are recorded.
- Added a method for aggregating dataset accounting records into summary records and an internal model for that record type.
- Added support for loading and unloading dataset summary records from an APEL format database.
- Added an “Infrastructure” field to the schema, separated the “AccessEvents” field into “ReadAcessEvents” and “WriteAccessEvents”, and changed the “DataSet” field into “DataSetID” with an associated “DataSetIDType” field.

4 Result of testing

The second prototype focussed on assigning DOIs to OneData Shares (Shares are Spaces that have been made public) and querying the REST interface to determine the usage. Unlike the first prototype, which relied on a known Space, the second prototype requires a list of DOIs be provided to it via a file. This will allow for easy integration with any tool that can return a list of DOIs resolving to a given storage system.

The prototype reads the provided DOI and queries a resolver to determine what object the DOI resolves to. Currently, due to limitations of the OneData interface, only DOIs that resolve to OneData Shares are supported.

From the resolved DOI, the prototype can extract an identifier for the Share associated with that DOI, the ShareID. Through multiple queries to the specified instance of OneData, the SpaceID corresponding to the ShareID and the Provider URL are determined. A final REST request is then sent to the Provider to determine the data access.

As before, the raw data response to the request is then processed and parsed into an XML format based on the OGF Usage Record, before being loaded into a database by the APEL server.

These records are then summarised in a similar way to Job and Cloud records. These summarise can then be unloaded in preparation for sending on to the Accounting Portal. The loading of these summaries has been partially tested, by getting the APEL server to load messages into memory, but not save them to a database. This testing was also done locally, not with the portal

The integration with OneData has been tested over 9 days by running the accounting software with a previously known DOI and pointing an instance of the SSM, modified to allow interaction with a REST endpoint, at the EGI DataHub instance of Onedata to extract usage data for the test DOI.

By determining the Provider of the Share corresponding to the DOI a JSON response like the following can be retrieved by querying the REST API (e.g. https://datahub.plgrid.pl/api/v3/oneprovider/metrics/space/1I8DOQUXXiezOAcTpAewz40HVNzy-Sr2mlBZZtEmpA?metric=storage_quota&step=1m) for the metrics of the space that the Share is stored in:


```
{
"rrd" : {
    "meta" : {
        "step" : 86400,
        "start" : 1494892800,
        "legend" : ["space BM-Qz-eXNLtjafuRbr6B7fsLdUJ-GXjaRiqH-nadcx0; metric data_access; oneprovider ID q-bwPyZKSqs-ZQYwggWtSkak77hoOAC5479MEYtG2jw; data_access_read[bytes/s]","space BM-Qz-eXNLtjafuRbr6B7fsLdUJ-GXjaRiqH-nadcx0; metric data_access; oneprovider ID q-bwPyZKSqs-ZQYwggWtSkak77hoOAC5479MEYtG2jw; data_access_write[bytes/s]"],
        "end" : 1497571200
        },
        "data" : [[null,null],[null,null],[null,null],[null,null],[null,null],[null,null],
```

```
[null,null],[null,null],[null,null],[null,null],[null,null],[null,null],[
null,null],[null,null],[null,null],[null,null],[null,null],[null,null],[n
ull,null],[null,null],[null,null],[null,null],[null,null],[0.0,0.0],[0.0,
0.0],[0.0,0.0],[0.0,0.0],[0.0,0.0],[0.0,0.0],[0.0,0.0],[null,null]],
        "about" : "RRDtool graph JSON output"
        },
    "providerId" : "q-bwPyZKSqs-ZQYwggWtSkak77hoOAC5479MEYtG2jw"
}
```


The JSON output has more data than the version retrieved with the first prototype. The returned data was then parsed into a more complete message format based on the OGF Usage Record to give the following XML record:


```
<?xml version="1.0" encoding="UTF-8"?>
<ur:UsageRecords                                    xmlns:ur="http://eu-
emi.eu/namespaces/2017/01/datasetrecord">
    <ur:UsageRecord>
        <ur:RecordIdentityBlock>
            <ur:RecordId>https://datahub.egi.eu:8443-
10.5072/OXFORDFLOWERDATASET.1-1497520702</ur:RecordId>
            <ur:CreateTime>1497520702</ur:CreateTime>
            <ur:ResourceProvider>q-bwPyZKSqs-
ZQYwggWtSkak77hoOAC5479MEYtG2jw</ur:ResourceProvider>
        </ur:RecordIdentityBlock>
        <ur:SubjectIdentityBlock>
        </ur:SubjectIdentityBlock>
        <ur:DataSetUsageBlock>
            <ur:DataSetID>10.5072/OXFORDFLOWERDATASET.1</ur:DataSetID>
            <ur:DataSetIDType>DOI</ur:DataSetIDType>
            <ur:ReadAccessEvents>0</ur:ReadAccessEvents>
            <ur:WriteAccessEvents>0</ur:WriteAccessEvents>
            <ur:StartTime>1497484800</ur:StartTime>
            <ur:Duration>86400</ur:Duration>
            <ur:EndTime>1497571200</ur:EndTime>
            <ur:HostType>OneData</ur:HostType>
```

```
        </ur:DataSetUsageBlock>
        </ur:UsageRecord>
</ur:UsageRecords>
```

The message was then saved for future loading, as currently happens with messages received via the message broker network. The message was then loaded into a database by starting a separate loader process with its own configuration file, modified to support the loading of this new format.

Example of a usage record in database:

```
UpdateTime: 2017-06-27 07:00:09
RecordId:       https://datahub.egi.eu:8443-10.5072/OXFORDFLOWERDATASET.1-
1498543204
CreateTime: 2017-06-27 06:00:04
ResourceProvider: q-bwPyZKSqs-ZQYwggWtSkak77hoOAC5479MEYtG2jw
Infrastructure: https://datahub.egi.eu:8443
GlobalUserId: None
GlobalGroupId: None
ORCID: None
DataSetID: 10.5072/OXFORDFLOWERDATASET.1
DataSetIDType: DOI
ReadAccessEvents: 56
WriteAccessEvents: 1
Source: None
Destination: None
StartTime: 2017-06-26 00:00:00
Duration: 86400
EndTime: 2017-06-27 00:00:00
TransferSize: NULL
HostType: OneData
FileCount: NULL
Status: None
```

Example of a summary record in the database:

```
UpdateTime: 2017-06-28 08:52:24
ResourceProvider: q-bwPyZKSqs-ZQYwggWtSkak77hoOAC5479MEYtG2jw
Infrastructure: https://datahub.egi.eu:8443
GlobalUserId: None
GlobalGroupId: None
ORCID: None
DataSetID: 10.5072/OXFORDFLOWERDATASET.1
DataSetIDType: DOI
TotalReadAccessEvents: 56
TotalWriteAccessEvents: 1
Source: None
Destination: None
EarliestStartTime: 2017-06-19 00:00:00
TotalDuration: 777600
LatestStartTime: 2017-06-27 00:00:00
Month: 6
Year: 2017
TotalTransferSize: NULL
HostType: OneData
TotalFileCount: NULL
Status: None
```

Example of an unloaded summary:

```
<?xml version="1.0" encoding="UTF-8"?>
<ur:UsageSummaryRecords                                    xmlns:ur="http://eu-
emi.eu/namespaces/2017/01/datasetsummary">
<ur:UsageSummaryRecord>
<ur:CreateTime>2017-06-28T10:12:21Z</ur:CreateTime>
<ur:ResourceProvider>q-bwPyZKSqs-
ZQYwggWtSkak77hoOAC5479MEYtG2jw</ur:ResourceProvider>
<ur:Infrastructure>https://datahub.egi.eu:8443</ur:Infrastructure>
<ur:DataSetID>10.5072/OXFORDFLOWERDATASET.1</ur:DataSetID>
```

```
<ur:DataSetIDType>DOI</ur:DataSetIDType>
<ur:TotalReadAccessEvents>56</ur:TotalReadAccessEvents>
<ur:TotalWriteAccessEvents>1</ur:TotalWriteAccessEvents>
<ur:EarliestStartTime>2017-06-19T00:00:00Z</ur:EarliestStartTime>
<ur:TotalDuration>777600</ur:TotalDuration>
<ur:LatestStartTime>2017-06-27T00:00:00Z</ur:LatestStartTime>
<ur:Month>6</ur:Month>
<ur:Year>2017</ur:Year>
<ur:HostType>OneData</ur:HostType>
</ur:UsageSummaryRecord>
</ur:UsageSummaryRecords>
```

Due to the limitations of the current Onedata implementation, it was not possible to extract the owner of the Share. The initiator of access events was not captured due to complexities extracting both usage assigned to a user and anonymous usage and the fact the test usage was anonymous.

5 Plan for exploitation and dissemination

Name of the result	Prototype dataset usage accounting system
DEFINITION	
Category of result	Software & service innovation
Description of the result	This prototype system extends the types of usage accounting that the EGI Accounting Repository can perform by adding features to support dataset usage accounting.
EXPLOITATION	
Target group(s)	RIs, international research collaborations, storage providers
Needs	Provide sufficient information about the location and storage of datasets to make more efficient use of computing infrastructures. Enable scientists to assess the impact of their work.
How the target groups will use the result?	With the right information on dataset usage, a dataset provider (i.e. an infrastructure provider like EGI) could create multiple replicas of a dataset if it is requested many times, and a scientist can know how many people have accessed the dataset created with their research.
Benefits	Demonstrate the potential that dataset usage accounting has to aid in fulfilling the needs above. Allow the Accounting Repository team to gather more specific feedback on dataset accounting and to identify any potential issues that will need to be overcome in future.
How will you protect the results?	Open source license (Apache License, Version 2.0)
Actions for exploitation	The prototype will be run by the Accounting Repository team as a test bed for future developments of dataset usage accounting. Selected resource providers will be asked to make REST endpoints available so that the Accounting Repository can extract dataset accounting records to further test the prototype. Feedback will be solicited from potential users of dataset accounting on how useful the current prototype is and what new features they would like to be included in the future. The software will be made available in a public repository.
URL to project result	https://github.com/gregcorbett/apel/tree/dataset_accounting

	https://github.com/gregcorbett/ssm/tree/dataset_accounting
Success criteria	Positive feedback received from customers
DISSEMINATION	
Key messages	Test system for dataset accounting can be made available so that feedback can be gathered.
Channels	Operations Management Board meetings, EGI Engagement channels, Competence Centres
Actions for dissemination	Present results at an OMB and solicit feedback on prototype
Cost	N/K
Evaluation	Quality of feedback

6 Future plans

It is intended that this prototype will be improved during future projects by using feedback following this release to ensure it will meet user requirements. The optimum balance between accounting granularity and data volume still needs to be investigated as well as Portal views for the data.

The resources requirements for running dataset accounting as a production service should be quite similar to the current services, namely a central database and a process to retrieve the accounting data. The current production services all run on the same physical host, but if pulling from a larger number of different REST interfaces is required, it may be beneficial to run separate virtual hosts to query these endpoints.

As discussed throughout this report, this new dataset accounting functionality could be integrated with a message broker system. An example model to adopt would be the same model as currently used in Grid, Cloud, or Storage Accounting; the usage is extracted at the Provider level by the APEL client, or a light weight script supplied with the tool to produce accounting records, and then send them to APEL using SSM via the brokers. Although this would likely require a re-implementation of the functionality developed so far. However, this is no different to the other types of accounting where bespoke scripts are used to extract accounting information from the different systems and so should take a similar amount of effort.

A current technical limitation is that two Python libraries are required to extract usage from OneData. httplib is required to make a HEAD request to a DOI resolver, but httplib cannot connect to the EGI DataHub due to version limitations of the APEL system. Connecting to the DataHub is handled by urllib2, but urllib2 cannot make the HEAD requests to the DOI resolver. This does make the new code more difficult to maintain. A move to a lightweight script supplied with OneData

may help to alleviate this problem, as there may be more freedom to upgrade library/Python versions or change languages.

In parallel with this EGI-based work, the APEL team is also involved in a number of other projects that are related to dataset usage. These are detailed further in Appendix I.

Appendix I. Related work

SeaDataCloud

The SeaDataNet[16] pan-European infrastructure connects over 100 marine data centres and provides discovery and access to data resources for all European researchers. However, more effective and convenient access is needed to better support European researchers.

SeaDataCloud[17] aims at considerably advancing SeaDataNet services and increasing their usage, adopting cloud and HPC technology for better performance. Data cover the wide range of in situ observations and remote sensing data. To have access to the latest cloud technology and facilities, SeaDataNet will cooperate with EUDAT[18], a network of computing infrastructures that develop and operate a common framework for managing scientific data across Europe. SeaDataCloud will improve services to users and data providers, optimise connecting data centres and streams, and interoperate with other European and international networks.

There is currently no data management middleware that exposes metrics appropriate for dataset accounting. To this end, the APEL team has become involved in the SeaDataCloud project to work on adding dataset usage accounting. Development work will be needed in one or more of the EUDAT services to gather the appropriate metrics and aggregate them in the APEL Repository.

AtlantOS

AtlantOS[19] is a BG 8 (Developing in-situ Atlantic Ocean Observations for a better management and sustainable exploitation of the maritime resources) research and innovation project that proposes the integration of ocean observing activities across all disciplines for the Atlantic, considering European as well as non-European partners.

As part of this, there is a group looking at the practices and tools used with logging dataset usage in networks or infrastructures (e.g. SeaDataNet). These practices should be homogeneous enough so that the logs of different data services can be aggregated together so that homogenous statistics on dataset usage can be computed. These statistics will especially be useful for data providers (e.g. platform operator, organisation) who would like to have a feedback on the usage of their datasets.

The APEL team is monitoring the progress of this group to learn what conclusions they draw about interoperable dataset usage logging.

[16] https://www.seadatanet.org/
[17] https://www.seadatanet.org/About-us/SeaDataCloud
[18] https://www.eudat.eu/
[19] https://www.atlantos-h2020.eu/