Grassroots Enhancements for iRODS

Simon Tyrrell
The Earlham Institute
Norwich Research Park,
Norwich, NR4 7UZ, UK
simon.tyrrell@earlham.ac.uk

Xingdong Bian
The Earlham Institute
Norwich Research Park,
Norwich, NR4 7UZ, UK
xingdong.bian
@earlham.ac.uk

Robert P. Davey
The Earlham Institute
Norwich Research Park,
Norwich, NR4 7UZ, UK
robert.davey@earlham.ac.uk

ABSTRACT

This paper explains tools we have created to enhance the interoperability and usability of iRODS storage. We have added functionality to the eirods-dav Apache module to expose iRODS collections, data objects and metadata as Frictionless Data Packages. Along with this we have created a tool to enhance the functionality of iRODS client commands within a BASH terminal by allowing for TAB key filename completion for iRODS collections and data objects.

Keywords

iRODS, FAIR Data, Frictionless Data, Bash

INTRODUCTION

The Grassroots Infrastructure [1] project aims to create an easily-deployable suite of computing middleware tools to help users and developers gain access to scientific data infrastructure that can easily be interconnected. With the data-generative approaches that are increasingly common in modern life science research, it is vital that the data and metadata produced by these efforts can be shared and reused. The Grassroots Infrastructure project wraps up industry-standard software tools along with our own custom open-source software tools to give a consistent API that can be federated with others in terms of both data and services. This means institutions and groups can deploy a simple lightweight software suite, locally or as a virtual machine, to expose institutional data, connect up any existing data services, and federate their instance of Grassroots with other remote instances.

One of the major aims of the Grassroots Infrastructure is to allow users to share their wheat data, although it is by no means organism-specific, as easily and seamlessly as possible based upon FAIR data principles [2]. For data storage, we use the iRODS [3] data grid system that gives users access to potentially differing file systems and data resources through a single data abstraction layer. Users are able to carry out typical filesystem actions as normal, such as creating files and directories and maintaining permissions, along with extra features such as distributed storage viewable across different institutions and the ability to add extensive metadata to files and directories.

Using iRODS as our data storage infrastructure, we looked to see what we could do to enhance its functionality and usability for both internal users accessing the system using the standard iCommands and external users by adding interoperability APIs. We now describe some functionality that we have added in both of these areas.

FRICTIONLESS DATA

The Grassroots Infrastructure is an integral part of the Designing Future Wheat (DFW) programme [4] and one of the tasks it is used for is to host a portal to share the datasets produced within the programme. These datasets can vary greatly in size and their data is heterogeneous so one of our goals is to standardise access to these datasets. As part of the goal to make the data FAIR as much as possible, we looked at how we could increase the interoperability

iRODS UGM 2021 June 8-11, 2021, Virtual

of these datasets. Given their heterogeneous nature, we needed a solution that was both flexible and extensible, along with having good support and tooling for any APIs that are provided. The solution that we found was Frictionless Data [5] which are an open set of data standards and provides software to work with data. Frictionless Data Packages are a container used to describe and package collections of data into JSON objects. These packages can store any types of data and metadata and be used with an existing set of Frictionless schemas as well as being extensible by using any custom schemas as well.

Basic Frictionless Data functionality

The Frictionless Data support can be configured to only be active at particular points in the Apache directory hierarchy using the `DavRodsFrictionlessData` configuration directive. For example, to have the data packages appear in all of the top-level child directories of */data*, but not in subsequent child folders of these, the configuration would be:

```
# Generate Data Packages for all child directories directly below /data
<LocationMatch "/data/[^\/]+/">
   DavRodsFrictionlessData true
</LocationMatch>

# Since Data Packages are generated for the child directories configured in
# the line above, exclude all directories further down
<LocationMatch "/data/[^\/]+/[^\/]+/">
    DavRodsFrictionlessData false
</LocationMatch>
```

To give a useful visual indicator to denote the Frictionless Data Packages files within the generated listings pages, a custom image can be specified using the `DavRodsFDDataPackageImage` configuration directive which specifies the path to the image file to use for these packages. For example,

```
DavRodsFDDataPackageImage /eirods_dav_files/images/archive
```

would use the image at */eirods_dav_files/images/archive* to represent the Data Packages in the generated web page listings.

Metadata configuration

Each of the datasets come with a standard set of metadata based upon the Minimum Information About a Plant Phenotyping Experiment (MIAPPE) [6] which is an open set of standards to allow for easier integration of data from plant phenotyping experiments. Each project has metadata that contains the list of authors, the name of the project that the dataset comes from, a title for the project, a description of the project, the licence and a unique persistent identifier for the dataset. These metadata values all exist within the Frictionless Data Package standard thus making the task of connecting the iRODS and Frictionless technologies together straightforward. The default mappings of iRODS metadata keys to their Frictionless counterparts along with the relevant Apache module configuration directives are shown in table 1. All of these values are completely configurable using the Apache module configuration directives. For example, to use the value associated with the metadata key *project_title* from the iMeta catalog for the title field of the Frictionless Data Package, you would need the following configuration setting:

```
DavRodsFDResourceDescriptionKey project_title
```

The metadata values stored in iRODS can be combined where necessary to produce the value required for the Frictionless Data Packages. The configuration directives can be set by a comma-separated string containing the iRODS

Data Package field	Default iRODS metadata key	Apache Configuration Directive
license_name	license	DavRodsFDResourceLicenseNameKey
license_url	license_url	DavRodsFDResourceLicenseUrlKey
description	description	DavRodsFDResourceDescriptionKey
name	name	DavRodsFDResourceNameKey
authors	authors	DavRodsFDResourceAuthorsKey
title	title	DavRodsFDResourceTitleKey
id	id	DavRodsFDResourceIdKey

Table 1. Default mapping of metadata keys between iRODS and mod_eirods_dav Frictionless Data Packages and the Apache module configuration directives.

metadata keys to use. For instance, if the value that you wish to use for the description is given by concatenating the *short_info* and *detailed_info* metadata values, then the configuration would be:

```
DavRodsFDResourceDescriptionKey short_info,detailed_info
```

As well as specifying the keys, white space, periods / full stops and newlines can be specified in the configuration value too. With the example above, if you would like to have a more complex value generated using the *short_info* metadata value followed by a period / full stop and two blank lines (denoted by the \n escape code), then the *detailed_info*, a space and the *footnote* metadata value, the configuration would be

```
DavRodsFDResourceDescriptionKey short_info,.,\n,\n,detailed_info, ,footnote
```

Frictionless Data tabular support

One of the schemas available as part of the Frictionless Data standards is for Tabular Data Resources which is designed to represent tables of data. Eirods_dav has the ability to automatically take tabular data objects such as csv or tsv files and store them as Tabular Data Resources. This is done by querying the iMeta catalog for the given data object. The first required key is *column_headings* which has a comma-separated list of the column headings for the tabular file. For each of these headings an additional key-value pair specifies the type of data in the given column of the file. The keys for these are the column name with a *_type* suffix. The core values that these types can take are defined as part of the Frictionless Data Table Schema [7] which in turn are based open those from JSON Schema [8]. These include types such as strings for text, number for floating point values, integer for whole numbers, *etc.*

For example, consider a file *data.csv* which has three columns containing a string, an integer and a floating point number respectively, shown below

```
var1,var2,var3
A,1,2.1
B,3,4.5
```

Using the Tabular Data Resource type definitions, the types of these columns are string, integer and number respectively. Eirods_dav begins by looking for the *column_headings* key in the iMeta catalog for *data.csv* which in this case

would be

```
column_headings: var1,var2,var3
```

This value is then tokenized by splitting this string into separate values using the comma delimiters. In our example this would be*var1*, *var2* and *var3*. Eirods_dav then appends *_type* to each of these to produce the names of the iMeta keys to use to query for the datatype of each column. In this case, these would be *var1_type*, *var2_type* and *var3_type*. This would make the required metadata key-value pairs

```
var1_type: string
var2_type: integer
var3_type: number
```

Frictionless Data Package storage

By default, the Frictionless Data Packages are virtual and generated on the fly for each incoming request. Although this may be fine for smaller datasets, as they grow in size the time that it takes to generate these packages increases and thus may have a perceivable impact upon serving these requests. To address this problem, there is the ability to write the generated Frictionless Data Package back to its parent collection, in effect, caching it. This is done by setting the configuration directive `DavRodsFDSaveDataPackages` to `true` and is equivalent to running *iput* for the generated file. If the content of the dataset changes, this generated file would become stale and need to be regenerated. Eirods_dav checks for the existence of this file upon each request so by simply deleting this Frictionless Data Package file, eirods_dav will automatically regenerate it upon the next relevant incoming request.

BASH

One of the features that is available in many command-line environments is the ability to generate matching paths given partial file and directory names. For users on Linux, the common command-line terminal is Bash. An extremely useful feature of Bash is the ability to complete any text that has been typed in the shell up to that point by pressing the TAB key. As standard, iRODS does not come with this feature for the components within its iCommands package. As the number of nested levels within the iRODS zones increases, the time that it takes to type these commands and the likelihood of mistyping a path becomes greater, and the usefulness of auto-completion is lost

To address this issue, we created iRODS Bash Completer[9] to fill this gap by implementing tab-completion for the various iRODS client iCommands. The Bash shell can take advantage of the iRODS Bash Completer to auto-complete paths for the data hosted in iRODS in the standard way as it does for other mounted file systems. It does not require any escalated user privileges to use and has no dependencies other than the libraries within the iRODS development package. When the TAB key is pressed, it parses all of the available iRODS zones that the current user has access to and produces a list of data objects and collections that match the partial path that the user has currently typed in. If no text has yet been entered, it can be used to traverse through the entire hierarchy of the iRODS-hosted data. The installation process requires a small number of steps beginning with building the irods_bash_completer program using the *makefile* within the repository. Once this is built it can be copied anywhere that the user wishes. The iRODS Bash Completer can be configured for each of the iCommands, or indeed any third party application, by editing the supplied *bash/icommmands.d* file to specify where the *irods_bash_completer* has been installed. This is done be setting setting the first part of the `opts` variable within this file to the path where the iRODS Bash Completer has been installed. For example, to specify that it has been installed to */home/billy/bin/irods_bash_completer* the initial

part of the *bash/icommmands.d* would be

```
_irods_completer()
{
    local cur prev opts
    COMPREPLY=()
    cur="${COMP_WORDS[COMP_CWORD]}"
    prev="${COMP_WORDS[COMP_CWORD-1]}"
    opts=‘/home/billy/bin/irods_bash_completer ${cur}‘
```

Any other iRODS-aware commands can have TAB completion enabled by adding an entry to the end of this file. For example, this functionality can be enabled for a third party program called *irods_examine*, say, by adding the following snippet

```
complete -F _irods_completer irods_examine
```

to the end of *bash/icommmands.d*.

Once you have completed the configuration, copying *bash/icommmands.d* into the */etc/bash_completion.d/* directory will complete the installation and enable the TAB completion functionality.

ACKNOWLEDGMENTS

The Grassroots project is strategically funded through the BBSRC Designing Future Wheat programme grant, BB/P016855/1, and aims to develop a lightweight reusable set of open source software tools to allow researchers to share and federate life science datasets.

AVAILABILITY

The source code is for iRODS Bash Completer is available at `https://github.com/TGAC/irods_bash_completer` under the Apache License Version 2.0. The source code for mod_eirods_dav is available at `https://github.com/billyfish/eirods-dav` under the GNU Lesser General Public License version 3.

REFERENCES

[1] Grassroots Infrastructure, `https://grassroots.tools`, Visited last on 06.06.2021

[2] FAIR Principles `https://www.go-fair.org/fair-principles/`, Visited last on 06.06.2021

[3] Hao Xu, Terrell Russell, Jason Coposky, Arcot Rajasekar, Reagan Moore, Antoine de Torcy, Michael Wan, Wayne Shroeder, Sheau-Yen Chen: iRODS Primer 2: Integrated Rule-Oriented Data System. Synthesis Lectures on Information Concepts, Retrieval, and Services, Morgan & Claypool (2017)

[4] Designing Future Wheat, `https://designingfuturewheat.org.uk/`, Visited last on 06.06.2021

[5] Frictionless Data, `https://frictionlessdata.io/`, Visited last on 06.06.2021

[6] MIAPPE, `https://www.miappe.org/`, Visited last on 06.06.2021

[7] Frictionless Data Table Schema Types, `https://specs.frictionlessdata.io/table-schema/#types-and-formats`, Visited last on 06.06.2021

[8] JSON Schema, `https://datatracker.ietf.org/doc/html/draft-zyp-json-schema-03#section-5.1`, Visited last on 06.06.2021

[9] Tyrrell, S.: iRODS Bash Completer, `https://github.com/TGAC/irods_bash_completer`, Visited last on 06.06.2021