Deliverable Title	**D6.3.1 Data services implementation**
Deliverable Lead:	INRIA
Related Work package:	WP6
Author(s):	Franca Perrina, Brian Pickering, Steve Taylor
Dissemination level:	Public
Due submission date:	30/04/2011
Actual submission:	
Project Number	258142
Instrument:	IP
Start date of Project:	01/06/2010
Duration:	30 months
Project coordinator:	THALES

Abstract	This document presents the first version of the Testbed Infrastructure Data Service prototype, giving a brief overview of the proposed solution and installation instructions of the implemented prototype.

Project funded by the European Commission under the 7th European Framework Programme for RTD - ICT theme of the Cooperation Programme.

License

Versioning and Contribution History

Version	Date	Modification reason	Modified by
0.1	28/04/2011	ToC proposal	Brian Pickering (ITI), Franca Perrina (INRIA)
0.15	04/05/2011	ITI Contribution	Brian Pickering (ITI)
0.2	05/05/2011	INR Contribution	Franca Perrina (INRIA)
0.3	05/05/2011	Comments/updates	Brian Pickering (ITI)
0.4	29/07/2011	Changed title and added Executive Summary as requested in Review Report	Brian Pickering (ITI)

TABLE OF CONTENT

LIST OF ACRONYMS

iCAT	*iRODS metadata catalogue*
iRODS	*integrated Rule Oriented Data System*
RPRS	*Research Platform Repository Service*
TCI	*TEFIS Connectors Interface*
TIDS	*Testbed Infrastructure Data Service*

1. Executive Summary

The TEFIS Data Services (Research Platform Repository Service, RPRS, and Testbed Infrastructure Data Service, TIDS) provide support for the maintenance of any data which is associated directly with an experiment and the individual testruns within that experiment. Such data would typically include any input information, such as configuration settings, test scripts and the like, as well as any results produced as a result of running the experiment, such as platform monitoring output and data coming directly from the application as it runs. The input data will include information about the environment set up (the hardware and software configuration), although generic information relating to the test resources used to run the experiments is stored in the TEFIS Resource Directory: it is only when those resources are identified and configured for a specific experiment arrangement that a copy will be stored in the RPRS.

Given that the TEFIS Data Services are intended to store and maintain data pertaining to experiments in this way then testbeds and other external parties, including the experimenters themselves and anyone interested in their work, naturally have an interest in being able to access the data as well add to the data. A testbed running a testrun or part of a testrun, for instance, would typically need access to configuration and application settings to be able to execute the test application, and then to return results to an appropriate location for later review and processing by the experimenter or as input for a subsequent testrun phase to be run on a different testbed. This document describes the mechanisms and associated data items specifically directed towards external users of the TEFIS facilities, the experimenters and the testbed providers.

The document is structured as follows:

- The *General Concepts* section recalls some of the main concepts identified in the requirement analysis phase and used as a guide for the design of the proposed solution for the Data Services;
- *The TEFIS Data Service Architecture overview* introduces the role of the TIDS and RPRS in the TEFIS Architecture;
- *RPRS/TIDS* gives an overview of:
    - a *Summary Approach* to the solution adopted for Data Services, including the way a test run is handled from a data perspective, according to the *RPRS/TIDS File System*, and including the different *Architectural Options* available;
    - what the main *Testbed Interactions* supported are;
    - how the annotation of experimental data is allowed and what kind of *Experimental Metadata* are expected to be managed;
- *TIDS Installation* provides specific instructions for the TIDS installation, both on the *Client side* and *Server side*.

The document is strongly related to [4], which describes the RPRS prototype, and to [5], which represents the reference guideline for all data treatment in TEFIS.

2. Introduction

The TEFIS Data Services support all data exchange and management within the TEFIS platform itself as well as in connection with any remotely connected testbed or facility. The Research Platform Repository Service (RPRS) [4] provides support locally for the data management requirements of the TEFIS components responsible for the planning and execution of experiments. The Testbed Infrastructure Data Services (TIDS) manage the exchange of experimental data with the testbed responsible for the execution of a particular testrun or –runs. This document describes the implementation of the TIDS.

The TIDS implementation needs some detailed clarification in terms of:

1. *Where data are to be stored, where they may be retrieved from, and where any results should be written to*. As configuration and application data are created and stored by the Experiment Manager, a testbed needs to be directed where those data are located to be able to execute the appropriate test. Similarly, once a testrun completes, any output, including monitoring data, need to be transferred to an appropriate location.

2. *Where data or treatments originated*. In managing experiments for end-users, TEFIS must record how the results for a specific experiment were created, not least to support any attempts at reproducibility. Such provenance information is described here in terms of metadata associated with an experiment and/or testrun.

3. *How data are to be managed*. Once data have been created, TEFIS cannot hold on to them indefinitely; may need to restrict access; or may need to transfer them elsewhere. The initial approach to data curation is also defined here.

In the sections which follow, the TIDS implementation is described along with deployment options and the respective benefits of each. This is followed by a description of interactions between testbed and TEFIS. The discussion of the Data services implementation finally includes some discussion of aspects of provenance and curation at their current level.

3. General Concepts

This section summarises what was presented in [5]. The TIDS deployment in this document along with the RPRS implementation described in [4] represent the first version of the realisation of what was outlined in that document.

3.1. TIDS and RPRS

Figure 1 shows the component structure of the TEFIS platform. The Presentation Layer (shown in yellow at the top of the figure) represents the experimenter's view on the platform.

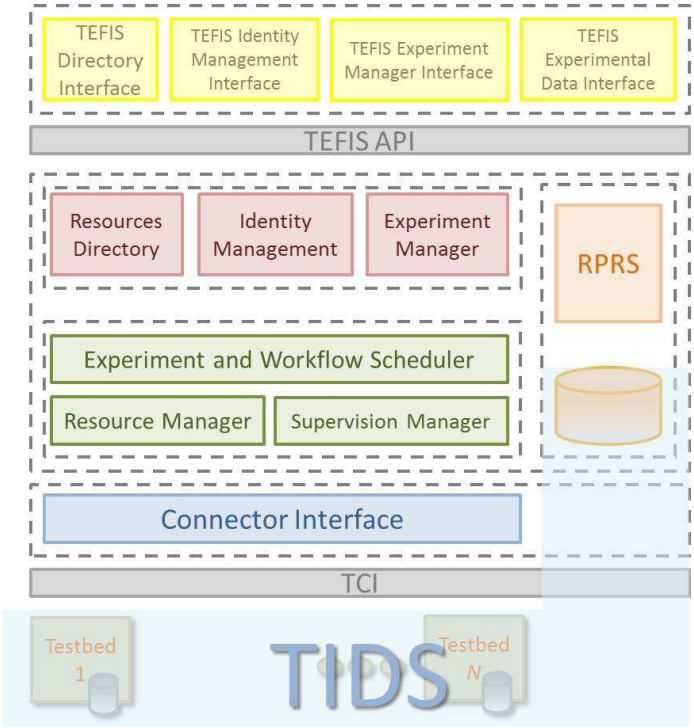

Figure 1: The TEFIS TIDS in relation to the RPRS

The Connector Interface (blue) and corresponding TCI provides the testbed provider's view on TEFIS. As outlined above, the RPRS is the main data service for all data pertaining to an experiment throughout its lifetime. It is one of the components within TEFIS, is responsible for creating the experimental data folder structure (Figure 3), and for the maintenance of data items and metadata within that structure.

Conversely, the TIDS provides the external view and interface onto the TEFIS data services. The TIDS provides a channel and mechanism for external parties, principally the testbed providers but also experimenters, to be able to interact with the data and metadata associated with experiments held in the TEFIS Data Services repository maintained and managed by the RPRS. Additionally, the TIDS acts as a distributed data store where big data/sensitive data, etc are held, whereas links or references to them would be held within the central data service, the RPRS. In fact one of the requirements behind the overall TEFIS Data Services design is the need for an efficient and flexible approach, to keep data distributed across the original sites, but making them accessible to the users (experimenters or testbed providers) when appropriate, through the TIDS service in charge of remote-to-remote transfer.

As explained in greater detail in [4], both the TIDS and the RPRS prototypes are based on iRODS [1], which provides a virtualised filesystem where data distributed across multiple sites can be shown in a unified virtual collection and be located, annotated and searched with appropriate tools.

3.2. The Test Profile

The *test profile* was introduced in [5] as an umbrella term to contain all the relevant metadata and links to real data associated with a given experiment. In reality, the *test profile* is the collection of data contained within the TEFIS experiment folder structure, and the metadata on individual folders and files. Some mechanism to collate and present the complete profile for an experiment is not yet available, but is under review.

3.3. Experimental Data

Data associated with experiments, including configuration information, application scripts, input data, resulting monitoring data and any other reports and output, should be stored within the relevant folders in the overall structure (see Figure 3).

Data can be retrieved from a given folder in the following ways:

1. If an iRODS client or server has been installed (see Section 6 below), then the folder structure can be downloaded (using the *test_sync.php* script). The file(s) will be available locally.
2. If no iRODS client has been installed:
    a. The testbed provider can use the TCI *getdata()* call;
    b. The testbed provider can use the RESTful interface call to the TIDS.

Method 1 is preferred, since any file transfer will be done using iRODS optimization. Method 2b is the least optimal, since file transfer via the http request may be slow.

Similar methods exist to upload or return files to TEFIS:

3. If an iRODS client or server has been installed, then once any data within the folder structure has been updated, the folder structure can be uploaded to TEFIS (using the *test_sync.php* script).
4. If no iRODS has been installed:
    a. The testbed provider can use the TCI *putdata()* call;
    b. The testbed provider can use the RESTful interface call to the TIDS.

The same observations apply: method 3 is preferred.

3.4. Experimental Metadata

Experimental metadata (see Section 5.3 below) are <name> ~ <value> pairs associated with the data objects within the iRODS system, and stored and referenced in the iCAT. The Data Services RESTful interface supports metadata querying and update. The iRODS interfaces also support metadata interactions. As discussed in more detail below (Section 5.3), the testbed provider should update and add metadata to support experiment provenance tracking and curation.

4. The TEFIS Data Service Architecture overview

The TEFIS Data Services are deployed and accessed locally within the TEFIS platform as indicated in Figure 2 below. The RPRS provides support for the other TEFIS components, including the portal and therefore the experimenters.

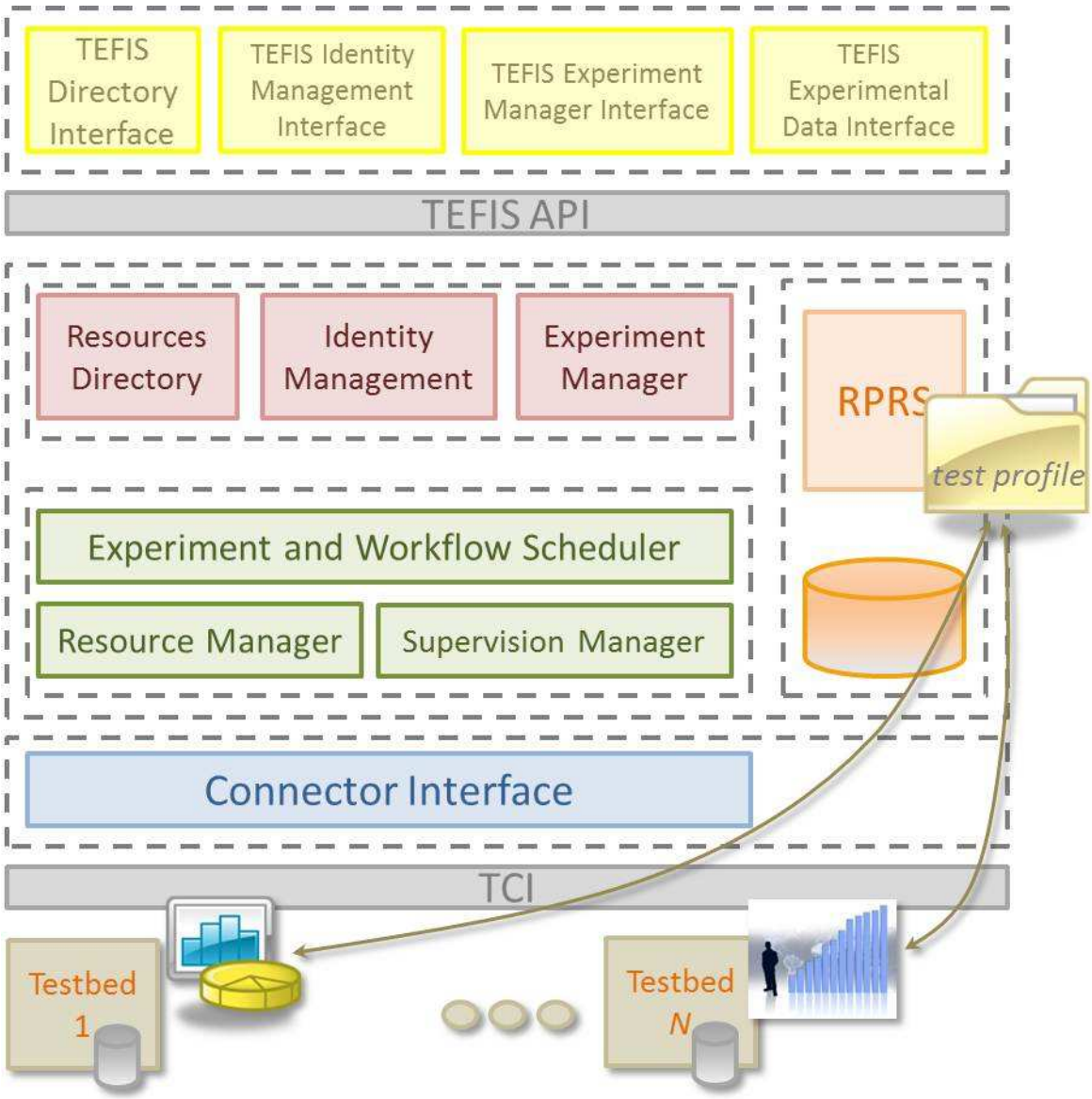

Figure 2: The TEFIS Component Architecture

Interactions from the testbed providers will come via the TIDS to the RPRS. In practice, if no iRODS components are installed by the testbed provider, then all such interaction is via the TCI and the Connector Interface. Note that, as anticipated in the section 3.1 above, typically the RPRS and TEFIS would not store large amounts of experimental data locally on the TEFIS platform itself. Instead, the RPRS would retain a pointer or link to remotely held data. The RPRS/TIDS would manage any requests to transfer the data or view it as described in [7].

5. RPRS/TIDS

5.1. Summary approach

This section introduces the overall approach for TEFIS Data Services, including the filesystem generated as part of the development and management of experiments as well as the installation options for the TIDS components.

5.1.1. RPRS/TIDS Filesystem

Since the TIDS and the RPRS respectively provide the external and the internal view onto the TEFIS data services, they share a data schema for experimental data. A well-defined folder structure, common to both TIDS and RPRS services, represents the shared data schema for experimental data, that broadly follows the Test Profile mentioned above. The whole structure is reported below, in Figure 3, while a detailed description of each item is reported in [4].

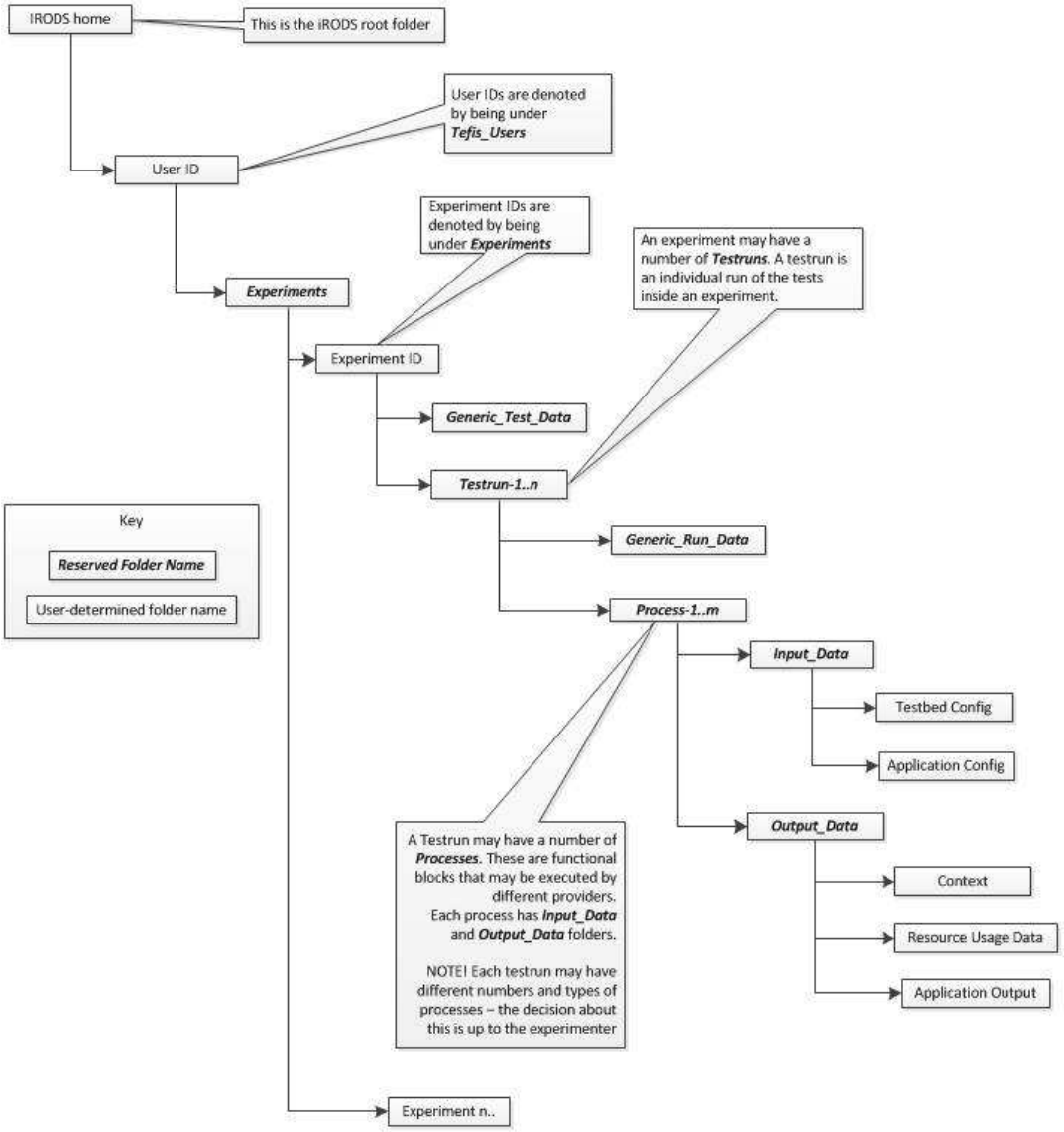

Figure 3: The TEFIS Data Services Folder Structure

During the creation of a TEFIS user (experimenter) account and the definition of their experiments and testruns, the folder structure set out in Figure 3 is generated. Although there is some flexibility, and the RESTful interface can be used by any party with appropriate access to interact with the filesystem, many of the folders and the overall structure are mandatory; other TEFIS components depend on this structure, and assume data items can be located accordingly.

From a testbed perspective, the main points of interest will include the *Input_Data* and *Output_Data* folders of a given *Process*, or workflow execution step. These folders are intended to hold the data and

information necessary to run a given step and the location to which output should be directed, respectively. Access to these areas, and other data interactions, are discussed in more detail in Section 5.2 below.

5.1.2. Architectural Options

This section is drawn from TEFIS deliverable [6], and reproduced here for convenience.

Since TEFIS provides a front-end for users (experimenters) to access and exploit remote testbeds, it is worth considering what the implementation options are with respect to the TEFIS data services. This will affect what the testbed provider can and cannot do in respect of data. In this section, therefore, we consider a number of different data service deployment options, their advantages and disadvantages and their relationship to the proposed RESTful interface[1] to the data services. The use cases for the testbed providers (use cases 1 to 3) also affect the Connectors Interface as well as the options available for monitoring in that for each component local installation on the testbed facility *versus* remote connection and operation will have significant implications.

The initial use cases may be differentiated by how much software a TEFIS testbed provider is prepared to install. They are:

1. The testbed provider can install a full TIDS (the iRODS server and an iRODS client);
2. The testbed provider can install a partial TIDS (the iRODS client only); and
3. The testbed provider will not or is unable to install any software.

There are a number of corresponding experimenter use cases, determined by how much software the TEFIS user is prepared to install. They are:

4. The user can install a partial TIDS (the iRODS client); and
5. The user will not or cannot install any software.

Each use case has advantages (usually to do with how much functionality and automation they get when they install software) and disadvantages (usually to do with loss of functionality or reliance on third parties).

In all cases, the TEFIS platform is assumed to be hosting the RPRS (with the iCAT) and a data server (an iRODS server). All remote data, either at an experimenter or testbed site can and should be synchronised to the RPRS.

1. *The testbed provider has a full TIDS instance installed, including both an iRODS client and server*

   Figure 4 shows the case where the testbed provider installs and runs a complete TIDS instance, including the relevant client and server parts of iRODS. Here the provider has maximum functionality. Since the testbed provider is acting as an iRODS server, they can store data locally,

[1] The RESTful interface to the data services is described in [6].

and allow it to be accessed by other TEFIS users. With an iRODS client installed, they can also synchronise many files to their own local server, or to the TEFIS master server, assuming appropriate access permissions are available.

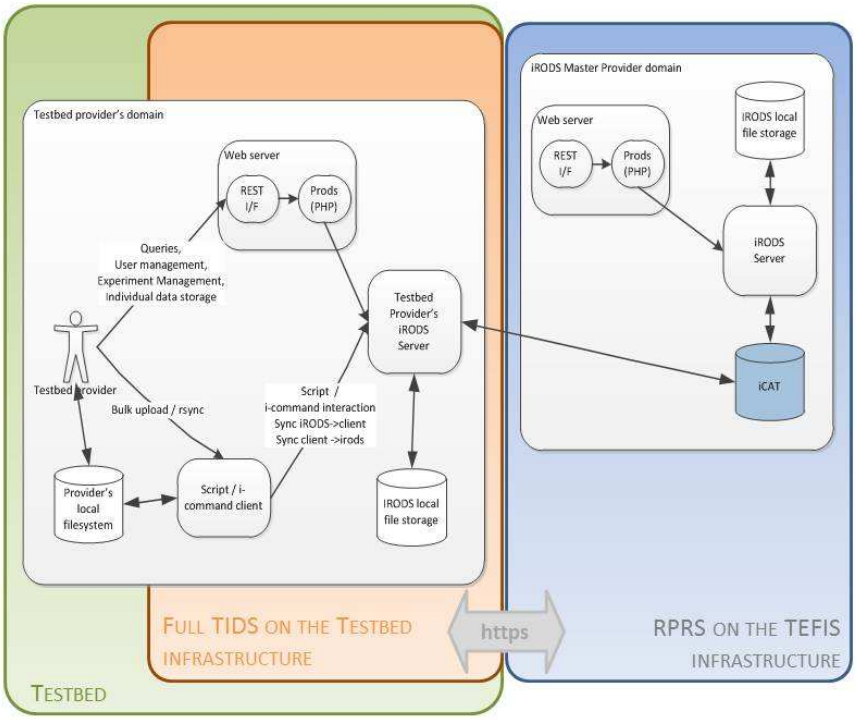

Figure 4: Testbed installs a full TIDS within their infrastructure

The advantages of this approach include:

- The testbed provider has a good level of functionality: they can synchronise large numbers of files with the iRODS server, and they can use the REST interface for management and individual file transmission.
- Large datasets can be stored at the testbed's local server and summary data can be stored at the TEFIS TIDS (the iRODS master server), reducing the data transmission costs.
- Users can interact directly with the REST interface or the iRODS server on this testbed (assuming they have the correct client software and are authorised).

The disadvantages include:

- The testbed provider has to install and maintain both the client and server-side software for iRODS.
- The testbed provider will be storing data, possibly long-term, so there will need to be a mechanism to make this storage reliable, so that future users can get access to the data. This

may be a business for the testbed provider, i.e. they get paid to store the data for a specified or indefinite time.

2. *The testbed provider installs a partial TIDS (including an iRODS client only)*

   Figure 5 illustrates the case where the testbed provider can or will only install the TIDS client on their facility. In this case, the testbed provider uses the iRODS client to access an iRODS server at the TEFIS iRODS Master, the RPRS and TIDS instances installed within the TEFIS infrastructure.

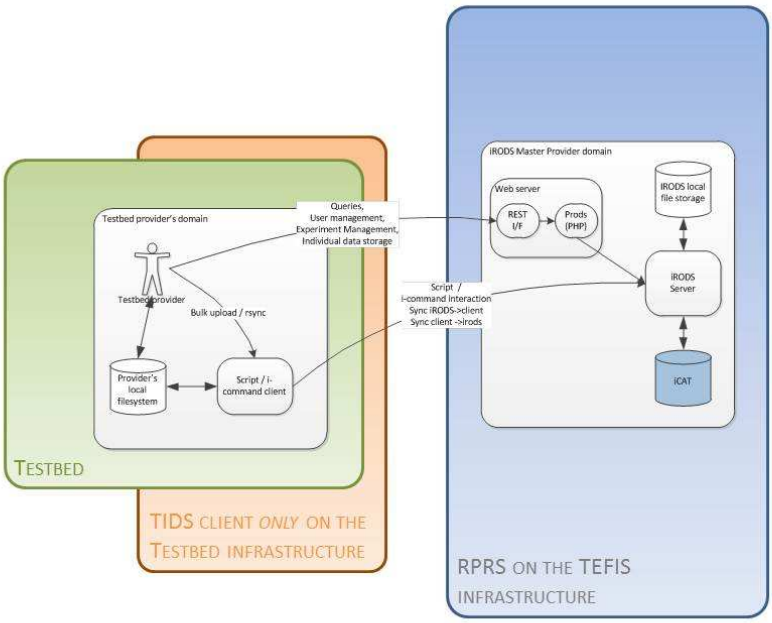

Figure 5: Testbed installs partial TIDS only

Advantages include:

- There is less software for the testbed provider to install and maintain than in case 1.
- The testbed provider has a reasonable level of functionality, in that they can synchronise large numbers of files with the iRODS server, and they can use the REST interface for management and individual file transmission.
- The testbed provider does not have to be relied upon to store any experimental data long-term – all experimental data is stored at the TEFIS TIDS instance.

Disadvantages include:

- There may be significant data transmission costs because all relevant data needs to be transmitted to the TEFIS TIDS instance.

- In addition, this will imply larger data storage requirements on the TEFIS TIDS instance than for case 1.

- All data is stored in one place, the TEFIS TIDS instance, so there is a need to backup this data or employ data replication to ensure recovery in case of server or disk failure.

3. *The testbed provider will or cannot install software at their facility*

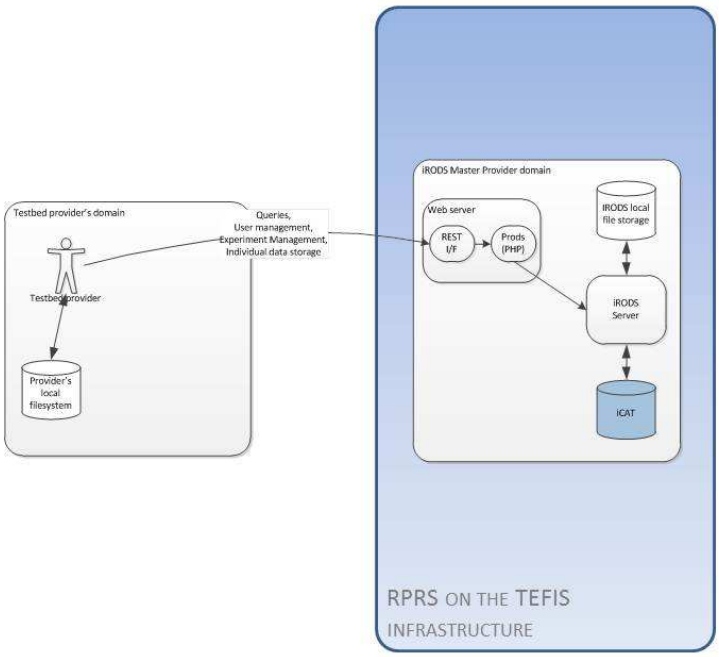

Figure 6: No software installed at the testbed facility

Figure 6 illustrates the case where no additional software can or will be installed at the testbed provider's site. The only interface for data services would be the RESTful interface, either directly (the testbed provider communicating with the interface from their own code) or indirectly via the TEFIS portal.

Advantages include:

- No software to install on the testbed provider

Disadvantages include:

- Functionality is severely limited.

- The testbed provider must use REST calls to upload and download data, or retain indefinitely at the local site.

- At the present time, this means that files must be uploaded / downloaded individually.

For the experimenter, the following cases apply.

4. *User installs a partial TIDS (including an iRODS client) at their site*

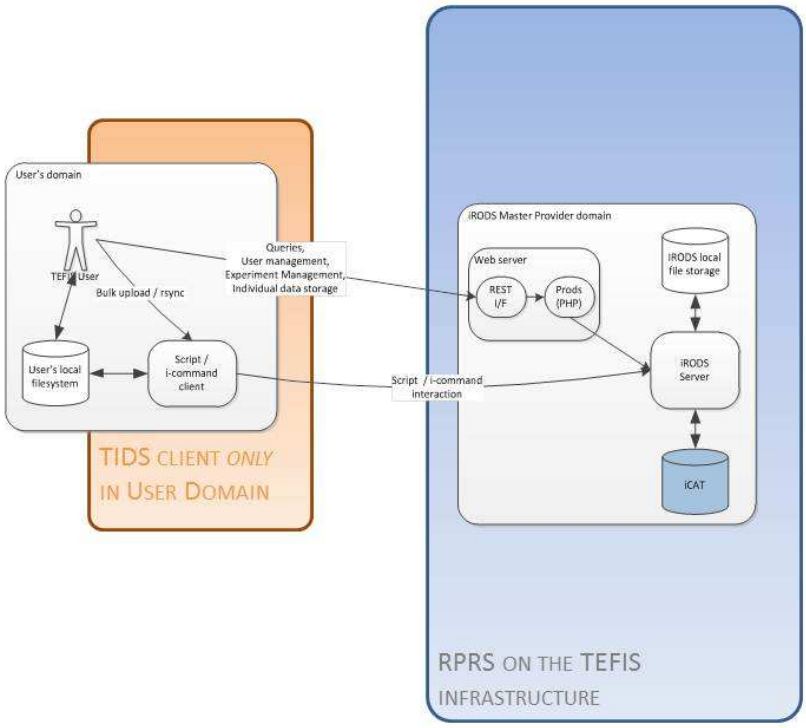

Figure 7: Partial TIDS client at experimenter's location

If the experimenter is prepared to install a partial client (one of the iRODS clients, for instance) at their own location is analogous to testbed provider case 2 (see Figure 7). As such the same advantages and disadvantages obtain: the experimenter has less software to install and maintain, but the function available is limited.

5. *The experiment can or will not install software locally.*

The final use case (see Figure 8) again is analogous to one of the testbed provider use cases (case 3).

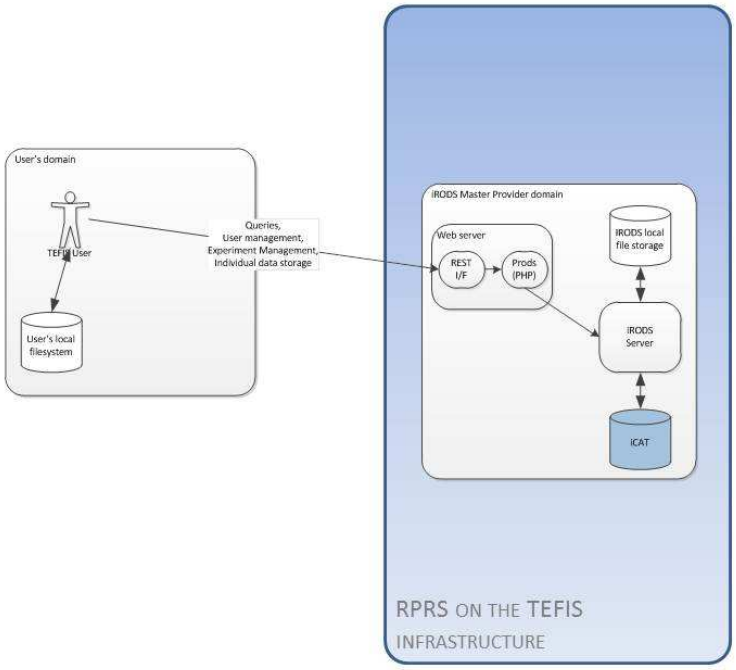

Figure 8: No additional software is installed at the experimenter's location

With no software to install at their own site, this is the simplest case for the experimenter. However, the downside is that they can only interact via the RESTful interface. This could be directly, if they are prepared to generate appropriate interface code; or indirectly via the TEFIS portal Experimental Data Interface. Either way, managing data in terms of moving large amounts of data between locations is less convenient and has to be on a file by file basis.

5.2. Testbed Interactions

Section 3.3 sets out the main options for data retrieval from and upload to TEFIS: the testbed provider may install appropriate iRODS support to be able to synchronise data using the full benefits of the iRODS system, use the TCI interface or even RESTful calls. Each of these options has its advantages and disadvantages. It should be noted, though, that there is a practical expectation that testbed providers will have to retrieve data and write data repeatedly to the experiment folder structure. Ultimately, the method used by the testbed provider to interact with TEFIS and the Data Services (see Figure 9) will come down to practicalities of efficient data transfer.

The options are these:

1. The testbed installs iRODS components. Data transfer is via iRODS-support synchronization. This is the most efficient method;

2. The testbed installs nothing. Data transfer is either via the TCI *getdata()* and *putdata()* calls or via the RESTful interface provided by the RPRS/TIDS; or

3. Tests could be deployed within a VM that includes iRODS and (temporary) data storage. The user would need to be alerted before the VM was destroyed to ensure all data had been transferred.

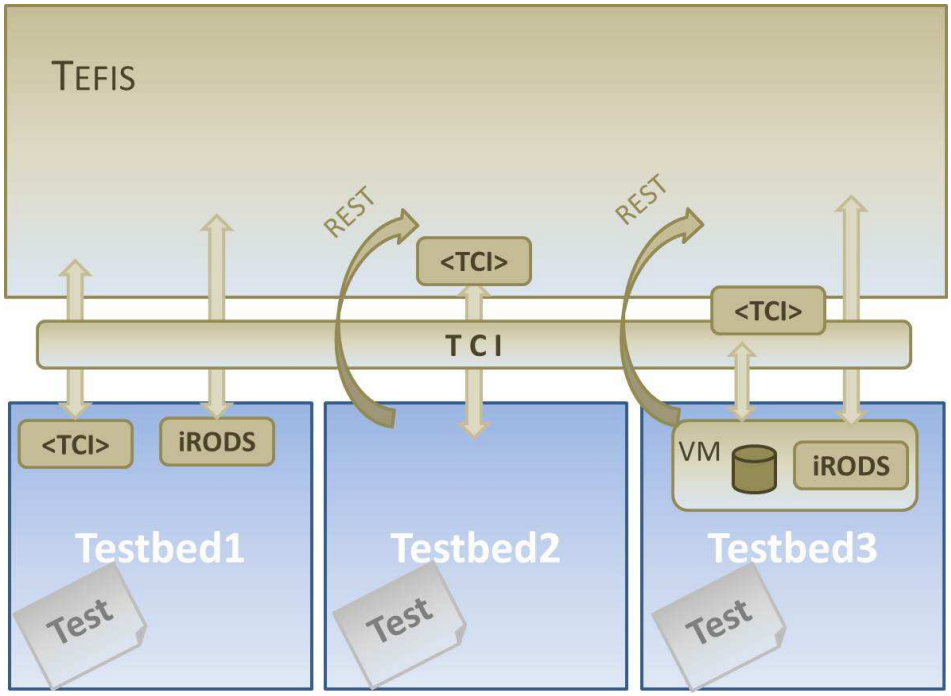

Figure 9: Interaction modes

5.3. Experimental Metadata

Metadata are used to identify various characteristics of the experiment and/or testrun for a given experimenter. The metadata are stored within the iRODS iCAT to enable searching and other classification. Much of the metadata will be supplied when the experiment is defined. However, some metadata can only effectively be defined and made available from the testbed. The metadata can be attached specifically to a resource via the appropriate treatment call on the RESTful interface (see [6], see also [8]). It is the responsibility of the individual testbeds to provide the experimental metadata associated with the workflow steps (the *Processes*). For instance, when a step has been executed, then the testbed would signal completion with a timestamp added to the *Process* folder. Collating all of the appropriate information into a single *test profile* will require some additional work, and is not yet supported[2].

Figure 10 shows the initial set of metadata that would be expected to be provided with a given experiment. These relate mainly to the data associated with running the experiment and its results. However, we also need to maintain information about *provenance* – the history of an experiment, with a view to being able to reproduce it – and *curation* – what should be done with the experiment and its associated data. These two areas are introduced in the next two sections.

[2] We are currently investigating generating an event log from iRODS based on metadata which may serve as an interim *test profile*.

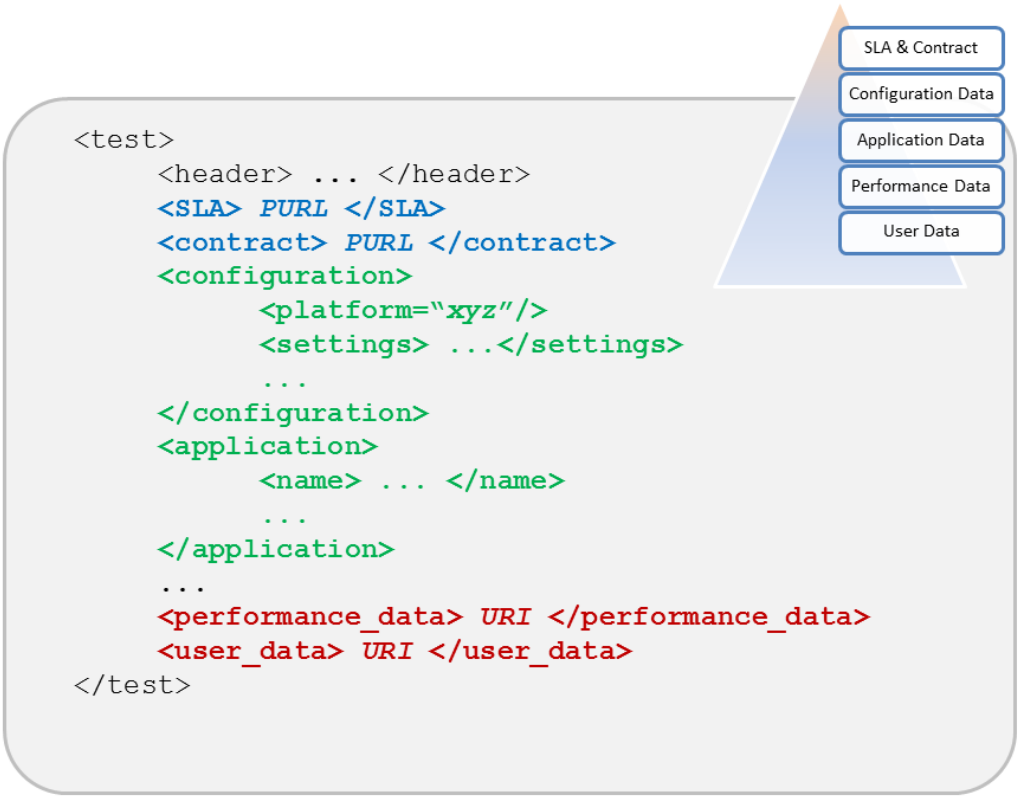

Figure 10: Metadata schema for the *Test Profile [5]*

5.3.1. Provenance

As an initial attempt to provide a suitable provenance trail, the following metadata should be provided. Ideally some of these would be updated by each testbed on completion of their individual process or processes, and propagated up to the *Generic_Run_Data* folder of the specific testrun.

Metadata Item	Recorded Where	Description
timestamp	On individual *Process* folder, propagated up to *Generic_Run_Data* folder for this testrun.	Identifies when a process completed or returned a result or error. It may be appropriate to split start and end times, especially for error cases.
testbed	On *Process* folder	Identifies the testbed where the process step was executed.
requester	On *Generic_Run_Data* folder	Identifies the experimenter OR the TEFIS component that initiated the run or request.
configuration_data		Identifies the configuration or application

Metadata Item	Recorded Where	Description
application_data		data settings, for instance via the URI of a configuration file, application modules and/or run scripts. Curation settings need to be used to decide whether or not the actual files and values should be retained.
results	On *Process* folder. Propagated up to the *Generic_Run_Data* folder.	Identifies where any data resulting from the specific execution step are located.
status_on_completion	On *Process* folder; this would then be used to derive a status indicator for the whole experiment.	Identifies whether the step completed, and whether it completed with or without errors. The error code, if any, would need to be retained somewhere.
iteration_count	On *Process* folder; possibly on the *Generic_Run_Data* folder as well.	Identifies how many times an execution step or testrun has been run.
precedent	On *Generic_Test_Data*	This would be used to link to any other experiments that were consulted or direct antecedents of the current experiment, or any experiments which the user flags as similar to their own. This may also be used for curation.

The table here summarises a list of metadata "names" that should be used to provide some indication of the provenance for a given experiment. Either directly or after some collation and processing, they will be stored in the *test profile*. Currently, there is no explicit mechanism to collect lessons learnt during a description. This would typically be stored in the *Generic_Test_Data* if it applied to the entire experiment or the *Generic_Run_Data* folder for an individual testrun. We should in future consider how to collect, store and then exploit such experience. For example, this may become part of the experiment search capabilities of TEFIS.

5.3.2. Curation

In addition to the metadata associated with the history of the experimental lifecycle, we should consider curation. Although this may in the future lead to the generation of additional information, such as the number of times an experiment was queried, how many other experimenters have created experiments directly based on it, for now we need to mark the following:

Metadata Item	Recorded Where	Description
expiry_date	On individual resources	Identifies when a resource should be removed
time_to_go		Identifies the difference between the current time and *expiry_date*
owner		Identifies the experimenter OR the TEFIS component that owns this resource
read_access		Identifies who may look at the resource
write_access		Identifies who may update the resource

At this stage, this is a preliminary set of possible metadata designators to be extended and exploited for the purpose of curation.

6. TIDS Installation

The TIDS, like the RPRS, has been built on the iRODS data system [1]. Installation is much the same as detailed in [4], and reproduced here for simplicity. Note that both the TIDS and RPRS are currently deployed in a VMware® virtual machine.

6.1. Client side

6.1.1. Requirements

Extract the PHP executables from

```
http://windows.php.net/download/
```

Use thread safe executables[3].

6.1.2. Contents of client package

The client package contains two main elements:

- The iRODS "`i-commands`". These are command line progams that interact with iRODS servers and perform file, metadata and administration operations. The i-commands have similar functionality to their unix namesakes – for example the i-command "`ils`" is an iRODS version of the unix "`ls`". The `i-commands` are described in detail in [3].

- A PHP script that customises the i-commands for TEFIS use called "`test_sync.php`". This performs data upload and download from a local file system to an iRODS server, either in bulk or singly.

The `i-commands` are discussed adequately in [3], as they are direct from iRODS, but the PHP script needs further explanation, and is discussed below.

6.1.1. Client Setup

Unzip the package into a folder and put the folder into your path. The folder contains the iRODS so-called "`i-commands`" and some PHP scripts that customise the `i-commands` for TEFIS use.

To be able to connect to iRODS, you will need to set up the client environment. This is described in [2]. It involves creating a folder named "`.irods`" in your home folder and inside this creating a file named "`.irodsEnv`".

Inside the "`~/.irods/.irodsEnv`" file you need to set some variables, as follows. For the default installation, you will probably only have to change the "`irodsHost`". In the future, it is likely that each iRODS user will get their own user account, so you will need to set these correctly as well.

Copy the following into the `.irodsEnv` file and save it.

```
# iRODS personal configuration file.
```

[3] If using Apache, then download VC6; if using IIS, then download VC9.

```
#
# iRODS server host name:
irodsHost TEFIS-Test1

# iRODS server port number:
irodsPort 1247

# Default storage resource name:
irodsDefResource 'demoResc'

# Home directory in iRODS:
irodsHome '/TEFISTest/home/TEFISUser1'

# Current directory in iRODS:
irodsCwd '/TEFISTest/home/TEFISUser1'

# Account name:
irodsUserName 'TEFISUser1'

# Zone:
irodsZone 'TEFISTest'
```

You can test you have the correct environment by using the i-command "`ienv`". This shows the environment you have set up.

If you try to connect to iRODS using any of the i-commands, you will be asked for the user password each time. You can set the password once, so you don't have to keep on typing it. This is done using the i-command "`iinit`", which asks for the user password (for the "`TEFISUser1`" user, the password is the same: "`TEFISUser1`"). The "`iinit`" command stores this password

You can test the connection to the iRODS server by using "`ils`" or "`ipwd`". These are i-command analogues of the unix commands "`ls`" and "`pwd`". If you get no errors, you should be connected to iRODS.

Test Data Synchronisation Script – Test_Sync.php

This scrpt addresses upload and download of data from an iRODS server. It takes the form of an "iRODS rsync", which synchronises between iRODS and a local copy of the data. The help is as follows.

```
This script synchronises TEFIS test data from a local version to the iRODS server.

You will need the iRODS i-Commands installed and their location in your path.

You will also need to be able to connect to the TEFIS iRODS server.

Usage:
test_sync [Up/Down] [TestOwner] [TestID] [?ProcessID] [?Local Folder]

        Up/Down tells the direction of the synchronisation
             Up = sync this local copy to the remote server
             Down = sync the remote version to this end

                (if this end is empty, the script will create it)

        Test Owner is the user ID of the owner of the test.

        Test ID is the identifier of the test.

        [optional] Process ID is the ID of the process you want to synchronise. If left out it
        will synch all Processes.

        [optional] Local Folder is the name of the folder you wish to synchronise to. This arg is
        optional - if not specified as an argument the local folder will be a concatenation of the
        test owner, ID and process ID under the folder you ran this command.
```

```
!!NOTE:

       If any argument has spaces, it must be enclosed with double quotes (""). For example the
       experiment ID t60 v2 must be specified as an argument as "t60 v2". If the local folder is
       auto-generated, an underscore (_) will replace the space.

With the --help, -help, -h, or -? options, you can get this help.
```

The PHP script is launched with a batch or shell script that launches PHP and the script. The batch or shell script will need to be edited to point to the correct paths of the PHP runtime executable and the script to be executed.

For example the batch script "test_sync.bat" contains:

```
"<PATH-TO-PHP>\php.exe"

"<PATH-TO-TEST_SYNC.PHP>\test_sync.php" %*
```

You will need to replace <PATH-TO-PHP> and <PATH-TO-TEST_SYNC.PHP> with the paths relevant to your particular client installation.

6.2. Server side

Requirements

VMware® Player & any installation of VirtualBox should be uninstalled.

Other VMware hosts may work but have not been tested.

6.2.1. Server – side package

The server is a VMware virtual machine. Inside this is a Ubuntu 10 distribution with iRODS and a LAMP (linux Apache Mysql PHP) server already configured upon it. The LAMP server is to host the web page that lets users create user accounts and experiments, and search for other experiments.

The Ubuntu installation has two users:

- "`irods`" with the password "`irods`". This is the main user, who controls the iRODS server.
- "`TEFIS`" with the password "`TEFIS`". This is the super user.

6.2.2. Administration

Login as the "`irods`" user. The home folder is `/home/irods/` and iRODS is installed in a subfolder named "`iRODS`". The path has already been configured so that `/home/irods/iRODS` is included. To start iRODS, execute the command:

```
irodsctl start
```

as the user "`irods`".

To stop the iRODS server execute the command

```
irodsctl stop
```

as the same user.

The other thing that needs to be started is the LAMP server (this is XAMPP). To start the XAMPP server:

1. Change to the super user "`TEFIS`"
2. Go to `/opt/lampp`
3. Execute `sudo ./lampp start`
4. You will be asked for the super user password for the "`TEFIS`" user.

To stop the XAMPP server, the process is exactly the same, except that the command is:

```
sudo ./lampp stop
```

in the same folder.

The PHP scripts and web pages used to make up the web site are in the "`htdocs/irods`" folder under "`/opt/lampp`".

To test the lampp server open a web browser and connect to:

http://localhost/irods/

On a remote machine (assuming bridged networking is working) connect to:

http://{TEFIS-name}/irods/

and you should see a basic web page for the TEFIS administration.

6.2.3. How to use server-side

Go to http://{TEFIS-name}/irods and use the items on the page to:

- Create a TEFIS user
- Create a TEFIS experiment
- Search existing tests

References

[1] https://www.irods.org

[2] https://www.irods.org/index.php/user_environment

[3] https://www.irods.org/index.php/icommands

[4] TEFIS Deliverable D6_2_1 RPRS Prototype

[5] TEFIS Deliverable D6_1_1 Specifications for Experimental Metadata

[6] TEFIS Deliverable D3.2 Building blocks integrated into the TEFIS Portal

[7] TEFIS Deliverable D3.3 User Tools Implementation

[8] TEFIS internal document : TEFIS_Data Services_REST0.1