Deliverable Title	**D6.2.1 Repository service implementation**
Deliverable Lead:	IT INNOVATION
Related Work package:	WP6
Author(s):	Steve Taylor, Brian Pickering
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 summarizes the D6.2.1 Prototype deliverable: the RPRS, or local data services, for the TEFIS platform. It describes the general concepts, structures and how to get started using the iRODS data management system.

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)
0.2	30/04/2011	First Draft	Brian Pickering (ITI)
0.3	03/05/2011	Incorporating SJT Comments	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

IIS	*Internet Information Services*
iRODS	*integrated Rule Oriented Data System*
RPRS	*Research Platform Repository Service*
TCI	*TEFIS Connectors Interface*
TIDS	*Testbed Infrastructure Data Service*

1. Executive Summary

This document provides general information and guidance about the TEFIS Data Services: the Research Platform Repository Service (RPRS). The RPRS and associated Testbed Infrastructure Data Service (TIDS) provide other TEFIS components and experimenters with the management and infrastructure services to store and maintain any and all data directly associated with the experiments run via the TEFIS facility; it does not hold resource information for the configuration of the testbed facilities to be used during such experiments (see the TEFIS Resource Directory component). Both the RPRS and TIDS are based on the iRODS data services, which is widely adopted for many large scale repository services.

The document is intended for TEFIS platform users who need to understand how the data services are configured. It is structured as follows.

- The *General Concepts* section introduces the main ideas behind the RPRS design;
- *Architectural Options* summarises the different ways that TEFIS end-users (both experimenters and testbed providers) can install the TEFIS data service components;
- *RPRS Installation* provides specific guidance on how to get started with the RPRS, including:
    - Instructions for the *iRODS Installation*;
    - The *RPRS Filesystem* which summarises the structure which the RPRS imposes for the management of experiments;
    - Finally, *Interacting with the RPRS* describes how to retrieve from and submit to the RPRS any experimental data
        - As an experimenter, or
        - Another TEFIS component

The document should be read in conjunction with [5], which describes the TIDS, and [6], which provides the background for all data treatment in TEFIS.

2. Introduction

The Data Services are central to all the processing on the TEFIS platform: each component must interrogate existing data and generate new data as the experiment moves through its lifecycle. Data must be stored locally within the TEFIS platform itself during the planning of a given experiment. In addition, during the execution of a given experimental run, data must be shared with the target testbed resources and then returned to the experimenter environment for review and analysis. Within the TEFIS environment itself, these data need to be stored within a recognisable, common folder structure, enhanced with appropriate metadata which describe aspects of provenance and the experiment lifecycle.

The Data Services for an entire experiment are distributed across two main environments:

1. The **Research Platform Repository Service** (RPRS) provides a repository service to the TEFIS platform and the TEFIS middleware components. As an experiment is generated, the Experiment Manager initiates the creation of the appropriate folder structure for this user and for this particular experiment. Within the experiment, individual runs (instances of an experiment) use separate folder structures to store data appropriate to that specific run, such as configuration and application data.

2. The **Testbed Infrastructure Data Service** (TIDS) provides the data management services associated with the TEFIS repository within the remote, testbed environment. The TIDS supports the exchange of data between TEFIS and the testbed, via the Connector interface.

In this document, the RPRS (the data repository service for the TEFIS platform itself) is described, how it has been implemented and the associated processes, interacting with other TEFIS components during experiment set up and execution.

3. General Concepts

The RPRS is built on iRODS ([1]) which provides a virtualised filesystem and the associated tools to be able to locate and search any data within the filesystem based on the metadata assigned to the data objects. The general approach to metadata associated with experiments is covered in [6]; below and in [5], we discuss specific metadata handling as appropriate.

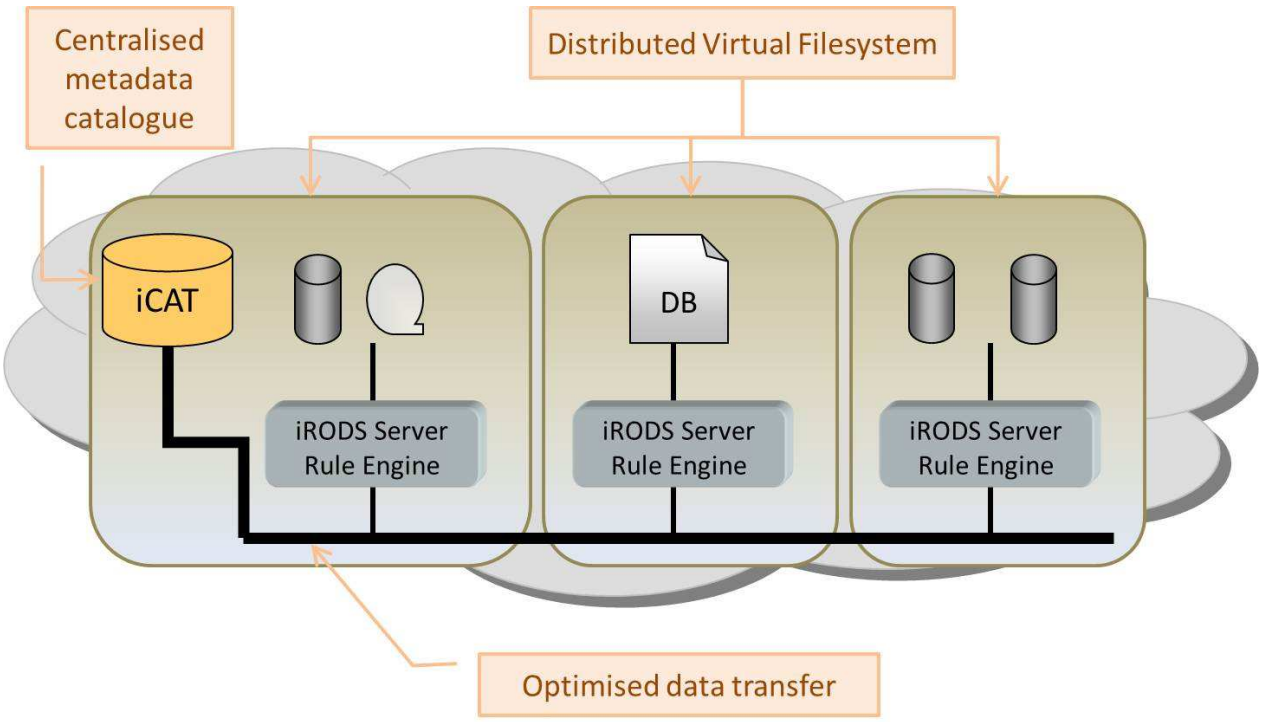

Figure 1 : The iRODS Architecture

The basic iRODS architecture is shown in Figure 1 above. There is one master catalogue database, iCAT, and many (at least one) servers. Users connect (via a client) to a server to upload, search and download. When a request is made by a client to a server, the server connects to the master catalogue to get metadata and information about other servers and the data they hold. The data is collected from the server holding the data and served back to the user via an optimised extension to the IP protocol for the transfer of extremely large data objects along parallel, packetised streams.

3.1. RPRS Terms

Although covered in other related documents (see, for instance, [6]), the table below summarises the main RPRS-specific terms used in this document.

Term	Description
Experiment	A general class of a test: the overall description of what the *experimenter* wants to run.
Testrun	An individual instance of an *experiment* workflow.

Term	Description
Process	An individual *workflow* step, taking input and producing output. *Processes* may be run on the same testbed or on several different testbeds as defined in the experiment workplan. A process is a component of a workflow – it is an atomic (from the point of view of the enacting component) task that may be executed at a remote site (for example a testbed). Workflows are compositions of tasks.
Experimenter	The TEFIS end-user requesting to run a specific test or set of tests.
Input data	Any data required to run an entire *experiment* (in which case it is held in the *Generic_Test_Data* folder), for a complete *testrun* (in which case it is held in the *Generic_Run_Data* folder) or for a specific *process* (in which case it is held in the *Input_Data* folder for that process).
Output data	Any data generated as a result of the execution of a specific *process* or a complete *testrun.* For individual workflow steps, or *processes*, such data would typically be stored in the *Output_Data* folder for that process. For complete testruns, there is currently no specific directory, other than the *Generic_Run_Data* for this purpose, since typically large data sets, such as *Monitoring* or *Application data* would typically remain on the testbed until processes by the experimenter.
Monitoring data	Data generated during a testrun or *process* relating to the performance of the platform. These data are handled by the *Supervision Module* (q.v.); they may be referenced via the TIDS as metadata associated with experiment provenance (see [5]).
Application data	Data generated during a testrun or *process* relating to the output from a specific application under test, such as a user dialogue for the Multimedia test scenario, or log data from a suitably instrumented script for the eTravel case. These data would typically reside on the testbed facility until processed by the *experimenter*.
Workflow	A workflow may be a template or an instance. A workflow template is a “program” made by connecting processes together to perform some desired function. A workflow instance is an individual execution of a workflow template, so there may be many instances of a workflow template. A workflow instance could be also called a

Term	Description
	"run" – so far there is no reason preventing "workflow instance" and "run" to be used interchangeably.

3.2. TEFIS implementation

For our implementation we currently have one iRODS master (iRODS server + iCAT)[1]. TEFIS users may be:

- *Test owners* – these are people that create tests, own and control them
- *Testbed providers* – these are people that execute component processes of tests
- other users – these are miscellaneous users that the owner of the test wants to access the test, including other TEFIS components.

[1] This is installed and running in the VM6 virtual machine of the TEFIS platform in INRIA: TEFIS6.inria.fr

An example of two of these users interacting with the iRODS master is shown in Figure 2 below.

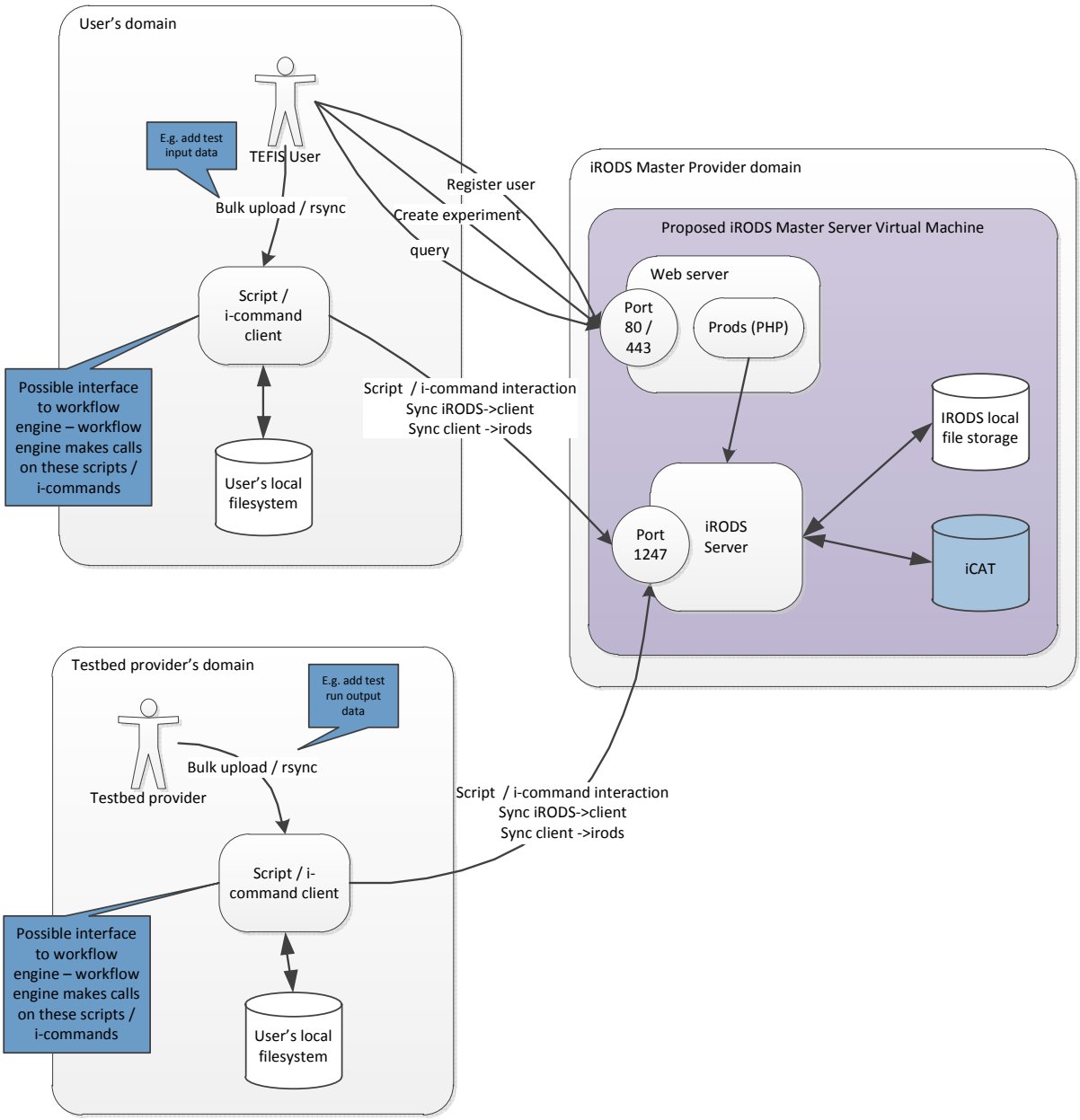

Figure 2: iRODS server and clients

The iRODS master server contains a number of elements:

- The iRODS iCAT database. This is the management database for iRODS and stores information about the data being stored, metadata, location of the data (which iRODS server it is stored at for example) and access control information.

- An iRODS server. This is the iRODS component that actually stores the data. There may be many iRODS servers at different locations connected to the iCAT database, providing a distributed storage system. However, in our first implementation, we have one iRODS server co-located with the iCAT database. We can expand this later to include additional iRODS servers if needed.

- A basic PHP-based web interface that enables users to create user accounts, tests and search for tests[2].

On the client side, there are tools to enable the user to upload and download data to and from the servers. To upload and download data to the iRODS servers, the users must have the client package installed. This lets them synchronise local folders with tests or processes on the server. Other installations options are described in the next section.

3.3. Typical Usage Cycle

The basic pattern of operation is the following:

1. An experimenter creates a user account using the web interface[2] at the iRODS master server.

2. The experimenter creates an experiment. They specify the name of the experiment and metadata describing it[3]. The system creates a defined folder structure on the iRODS server for that test (see Figure 3).

3. The user "down synchronises"[4],[5] the test with a local folder onto their own machine. This creates the same folder structure on their machine as on the server.

4. The user can put input or other data into their local copy and can "up synchronise"[4,5] their local copy to the server. This copies all the files from the local machine to the server.

5. The owner of an experiment can give control of one or more processes to a testbed provider so they can do their work as part of the testrun[6].

    a. The testbed provider can "down synchronise" to get a local copy of the process in the same way as the experiment owner.

    b. The testbed provider can run the test and put whatever data they want into their local copy. The data the testbed provider most likely creates is output data from a particular process.

    c. The testbed provider can then "up synchronise" to copy their local data to the iRODS server

    d. The owner of the test can then "down synchronise" to get the testbed provider's output data.

    e. This can happen for each process, which may occur at different testbed providers.

[2] This is in the process of being migrated to the Experimental Data interface.
[3] The individual testruns and processes need to be created separately.
[4] The "down synchronise" and "up synchronise" operations are done using the client-side PHP script "test_sync.php", described in the section on client-side software.
[5] This assumes the user has installed some of the iRODS components (ie. a partial TIDS installation as described in [5]) on their local machine.
[6] The steps here assume at least a partial TIDS installation. Data may also be retrieved from the RPRS and submitted to it via the TCI getdata() and putdata() calls.

4. Architectural Options

Unlike the TIDS (see [5]), the RPRS must be installed locally with the TEFIS platform. This includes the iCAT instance, holding all the information required to identify and locate data within the virtual filesystem.

5. RPRS Installation

The iRODS system (see [2]) is deployed as described in Section 5.1. Once installed, the data services will be ready to generate the folder structure as described in Section 5.3 below, and users as well as TEFIS components can interact as set out in Section 5.4.

5.1. iRODS Installation

The guidelines here for the installation of the iRODS system assumes that the immediate environment will be in a virtual machine.

5.1.1. Server Side

Requirements

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

Other VMware hosts may work but have not been tested.

5.1.2. 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.

5.1.3. 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.

5.1.4. 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

5.2. Client Side

5.2.1. Requirements

Extract the PHP executables from

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

Use thread safe executables[7].

5.2.2. 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 [4].

- 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 [4] as they are direct from iRODS, but the PHP script needs further explanation, and is discussed below.

5.2.1. 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 [3]. 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.
#
# 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'
```

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

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.

5.3. RPRS Filesystem[8]

The test structure follows broadly the test profile described in D6.1, and is shown below in Figure 3 below. The structure is created automatically when a TEFIS user, the experimenter, creates an experiment indirectly: the actual request to generate the appropriate structures will be the RESTful calls from the Experiment Manager interface. The structure must be adhered to.

The purpose of the structure is broadly to separate the processes that different testbed providers may execute and their input and output data. Access control will need to be applied to processes later, but the intention here is to provide a structural separation to simplify this. The restrictions imposed by the test structure are not severe, or particularly limiting, as once a user is inside a Process' "input data" or "output data" folder, they can create whatever folder structure they like. Similarly, any TEFIS component can create folders using the RESTful calls or i-commands as required.

Folder	Created when	Created for
User ID	Experimenter requests an account with the Data Services via eMail; TEFIS administrator creates the account and this directory. The *User ID* is the usre name.	May hold any data associated with the experimenter, user-generated or system generated[9].
Experiment	In response to a request from the Experiment Manager interface	May hold any data associated with the experiment, user-generated or system generated[9].
Generic_Test_Data	Created **with** Experiment	Intended to hold data relevant to the entire experiment.
Testrun	In response to a request from the Experiment Manager interface.	May hold any data associated with the testrun, user-generated or system generated[9].
Generic_Run_Data	Created **with** the *Testrun*. The number of processes must be	Intended to hold data relevant to the entire testrun.

[8] Support for a ***testrun*** in addition to the experiment itself is about to be released from the local test machine. The previous version, with Experiment referred to as Test, and no testrun substructure may still be deployed in some environments. Check that the latest version is installed.

[9] That is by one of the TEFIS components; NB that there is currently no access permission support, and so any folder can be viewed or up/down-sync'd.

Folder	Created when	Created for
Process	specified on the testrun request.	Not intended to hold any data.
Input_Data		Intended to hold any data relevant to the execution of the associated *Process*. The testbeds would read from this folder (*getdata()* call to the TCI, or via *down-sync*).
Output_Data		Intended to hold any data generated during the execution of the associated *Process*. The testbeds would write to this folder (*putdata()* call to the TCI, or via *up-sync*).

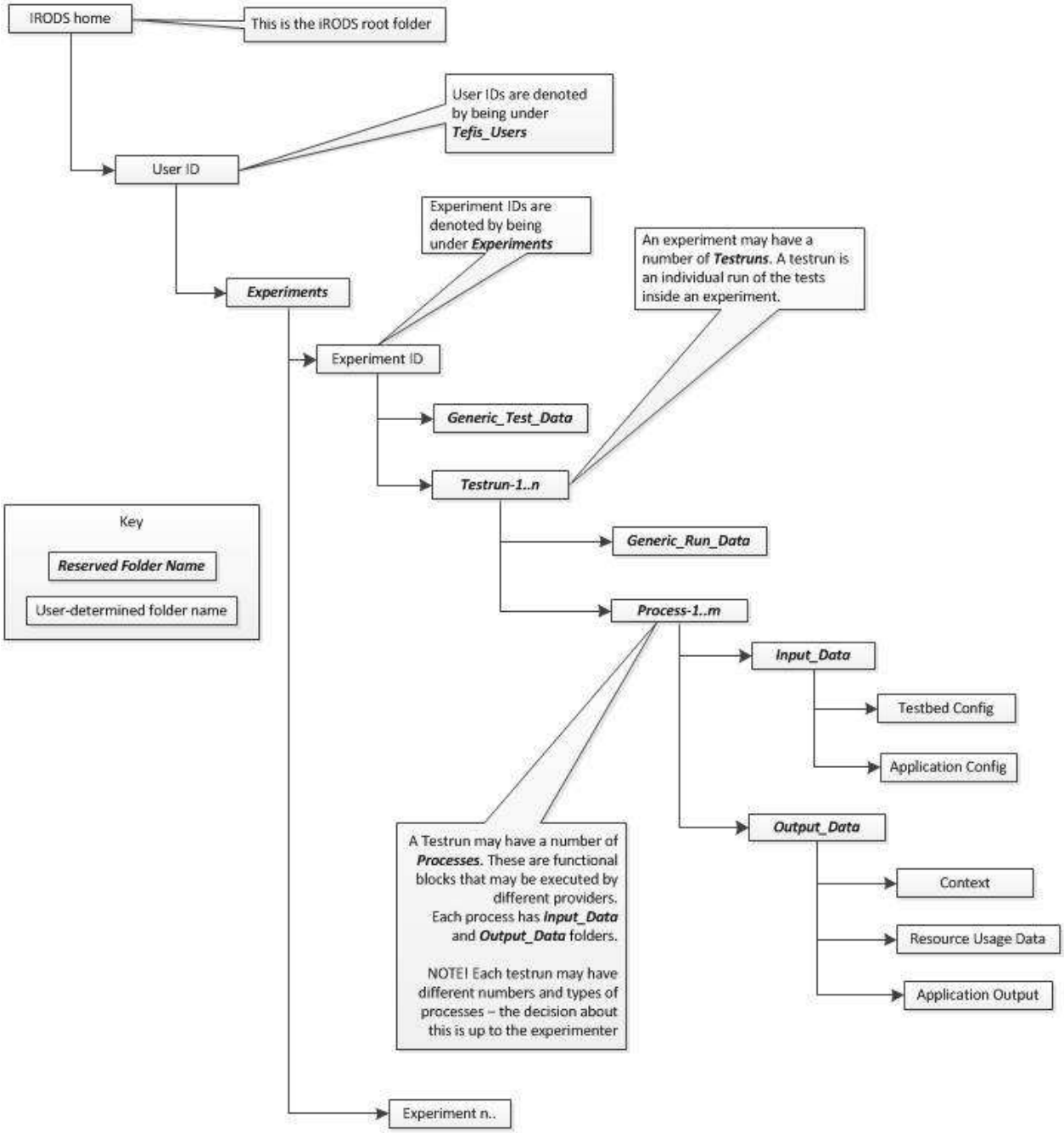

Figure 3: The RPRS Filesystem Structure

5.4. Interacting with RPRS

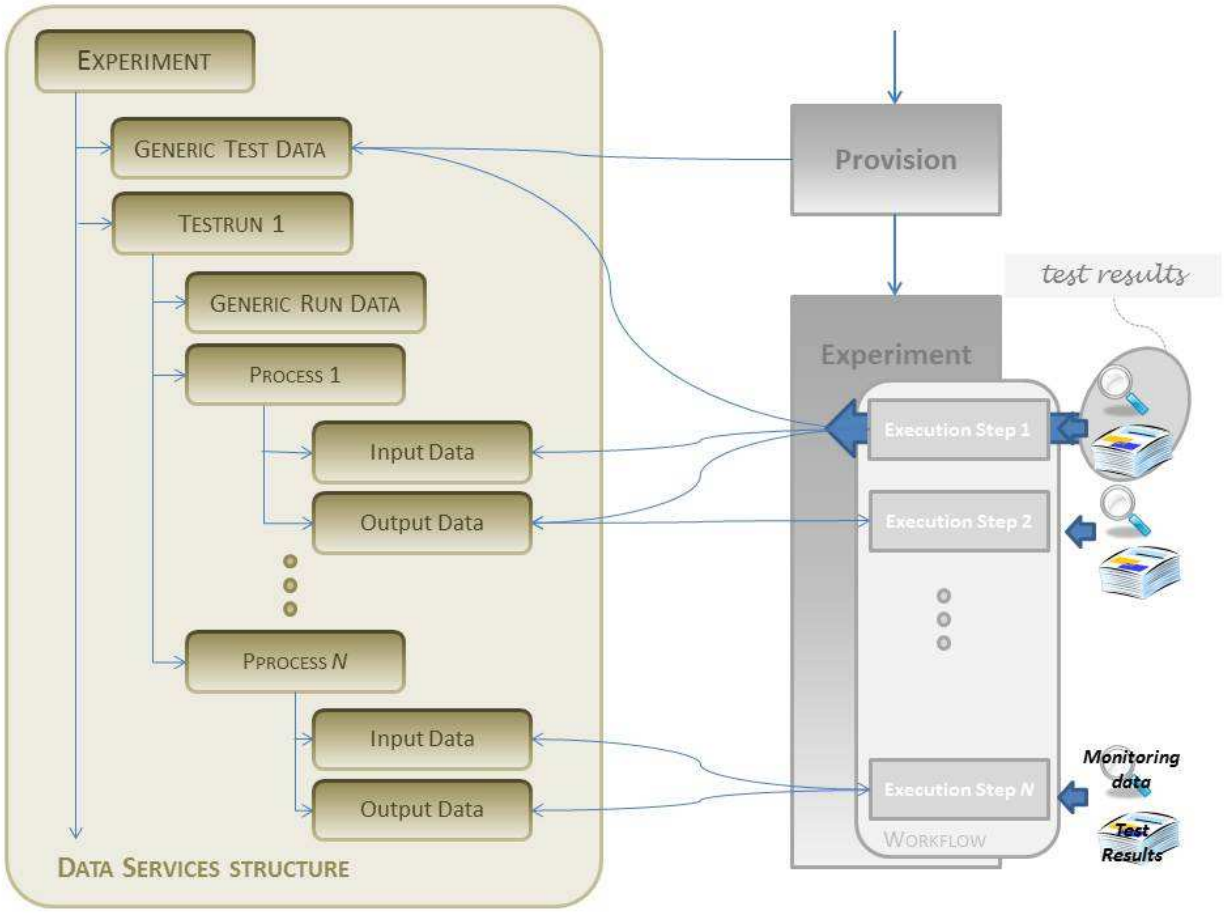

Figure 4: Overview of interactions with the RPRS Filesystem

Figure 4 summarises the intention behind the RPRS folder structure introduced above (Section 5.3). When an experiment is initially provisioned, then the associated information would typically be written to the *Generic_Test_Data* folder. From there, it may be retrieved during execution as required.

From then on, for individual execution steps, data may be retrieved from the *Input_Data* folder of the associated Process folder, and written to the *Output_Data*. Note that there is no implicit support for retrieving input data for a process step from the *Output_Data* folder of the preceding process step: any such redirection should be handled explicitly in the testplan or workflow. Output data from the execution of a Process may include monitoring and/or application or test data. These data may be uploaded directly, but should preferably be handled separately by the experimenter once they have been reviewed.

5.4.1. TEFIS Users: *Experimenters*

TEFIS experimenters interact with the RPRS via the *Experimental Data Interface,* as described in [7]. However, as stated above, there are some PHP pages to allow for:

- Creating a user account (Experimenter ID);
- Creating an experiment; and
- Searching for an experiment.

These will be replaced by the interface described (*op.cit.*) over time.

5.4.2. TEFIS Components

Other TEFIS components may interact with the RPRS via:

- The i-commands (see [4]) directly; or
- The RESTful interface, described in [8].

The TEFIS components have much flexibility to add data and folders within the RPRS system. It is important though that the overall structure described (see Figure 3) is preserved.

5.4.3. Testbeds

Interactions with the TEFIS platform data service is covered in greater detail in [5]. In summary, much will depend on how much of the TIDS has been installed on the testbed: whether there is no additional software, an iRODS client or and iRODS server. In the latter, optimal case, then the testbed would simply need to down- and up-synchronise the folder structure associated with the experiment, testrun or individual process as required, making changes in the local copy before up-synchronising it again. When there is no software installed, then data retrieval and data submission has to be effected via the TCI *getdata()* and *putdata()* calls respectively.

6. Conclusion

In the preceding sections, the RPRS as currently deployed in VM6 in the INRIA TEFIS platform has been described along with the associated components and structures. The document will be updated as and when required when other or modified function becomes available in that environment.

7. References

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

[2] https://www.irods.org/pubs/iRODS_Fact_Sheet-0907c.pdf

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

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

[5] TEFIS Deliverable D6_3_1 TIDS Prototype

[6] TEFIS Deliverable D6_1_1 Specifications for Experimental Metadata

[7] TEFIS Deliverable D3.3 User Tools Implementation

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