

# The iREAD Project: *iRODS™ Evaluation and Demonstrator.*

Martyn Fletcher

# iREAD project objectives.

- Evaluation of iRODS:
  - A demonstration implementation of the iRODS on the White Rose Grid.
  - An evaluation of the iRODS system in respect of:
    - Authentication.
    - Authorisation.
  - Analysis of the capabilities of iRODS for integration with existing e-Science SOA infrastructure.

# The iREAD Project

- Work funded by JISC:

<http://www.jisc.ac.uk/whatwedo/programmes/einfrastructure/iread.aspx>

- All documentation, results and demonstrations are available on the public website see:

<http://www.wrg.york.ac.uk/iread>

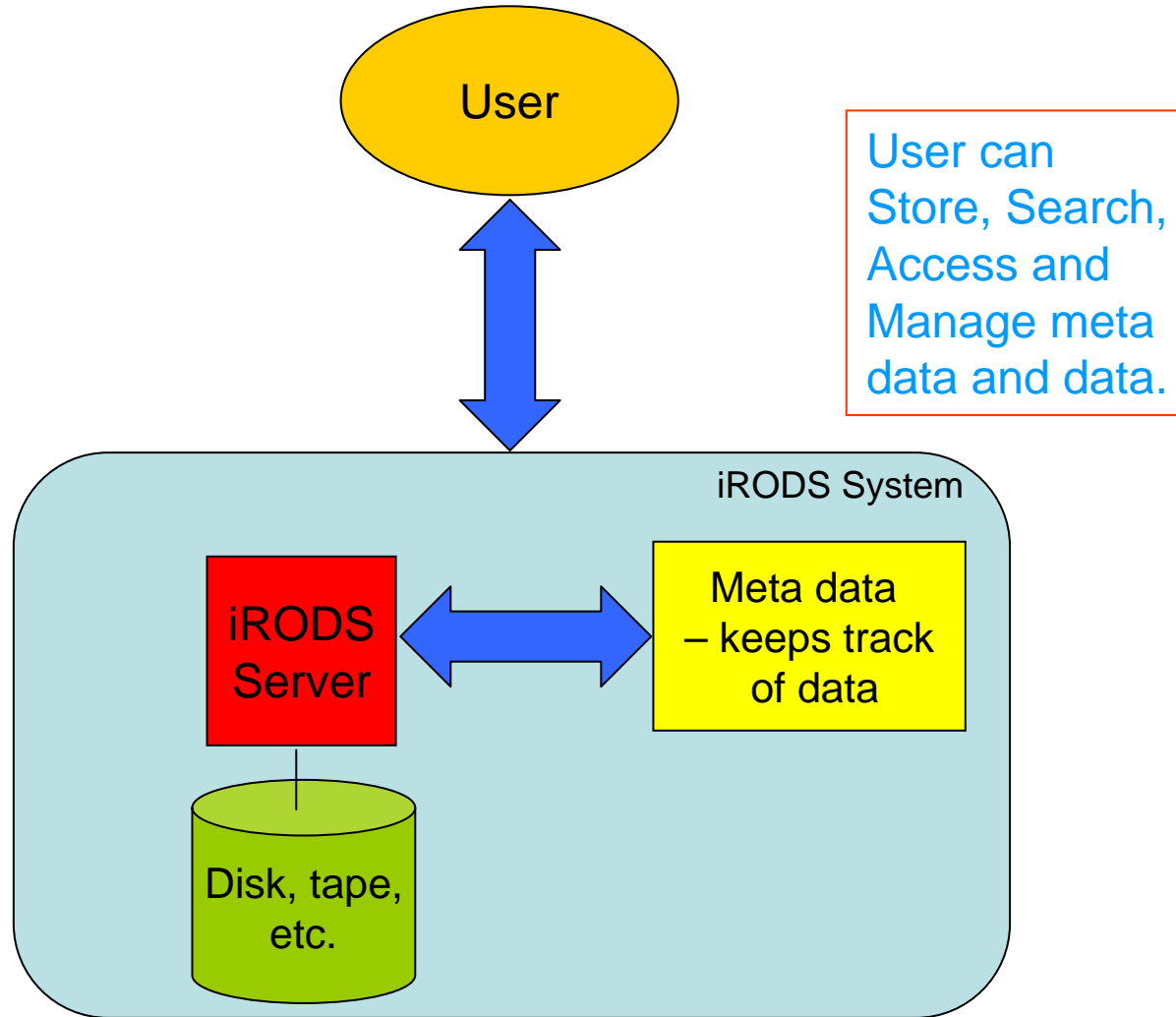
# Contents

- What is iRODS?
- Users + iRODS.
- What is an iRODS System?
- What are iRODS Micro-services / Rules?
- How does a user connect to iRODS (and programmatically)?
- iRODS Authentication and Authorisation?
- iRODS Zones / Federation?
- iREAD Demonstrations:
  - Data conversion.
  - Pattern searching.
- Conclusions / further work.

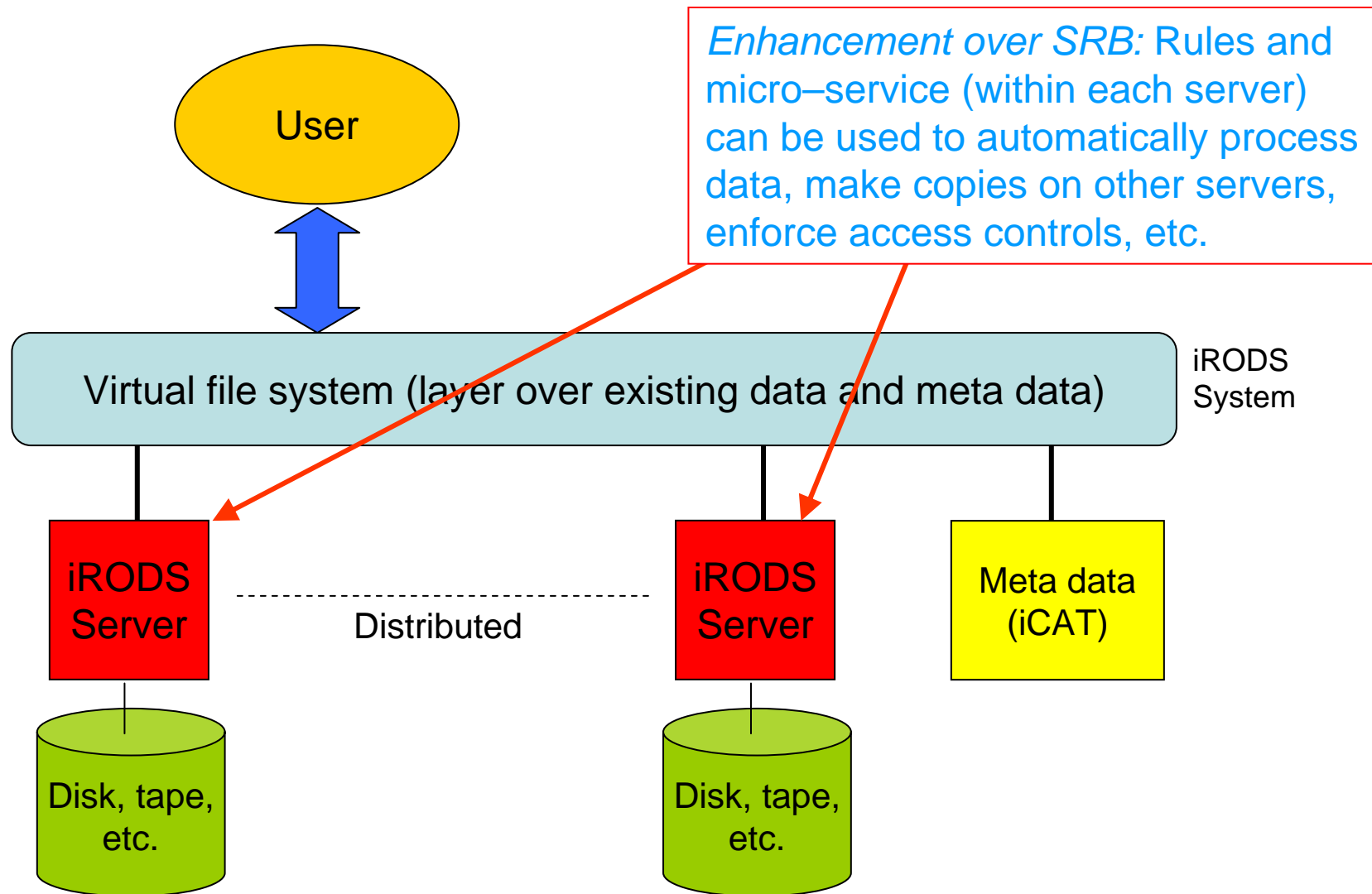
# What is iRODS?

- iRODS™ (Integrated Rule-Oriented Data System)  
<https://www.irods.org/> :
  - Data grid software system – virtual file system.
  - Like SRB supports Data Grids, Digital Libraries, Persistent Archives, etc.
  - Main addition (cf SRB):
    - Rules Engine executes rules: to decide how the system is to respond to various requests and conditions, apply management policies, etc.
    - Rule execute other rules and micro services.
  - Development of SRB by the Data Intensive Cyber Environments (DICE) group.
  - Written in C.
  - Eventually iRODS is expected to replace SRB.
  - Open source under a BSD license.

# Users + iRODS (Data + Meta data).



# Users + iRODS (virtual file system).



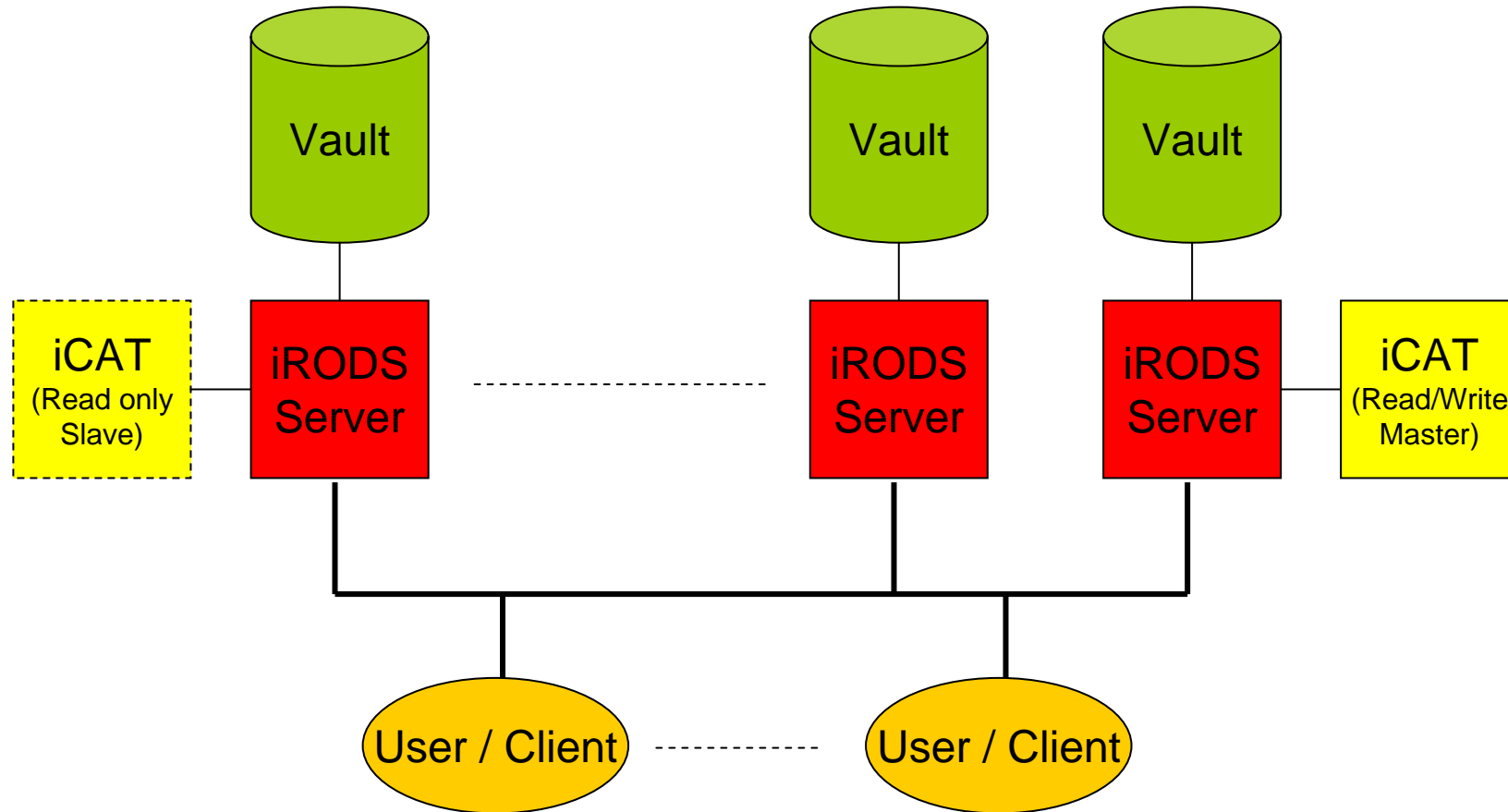
# What is an iRODS system?

- One or more interconnected resources:
  - Host machine each running an iRODS server.
  - Providing storage - in a “Vault” area within it’s file system.
  - Provides rule and micro service execution.
- Manage data files.
- Manages data about the files (meta data) stored in a catalog known as an iCAT (iRODS CATalog):
  - Managed by a Postgres server:
  - Contains all metadata for everything that is managed locally including Domains, Resources and Metadata of Objects, list of administrators within the zone.
  - Each iRODS system must have one (and only one) read / write iCAT (master).
  - Other resources generally do not have an iCATs.
  - But ... if there are iCAT access bottlenecks then these can be mitigated using additional read only iCAT (slave).

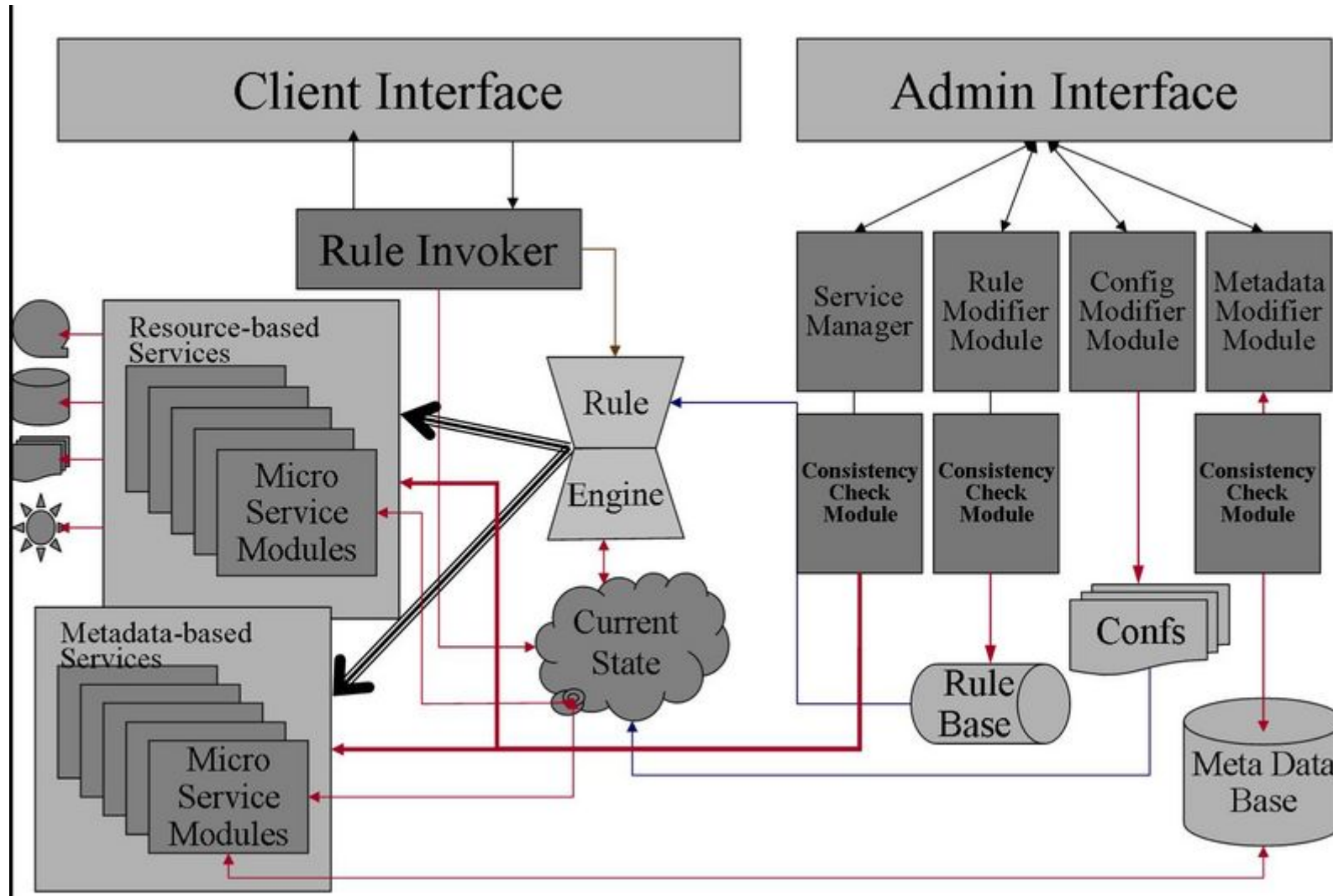
# What is an iRODS system?

- iRODS (version 2.0.1.) can use a Unix file system or a Windows file system as a resource.
- Clients:
  - CLI or GUIs.
  - Have access to all resources and can usually direct files to be stored on any resource.
  - Can operate on remote or local data on different types of resources through a common interface.
- Resources and clients distributed across the Internet.

# iRODS system



# iRODS server



# What are iRODS micro services?

- Actually perform actions.
- Only executed as part of a rule.
- Inbuilt micro services for accessing various parts of the system:
  - create , delete, read files and collections (directories), deny access to files, simple workflow constructs, etc.
- Users can create their own micro services:
  - Operate on data, run external web services, etc
  - We have created ones for searching, data conversion, etc. e.g. *msiSearch*.
- Adding or changing a micro service necessitates:
  - Re-compilation of the iRODS installation.
  - A stop / start of the iRODS system.
  - This situation may change in future.

# What are iRODS rules?

- A rule is a “grouping” of:
  - Other rules.
  - Micro-services.
- Can be executed:
  - Automatically on events, time, etc. – part of the rule base (core.irb file).
  - By users on command from API or rule files.
- Can be added to a live iRODS system:
  - No need for a recompile and stop / start.

# What are iRODS rules?

- They are of the form:

`Action_Name | Condition | Workflow_Chain | Recovery_Chain`

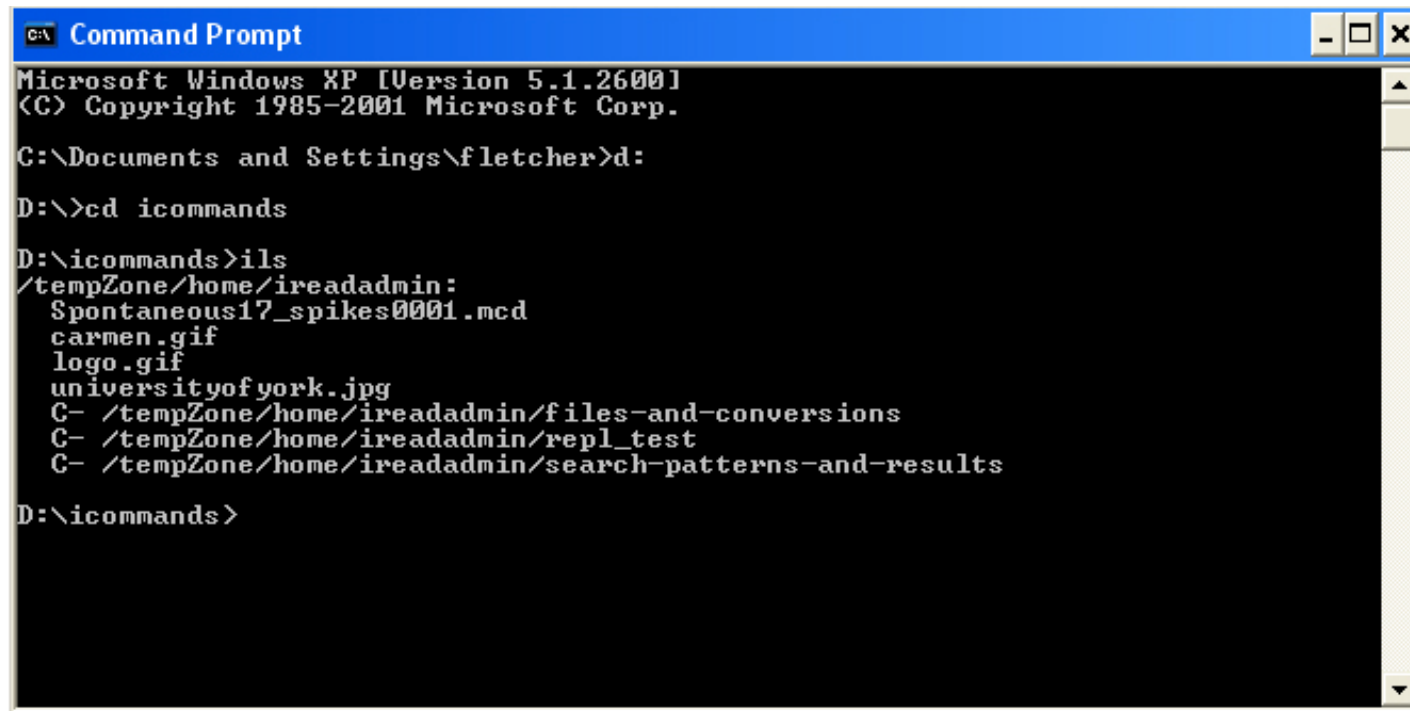
- *Action\_Name* is the name of the rule.
- *Condition* is the condition for execution of the rule.
- *Workflow\_Chain* is a list of micro-services or other rules to be executed.
- *Recovery\_Chain* is a list of recovery actions if any of the *Workflow\_Chain* fails.

# How do you connect to iRODS (1)?

- Clients available:
  - Command line
    - Install iRODS client and run icommands.
    - Client machine must have access to iRODS port (1247).
  - iRODS Explorer
    - Windows desktop application (cf Windows Explorer).
    - Client machine must have access to iRODS port (1247).
  - iRODS Web Browser:
    - Client machine does not need access to iRODS port (1247).

# Command line client

- Linux or Windows.
- Uses irodsEnv, etc. file for authentication.
- Connects to iRODS port 1247.
- Can be used to do almost anything (permissions, etc.).



```
C:\> Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\fletcher>d:

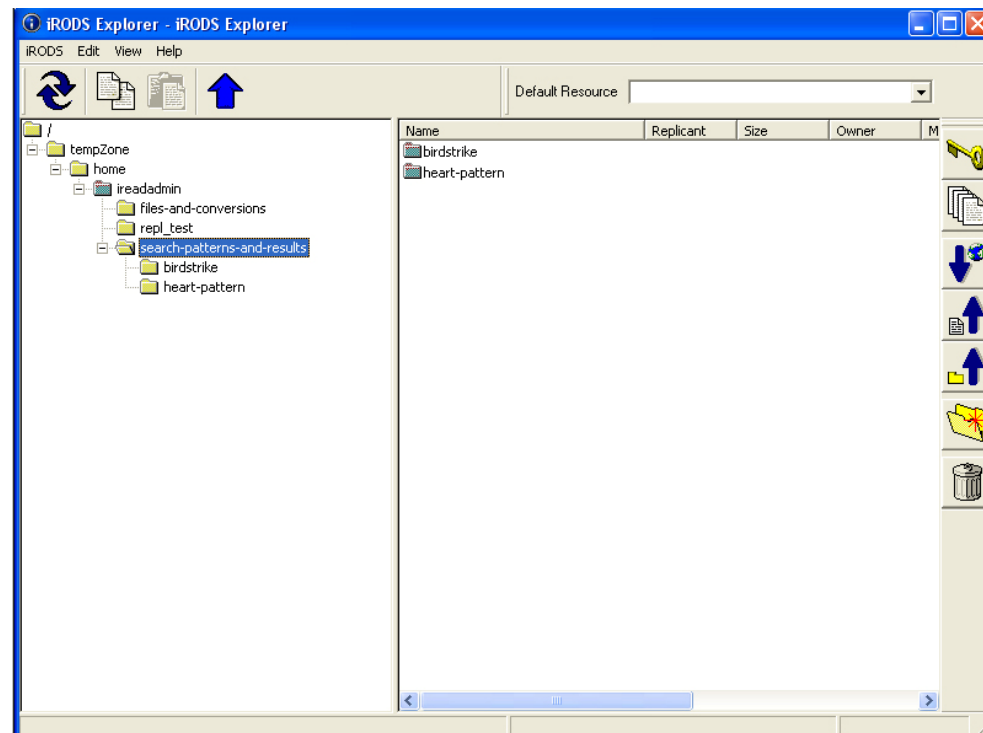
D:\>cd icommands

D:\icommands>ils
/tempZone/home/ireadadmin:
  Spontaneous17_spikes0001.mcd
  carmen.gif
  logo.gif
  universityofyork.jpg
C- /tempZone/home/ireadadmin/files-and-conversions
C- /tempZone/home/ireadadmin/repl_test
C- /tempZone/home/ireadadmin/search-patterns-and-results

D:\icommands>
```

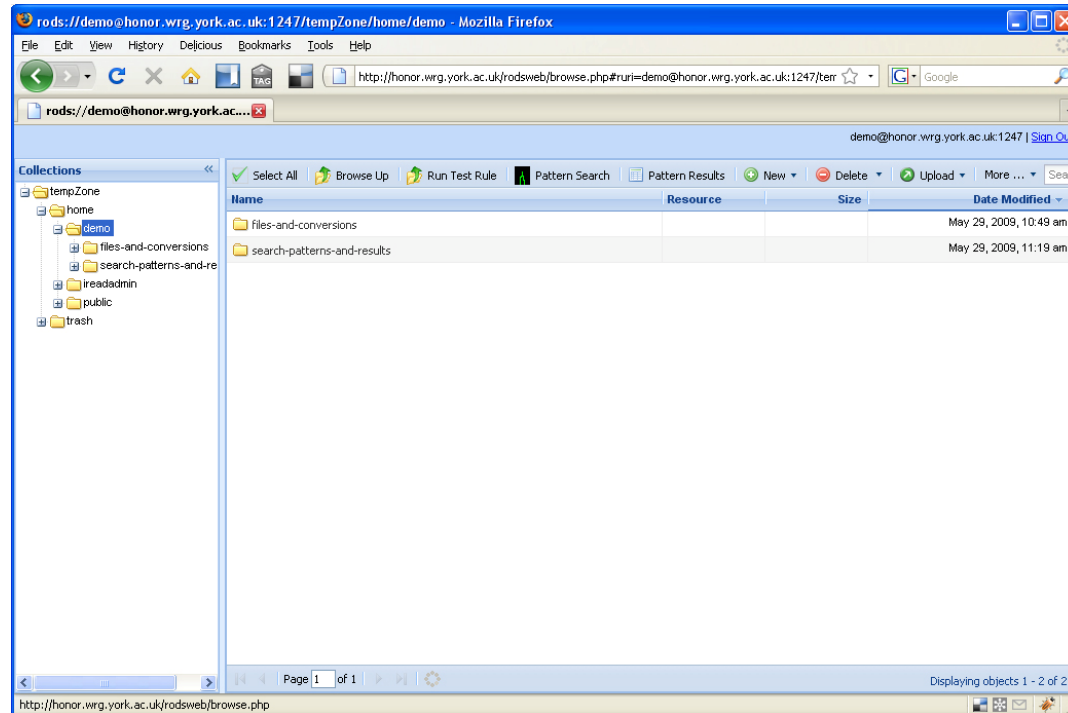
# Desktop client: iRODS Explorer

- Windows only desktop application.
- Uses irodsEnv, etc. file for authentication.
- Connects to iRODS port 1247.
- Can set permissions.

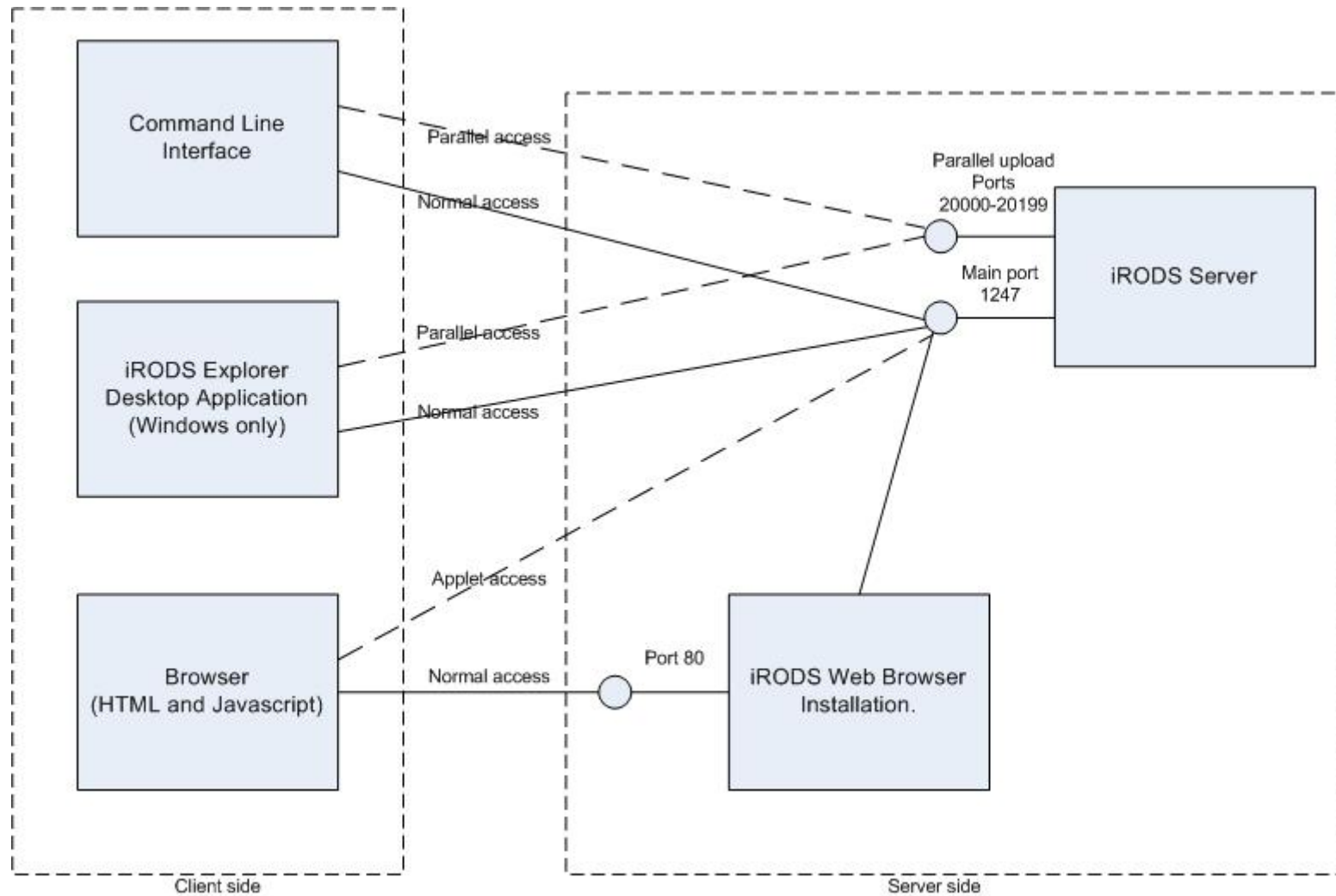


# iRODS web browser client

- Does not need to connect to iRODS port 1247 unless using files and directories (applet) upload.
- Authenticates through login page or url parameters.
- Cannot set permissions.



# Client port usage



# How do you programmatically connect to iRODS (2)?

- Client APIs available:
  - JARGON – a Java API.
  - PRODS - a PHP API - used by the browser client. Use in iREAD to run the search rules from the browser.
  - Recent additions:
    - Python.
    - Perl.

# How are users authenticated?

- Current released options available:
  - Default Encrypted Password.
  - Grid Security Infrastructure (GSI) – users have two names the iRODS name and the GSI DN.
- Kerberos is available (in the latest IRODS builds) and will be in next release.
  - As with GSI - users have two names the iRODS name and their Kerberos name.
- Future Use of Shibboleth via the ASPiS project.

# How are users authorised (1)?

- Access and ownership can be manipulated through:
  - The *ichmod* command (analogous to the Linux *chmod*) and Access Control Lists(ACL).
- Permissions can be set on:
  - A file or collection basis.
  - Can also use *inherit*.
  - Can create user groups.
- When a user is created:
  - A collection (directory) is created for the user in the user's name.
  - User is the owner of any files stored and has full control: read, write or delete.

# How are users authorised (2)?

- Rules can be used to:
  - Access ACLs and to modify access to dataObjects (iRODS files) and collections (directories).
  - Provide fine access control e.g. *acPreprocForDataObjOpen* can be used to check access is allowed before an object is opened.
- Use of more extensive set of authorization mechanisms probably necessary:
  - For example, PERMIS is would be more desirable for complex sharing requirements.
  - The ASPiS project is looking at PERMIS.

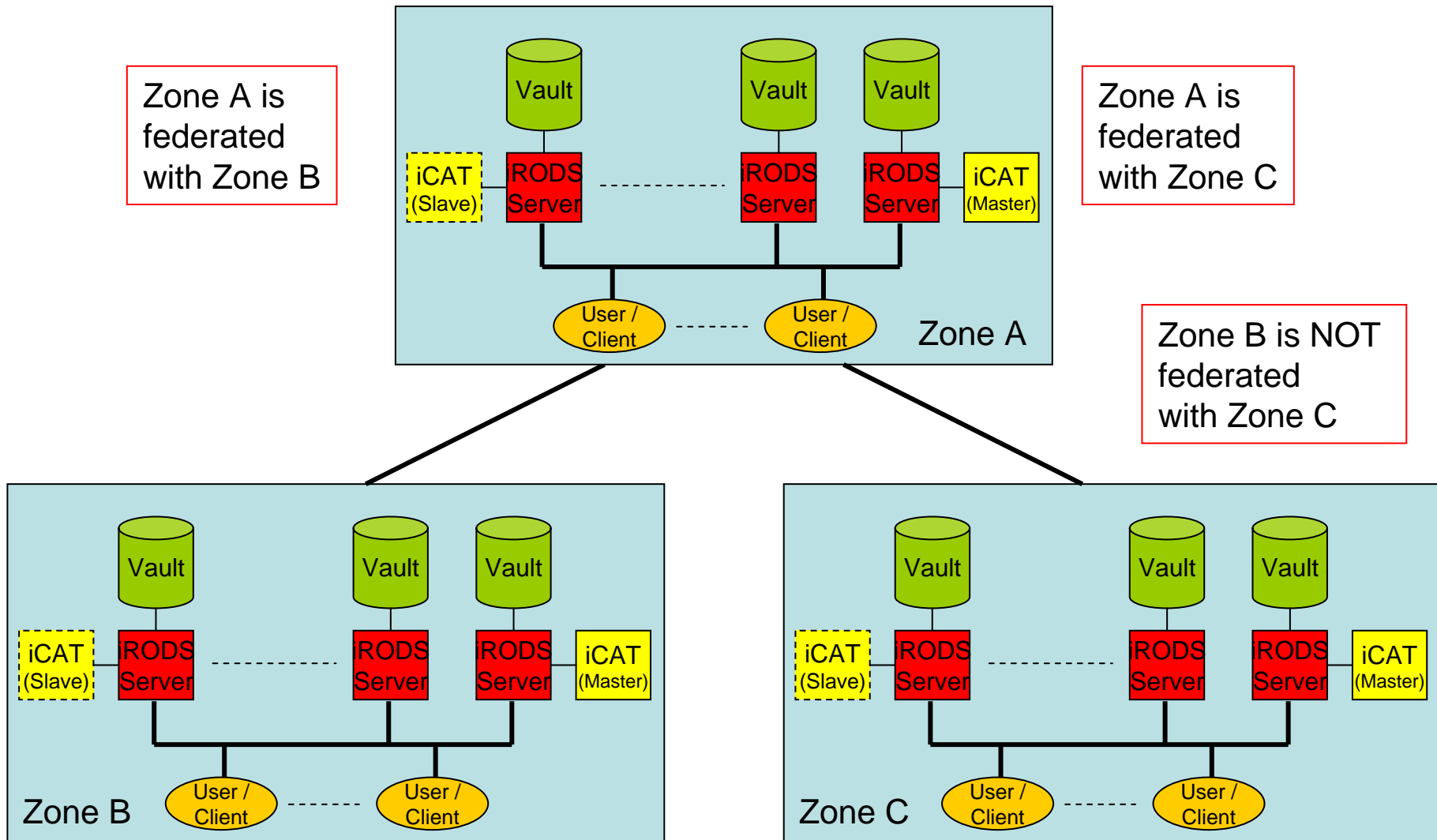
# Resource Groups

- Resource groups can be created and used.
- Resource groupings and rules can be used to direct incoming files at resources in the group:
  - Randomly if necessary.
  - Or using storage load balancing (this capability is in progress).
- Resource groups could also allow some form of coarse access control:
  - E.g. allow only certain users access to particular resources.

# What are Zones?

- An iRODS Zone is an independent iRODS system as described previously (multiple resources, etc.).
- A zone can interoperate with other independent iRODS zones (i.e. federate).

# Federation of Zones



# User and Zones

- If two zones federate, then they will exchange their user identification:
  - One zone knows about users in the other zone.
- The iCAT also contains information on other zones:
  - A zone table which contains information such as zoneName, the address of the iCAT enabled server in that zone, etc.
  - External user and user group information from other zones, etc.
- Every user has one and only one home zone, and is an alien in other zones (foreign zones) – users should:
  - Logon to their own home zone server – then all cross-zone operations will work.
  - NOT logon to alien zones – they can but cross-zone operations will not work.

# How do you convert an SRB system to use iRODS?

- A Perl conversion script *m2icat.pl* script is provided – it acts as a client and extracts meta catalogue data from the SRB MCAT and places it in the iRODS iCAT.
- The script takes data from a live SRB system and places it in a live iRODS system and has three phases:
  1. Get information from SRB installation.
  2. Build the command set to place data in iRODS.
  3. Run the command set and store data in iRODS installation.
- This script does not copy the actual file and collections stored – just the meta data.
  - The new iRODS instance will access the actual file and collections in the SRB Vault.
  - There may be issues here with SRB naming.
- This may not cover all instances but will be developed further over time.

# iREAD demonstration.

- The iREAD iRODS demonstration setup is as follows:
  - iRODS version 2.0.1.
  - The iRODS browser installation (part of the extrods 1.1.0.1 package) for connection to iRODS.
  - Apache web server 2.2.10 for the browser.
  - PHP 5.26 for the browser.
- Rules and micro-services have been added to iRODS as necessary.
- The browser installation and PHP service have also been modified for demonstration purposes.

# Data Conversion demonstration (1).

- Image conversion micro-services have been compiled and installed.
- Additional skeleton micro-services have been included for future conversion of Neurophysiology data formats.
- Installed rules (*acPostProcForPut*) for post processing of the put command.
- The *acPostProcPut* rules – only one is executed (each is a single line in the *core.irb* file):
  1. `acPostProcForPut|$objPath like /tempZone/home/demo/files-and-conversions/*.gif|msImageConvert($objPath,null,$objPath.jpg,null)|nop`
  2. `acPostProcForPut|$objPath like /tempZone/home/demo/files-and-conversions/*.bmp|msImageConvert($objPath,null,$objPath.png,null)|nop`
  3. `acPostProcForPut|$objPath like /tempZone/home/demo/files-and-conversions/*.mcd|msiDataObjCopy($objPath,$objPath.ndf,demoResc,*Junk)|nop`
  4. `acPostProcForPut||nop|nop`

# Data Conversion demonstration (2).

- Demonstration of data conversion from <http://www.wrg.york.ac.uk/iread>

# Search demonstration using web services (1).



# Search demonstration using web services (2).

- Created micro services which access the web service:
  - Used GSOAP for web service interface.
- Created rule to run the search micro services.
- Modified browser to add buttons, display functions, etc. (Javascript / HTML.)
- Created PHP “service” to run rule.

# Search demonstration using web services (3).

- The pattern search rule:

```
MyNewSearchTestRule||msiDataObjUnlink(*resultsPath,*STATUS)#  
#msiDataObjOpen(*queryPath,*QUERY_FD)##msiDataObjRead(*Q  
QUERY_FD,10000,*QUERY_BUF)##msiDataObjClose(*QUERY_FD,*  
junk)##msiNewRemoteMthSearch(*QUERY_BUF,*CIIP,*HtIP,*HtTim  
e,*HtIndex)##msiNewRemoteGetResult(*CIIP,*HtIP,*HtTime,*HtInde  
x,*results)##msiDataObjCreate(*resultsPath,null,*RESULTS_FD)##  
msiDataObjWrite(*RESULTS_FD,*results,100)##msiDataObjClose(*  
RESULTS_FD,*junk)|nop##nop##nop##nop##nop##nop##nop##no  
p##nop
```

# Search demonstration using web services (4).

- Micro services used (in following :
  - `msiDataObjUnlink` - delete the existing *results.txt* file.
  - `msiDataObjOpen` - open the *query.txt* file.
  - `msiDataObjRead` - read the *query.txt* file.
  - `msiDataObjClose` - close the *query.txt* file.
  - `msiNewRemoteMthSearch` - invoke the search web service and pass the query data for matching.
  - `msiNewRemoteGetResult` - poll the search service and when it has finished search retrieve the results.
  - `msiDataObjCreate` - create a new *results.txt* file.
  - `msiDataObjWrite` - write the retrived results to the new *results.txt* file.
  - `msiDataObjClose` - close the *results.txt* file.

# Search demonstration using web services (5).

- Demonstration of searching from <http://www.wrg.york.ac.uk/iread>

# Conclusions / experience (1)

- iRODS rules and micro services have been evaluated and demonstrations are provided which perform
  - automated data conversion.
  - execution of external web service examples including pattern searching.
- The rules and micro services can be executed automatically or as required without the browser.
- iRODS can be integrated into a system such as CARMEN could provide data conversion and other automatic operations.

# Conclusions / experience (2)

- iRODS could be used as the underlying system in a distributed architecture:
  - Rules and micro services can be run remotely on other servers (on other resources in the same zone and in other zones) using the *remoteExec* micro service.
  - These could also be used to run web service as necessary.
- Parallel execution and broadcast execution modes would greatly assist the use of iRODS for distributed operation. These are promised.

# Conclusions / experience (3)

- Micro services (and rules) need to be installed on every server where they need to be executed.
- Rules can become very long and unwieldy and it is necessary to plan them carefully and use multiple sub rules to keep the situation manageable.
- It would be beneficial to be able to add micro services to a live system without:
  - Recompiling the iRODS server.
  - Starting and stopping the server.

# Conclusions / experience (4)

- Authentication provided: encrypted password and / or GSI - recent additions to iRODS include
  - Kerberos authentication.
  - Soon the ASPiS project will provide a Shibboleth capability for the iRODS web browser.
- Authorisation provided:
  - User groupings similar to that provided in Unix.
  - Rules can be used to provide fine access control, for example, the *acPreprocForDataObjOpen* rule can be used to check access is allowed before an object is opened.
  - A better authorization mechanism (e.g. PERMIS) is probably required for extensive sharing systems.

# Conclusions / experience (5)

- Debugging:
  - Information not extensive:
    - E.g. a micro service can fail producing a non informative messages on the command line and in the *rodsLog*, (iRODS log file) particularly if a micro service fails catastrophically.
    - Usually had to comment out code in the micro service and add *rodsLog* messages as appropriate.
  - When debugging browser operation / modifications, Firebug (a Firefox plugin) was found to be very useful for Javascript debugging and viewing data sent back from PHP calls.
  - Also, when debugging services, extensive use was made of the GSOAP logging - the SOAP log files are located in *server/bin/SENT.log* and *server/bin/RECV.log*.

# Conclusions / experience (6)

- When using the iRODS web browser:
  - The user needs to be aware of browser caching issues; e.g. file downloads can present the previous version of a newly changed file.
  - The *Upload - Files and Directories* menu option uses an applet which connects directly to the iRODS server port. For security reasons the iREAD iRODS port connection is not currently open to the outside world.

# Potential Future Work (1)

- Provide automatic:
  - Generation of the search databases (as new data arrives).
  - Search for known conditions in bulk data.
- Automated adjunct to the Signal Data Explorer.
- Wide range of potential applications:
  - Many engineering disciplines.
  - Health care (particularly if image recognition micro services were also utilized).

# Potential Future Work (2)

- Scalability:
  - Put together (or being able to use) an iRODS arrangement with
    - Multiple zones
    - Each zone has multiple resources.
  - Run rules and micro services across resources and zones.
  - Test scalability of distributed operations (e.g. searching) using iRODS – analogous to the Pattern Match Controller architecture used with the Signal Data Explorer.
- Distributed usage - greatly assisted greatly if broadcast execution is implemented within future iRODS versions.

# Potential Future Work (3)

- iRODS would provide benefits to the CARMEN project if it was installed in place of the current SRB storage system:
  - To provide automatic data conversion to the standard NDF format as data is uploaded.
  - Would probably have to include an iRODS server installed on a Windows machine because many of the data converters only work under Windows.

# The iREAD Project

- All documentation, experiences, results and demonstrations are available on the public website see: <http://www.wrg.york.ac.uk/iread>
- A paper has been submitted to the All Hands Conference 2009.
- Questions?