EUDAT

Introduction to iRODS

What is it, what it can do

Mark van de Sanden
SURFsara
Dutch National HPC center, The Netherlands
Training Day, 2nd EUDAT Conference, Rome, Italy
28 October 2013

Outline

- What is iRODS
- iRODS
  - Rule Engine
  - Metadata Catalog
  - Data Organization
  - Interfaces
  - Storage Resources
  - iRODS Federation
- What makes iRODS suitable for use in EUDAT

What is iRODS?

You, Researchers, Students, etc.

Want to easily Find, Access, Use, Move, Share Data, and more…

With your Interfaces, your Applications, your Workflows

iRODS Data System – "Middleware"

A "layer" that "connects the dots" while masking and automating your interactions with diverse infrastructure.

The "World of Infrastructure"

Your and other's Storage, Networks, Admin. Domains, Computing Services, Web Services, etc.

D·I·C·E

i·R·O·D·S

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

UCSD

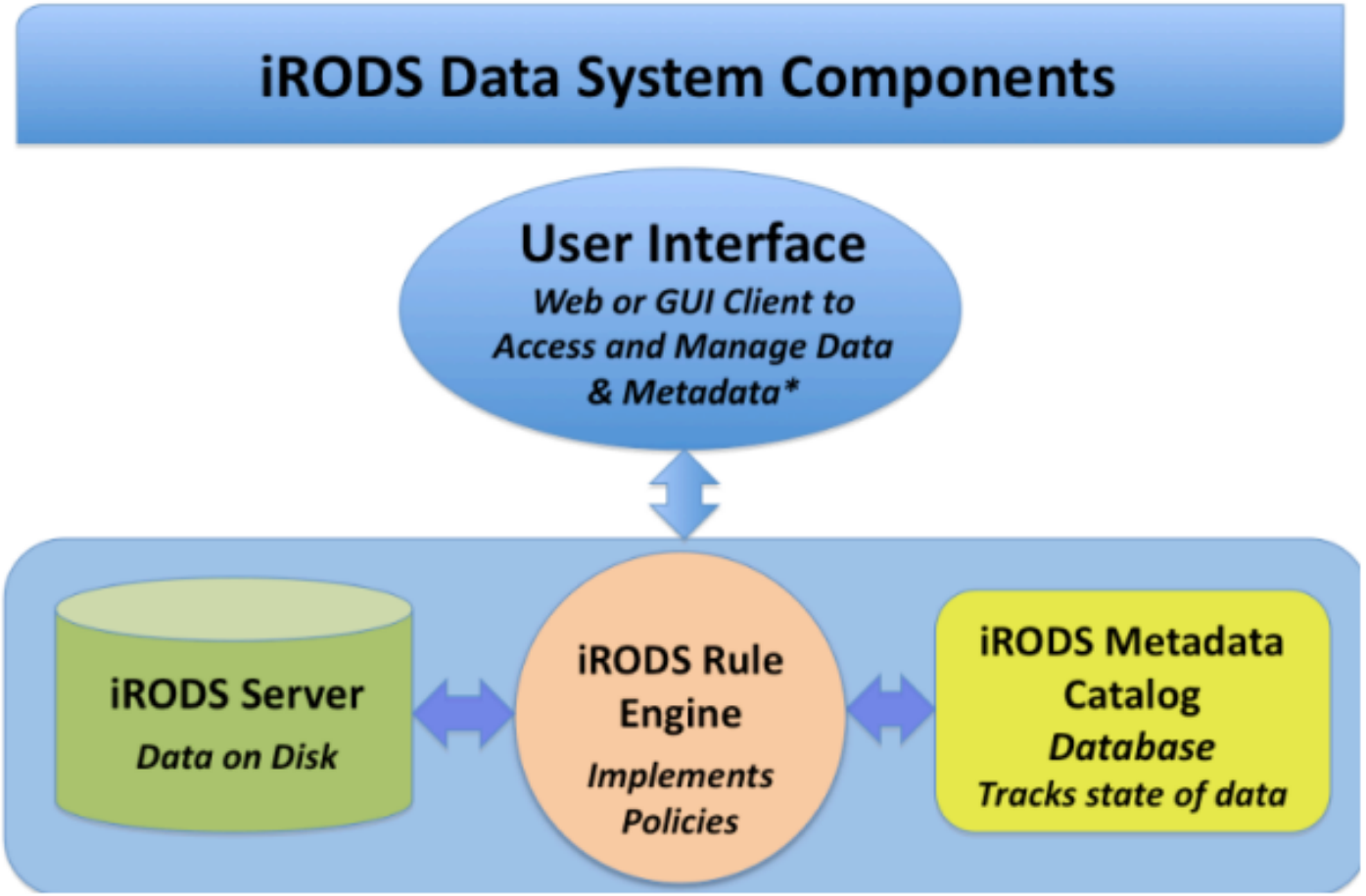
NSF

THE NATIONAL ARCHIVES

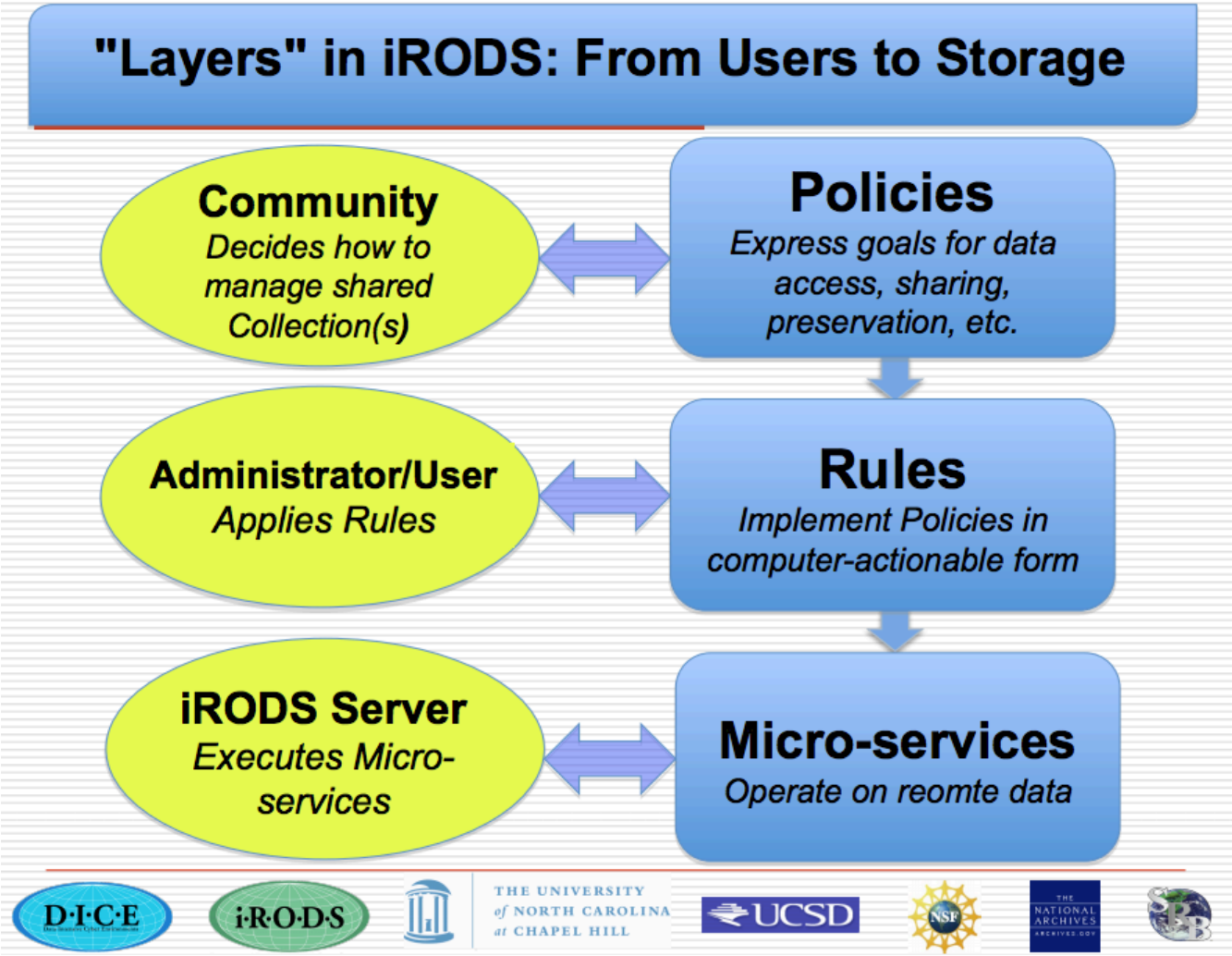
What does iRODS provide?

- Storage virtualization of different disk and tape storage systems
- A logical namespace across storage locations
- A policy engine to can automate data management according to defined rules
- A method to create and define user specific procedures and functions
- Various client interfaces
- A flexible architecture

iRODS Components



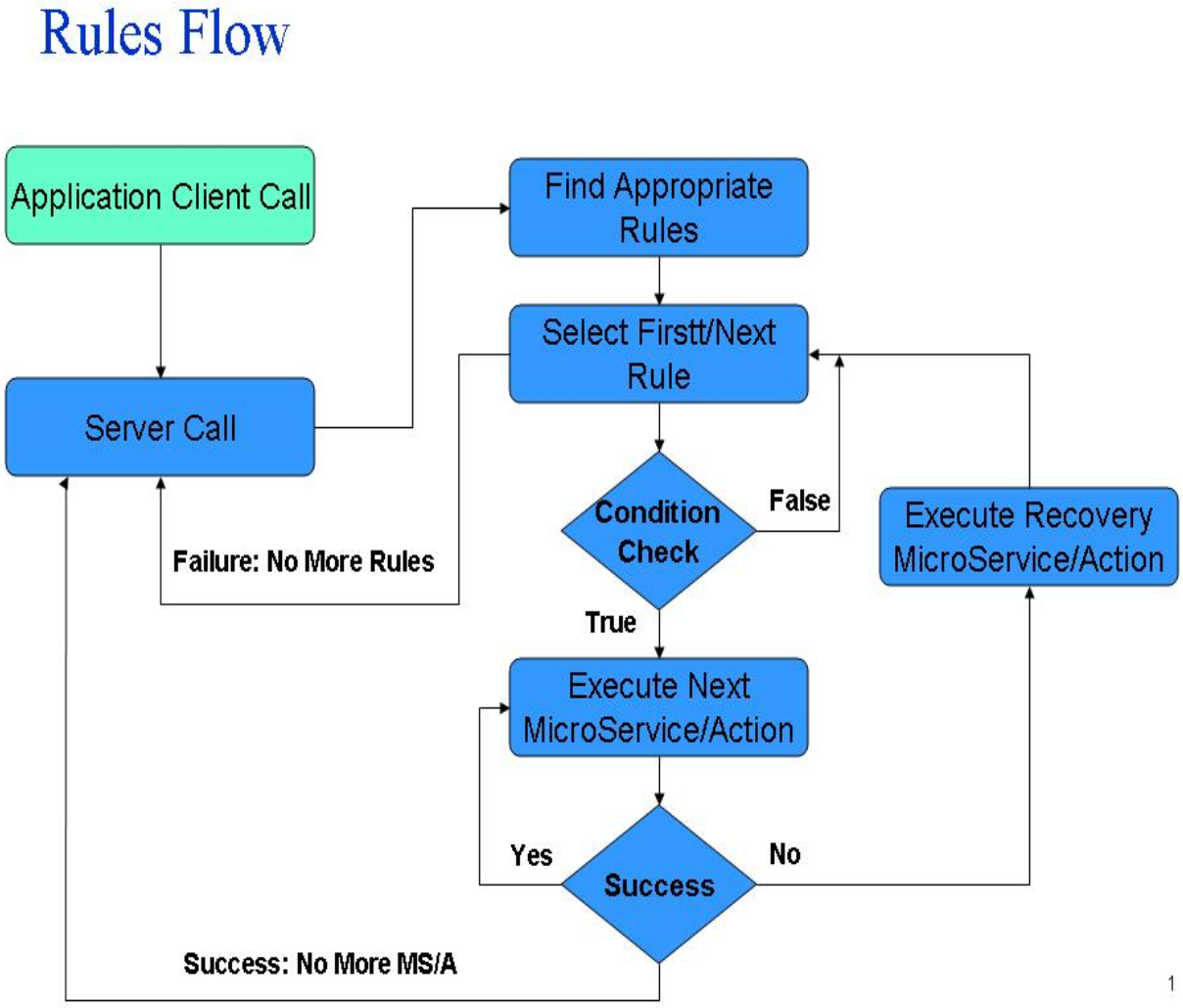
iRODS Rule Engine



iRODS Rule Engine

- Workflow engine to automate:
  - Policies on data access, sharing, preservation, …
  - To implement user or administrator applied rules to enforce defined policies administrator
  - Rules are implemented via executing micro services
  - Micro services are small programs to implement some functionality

iRODS Rules Flow



Policy Enforcement Points

- Locations within iRODS framework where an event or state (of the environment) prompts a rule to execute
  - Each action may invoke multiple policy enforcement points
- Policies enforcement points
  - Pre-action policy (e.g selection of storage location)
  - Execution/action policy (e.g. file deletion)
  - Post-action policy (e.g. create secondary data products)
- Actions (trigger rules) are contained in

  iRODS/server/config/reConfigs/core.re

Policy Enforcement Points (71)

ACTION

acCreateUser
acDeleteUser
acGetUserbyDN
acTrashPolicy
acAclPolicy
acSetCreateConditions
acDataDeletePolicy
acRenameLocalZone
acSetRescSchemeForCreate
acRescQuotaPolicy
acSetMultiReplPerResc
acSetNumThreads
acVacuum
acSetResourceList
acSetCopyNumber
acVerifyChecksum
acCreateUserZoneCollections
acDeleteUserZoneCollections
acPurgeFiles
acRegisterData
acGetIcatResults
acSetPublicUserPolicy
acCreateDefaultCollections
acDeleteDefaultCollections

PRE-ACTION POLICY

acPreProcForCreateUser
acPreProcForDeleteUser
acPreProcForModifyUser
acPreProcForModifyUserGroup
acChkHostAccessControl
acPreProcForCollCreate
acPreProcForRmColl
acPreProcForModifyAVUMetadata
acPreProcForModifyCollMeta
acPreProcForModifyDataObjMeta
acPreProcForModifyAccessControl
acPreprocForDataObjOpen
acPreProcForObjRename
acPreProcForCreateResource
acPreProcForDeleteResource
acPreProcForModifyResource
acPreProcForModifyResourceGroup
acPreProcForCreateToken
acPreProcForDeleteToken
acNoChkFilePathPerm
acPreProcForGenQuery
acSetReServerNumProc
acSetVaultPathPolicy

POST-ACTION POLICY

acPostProcForCreateUser
acPostProcForDeleteUser
acPostProcForModifyUser
acPostProcForModifyUserGroup
acPostProcForDelete
acPostProcForCollCreate
acPostProcForRmColl
acPostProcForModifyAVUMetadata
acPostProcForModifyCollMeta
acPostProcForModifyDataObjMeta
acPostProcForModifyAccessControl
acPostProcForOpen
acPostProcForObjRename
acPostProcForCreateResource
acPostProcForDeleteResource
acPostProcForModifyResource
acPostProcForModifyResourceGroup
acPostProcForCreateToken
acPostProcForDeleteToken
acPostProcForFilePathReg
acPostProcForGenQuery
acPostProcForPut
acPostProcForCopy
acPostProcForCreate

Format of a Rule

- Action | Condition | MS_1, …, MS_n | RMS_1, …, RMS_n
- Action
  - Name of action to be performed
  - Name known to the server and invoked by server
- Condition – condition under which the rule apply
- Micro-services - If applicable micro services will be executed
- Recovery micro-service - If any micro service fails, recovery micro service(s) executed to maintain transactional consistency
- Example of MS/RMS
  - createFile(*F) removeFile(*F)
  - ingestMetadata(*F,*M) rollback

Format of a Rule

```
Rule_name{
    microservice1(…,*A,…,*B);
    microservice2(*A,…);
}
INPUT *A=”first_input", *B=”second_input"
OUTPUT ruleExecOut
```

(*A and *B are here just for illustrative purposes…)

OR

“ruleExecOut” is a structure managed by iRODS.

```
Rule_name(*arg)  {
    on(exp)  {
      microservice1(…,*arg);
      microservice2(…);
    }
}
INPUT null
OUTPUT ruleExecOut
```

- A rule can take arguments.
- A rule can be executed conditionally.
- Use “null” if there are no input parameters.

Example Policy Implementation

Using asPostProcForPut to implement policy: inputs to a specific resource

Data coming in to a target iRODS **collection** triggers a script that takes some desired action (sending data to a remote ftp site)

```
acPostProcForPut{ on($objPath like "/eu00Zone/home/data/*") {
      writeLine("serverLog", "$userNameClient sending $objPath");
      msiSplitPath($filePath,*fileDir,*fileName);
      msiExecCmd("ftp_to_remote.sh","*fileDir \
   *fileName" ,"null","null","null",*Out);
}
```

Micro-services (MSs)

- Well-defined Server-side Procedures and Functions
- C functions on servers
- MSs can be chained to form workflow using '##'

```
msiDataObjOpen(*A,*S_FD)##
msiDataObjRead(*S_FD,10000,*R_BUF)##
msiDataObjClose(*D_FD,*stat)
```

- Flow control
  - whileExec - while loop
  - forExec – for loop
  - forEachExec – for each in the table or list
  - break
  - ifExec – if-else

Out-of-the-Box Services

Microservices for…

- Queries on metadata catalog
- Interaction with web services
- Invocation of external applications
- Workflow constructs (loops, conditionals, exit)
- Remote and delayed execution control

Available at:

https://www.irods.org/doxygen/

Micro services (1/2)

print_hello_arg
msiVacuum
msiQuota
msiGoodFailure
msiSetResource
msiCheckPermission
msiCheckOwner
msiCreateUser
msiCreateCollByAdmin
msiSendMail
recover_print_hello
msiCommit
msiRollback
msiDeleteCollByAdmin
msiDeleteUser
msiAddUserToGroup
msiSetDefaultResc
msiSetRescSortScheme
msiSysReplDataObj
msiStageDataObj
msiSetDataObjPreferredResc
msiSetDataObjAvoidResc
msiSortDataObj
msiSysChksumDataObj
msiSetDataTypeFromExt
msiSetNoDirectRescInp
msiSetNumThreads
msiDeleteDisallowed
msiOprDisallowed

msiDataObjCreate
msiDataObjOpen
msiDataObjClose
msiDataObjLseek
msiDataObjRead
msiDataObjWrite
msiDataObjUnlink
msiDataObjRepl
msiDataObjCopy
msiExtractNaraMetadata
msiSetMultiReplPerResc
msiAdmChangeCoreIRB
msiAdmShowIRB
msiAdmShowDVM
msiAdmShowFNM
msiAdmAppendToTopOfCoreIRB
msiAdmClearAppRuleStruct
msiAdmAddAppRuleStruct
msiGetObjType
msiAssociateKeyValuePairsToObj
msiExtractTemplateMDFromBuf
msiReadMDTemplateIntoTagStruct
msiDataObjPut
msiDataObjGet
msiDataObjChksum
msiDataObjPhymv
msiDataObjRename
msiDataObjTrim
msiCollCreate

msiRmColl
msiReplColl
msiCollRepl
msiPhyPathReg
msiObjStat
msiDataObjRsync
msiFreeBuffer
msiNoChkFilePathPerm
msiNoTrashCan
msiSetPublicUserOpr
whileExec
forExec
delayExec
remoteExec
forEachExec
msiSleep
writeString
writeLine
writeBytesBuf
writePosInt
writeKeyValPairs
msiGetDiffTime
msiGetSystemTime
msiHumanToSystemTime
msiStrToBytesBuf
msiApplyDCMetadataTemplate
msiListEnabledMS
msiSendStdoutAsEmail
msiPrintKeyValPair

msiGetValByKey
msiAddKeyVal
assign
ifExec
break
applyAllRules
msiExecStrCondQuery
msiExecStrCondQueryWithOptions
msiExecGenQuery
msiMakeQuery
msiMakeGenQuery
msiGetMoreRows
msiAddSelectFieldToGenQuery
msiAddConditionToGenQuery
msiPrintGenQueryOutToBuffer
msiExecCmd
msiSetGraftPathScheme
msiSetRandomScheme
msiCheckHostAccessControl
msiGetIcatTime
msiGetTaggedValueFromString
msiXmsgServerConnect
msiXmsgCreateStream
msiCreateXmsgInp
msiSendXmsg
msiRcvXmsg
msiXmsgServerDisConnect
msiString2KeyValPair
msiStrArray2String
msiRdaToStdout

Micro services (2/2)

msiRdaToDataObj
msiRdaNoResults
msiRdaCommit
msiAW1
msiRdaRollback
msiRenameLocalZone
msiRenameCollection
msiAclPolicy
msiRemoveKeyValuePairsFromObj
msiDataObjPutWithOptions
msiDataObjReplWithOptions
msiDataObjChksumWithOptions
msiDataObjGetWithOptions
msiSetReServerNumProc
msiGetStdoutInExecCmdOut
msiGetStderrInExecCmdOut
msiAddKeyValToMspStr
msiPrintGenQueryInp
msiTarFileExtract
msiTarFileCreate
msiPhyBundleColl
msiWriteRodsLog
msiServerMonPerf
msiFlushMonStat
msiDigestMonStat
msiSplitPath
msiGetSessionVarValue
msiAutoReplicateService

msiDataObjAutoMove
msiGetContInxFromGenQueryOut
msiSetACL
msiSetRescQuotaPolicy
msiPropertiesNew
msiPropertiesClear
msiPropertiesClone
msiPropertiesAdd
msiPropertiesRemove
msiPropertiesGet
msiPropertiesSet
msiPropertiesExists
msiPropertiesToString
msiPropertiesFromString
msiRecursiveCollCopy
msiGetDataObjACL
msiGetCollectionACL
msiGetDataObjAVUs
msiGetDataObjPSmeta
msiGetCollectionPSmeta
msiGetDataObjAIP
msiLoadMetadataFromDataObj
msiExportRecursiveCollMeta
msiCopyAVUMetadata
msiGetUserInfo
msiGetUserACL
msiCreateUserAccountsFromDataObj
msiLoadUserModsFromDataObj

msiDeleteUsersFromDataObj
msiLoadACLFromDataObj
msiGetAuditTrailInfoByUserID
msiGetAuditTrailInfoByObjectID
msiGetAuditTrailInfoByActionID
msiGetAuditTrailInfoByKeywords
msiGetAuditTrailInfoByTimeStamp
msiSetDataType
msiGuessDataType
msiMergeDataCopies
msiIsColl
msiIsData
msiGetCollectionContentsReport
msiGetCollectionSize
msiStructFileBundle
msiCollectionSpider
msiFlagDataObjwithAVU
msiFlagInfectedObjs

Example Micro Service

```
00114 /**
00115  * \fn msiWriteRodsLog (msParam_t *inpParam1,  msParam_t *outParam, ruleExecInfo_t *rei)
00117  * \brief Writes a message into the server rodsLog.
00119  * \module core
00121  * \since 2.3
00123  * \author  Jean-Yves Nief
00124  * \date    2009-06-15
00126  * \note  This call should only be used through the rcExecMyRule (irule) call
00127  *       i.e., rule execution initiated by clients and should not be called
00128  *       internally by the server since it interacts with the client through
00129  *       the normal client/server socket connection.
00131  * \usage See clients/icommands/test/rules3.0/
00133  * \param[in] inpParam1 - A STR_MS_T which specifies the message to log.
00134  * \param[out] outParam - An INT_MS_T containing the status.
00135  * \param[in,out] rei - The RuleExecInfo structure that is automatically
00136  *    handled by the rule engine. The user does not include rei as a
00137  *    parameter in the rule invocation.
00139  * \DolVarDependence none
00140  * \DolVarModified none
00141  * \iCatAttrDependence none
00142  * \iCatAttrModified none
00143  * \sideeffect none
00144  *
00145  * \return integer
00146  * \retval 0 upon success
00147  * \pre N/A
00148  * \post N/A
00149  * \sa N/A
00150 **/
```

Example Micro Service

```
00151 int
00152 msiWriteRodsLog (msParam_t *inpParam1,  msParam_t *outParam, ruleExecInfo_t *rei)
00153 {
00154    rsComm_t *rsComm;
00155
00156    RE_TEST_MACRO (" Calling msiWriteRodsLog")
00157
00158    if (rei == NULL || rei->rsComm == NULL) {
00159       rodsLog (LOG_ERROR,
00160       "msiWriteRodsLog: input rei or rsComm is NULL");
00161       return (SYS_INTERNAL_NULL_INPUT_ERR);
00162    }
00163
00164    rsComm = rei->rsComm;
00165
00166    if ( inpParam1 == NULL ) {
00167       rodsLogAndErrorMsg (LOG_ERROR, &rsComm->rError, rei->status,
00168       "msiWriteRodsLog: input Param1 is NULL");
00169       rei->status = USER__NULL_INPUT_ERR;
00170       return (rei->status);
00171    }
00172
00173    if (strcmp (inpParam1->type, STR_MS_T) == 0) {
00174       rodsLog(LOG_NOTICE,
00175         "msiWriteRodsLog message: %s", inpParam1->inOutStruct);
00176    } else {
00177       rodsLogAndErrorMsg (LOG_ERROR, &rsComm->rError, rei->status,
00178       "msiWriteRodsLog: Unsupported input Param1 types %s",
00179       inpParam1->type);
00180       rei->status = UNKNOWN_PARAM_IN_RULE_ERR;
00181       return (rei->status);
00182    }
00183
00184    rei->status = 0;
00185
00186    fillIntInMsParam (outParam, rei->status);
00187
00188    return (rei->status);
00189 }
```

EUDAT Rules

- Rules for Replication and PID handling

getEpicApiParameters
getSharedCollection
writeFile
logInfo
logDebug
logError
logWithLevel
readFile
updateCommandName
updateMonitor
retrieveChecksum
triggerReplication

triggerCreatePID
triggerUpdateParentPID
processReplicationCommandFile
processPIDCommandFile
doReplication
createPID
createPIDgriffin
addPIDWithChecksum
searchPID
searchPIDchecksum
CheckReplicas
updatePIDWithNewChild
getRorPid

Discovery: Metadata

iCAT Database

- System Metadata
  - User name space
    - Address / e-mail / telephone number
    - Role (administrator, curator, user)
  - File name space
    - Creation date / size / location / checksum
    - Owner / access controls
  - Storage resource name space
    - Capacity / quotas / Type (archive, disk, fast cache)
- Domain Metadata
  - User-given metadata
    - Key-Value-Unit Triplets, Annotations
    - Relational / XML Metadata
    - Domain-specific Schemas
      - Dublin Core, Darwin Core, FITS, DICOM, …

user/passwords, GSI, Kerberos, OS_auth, PAM/LDAP

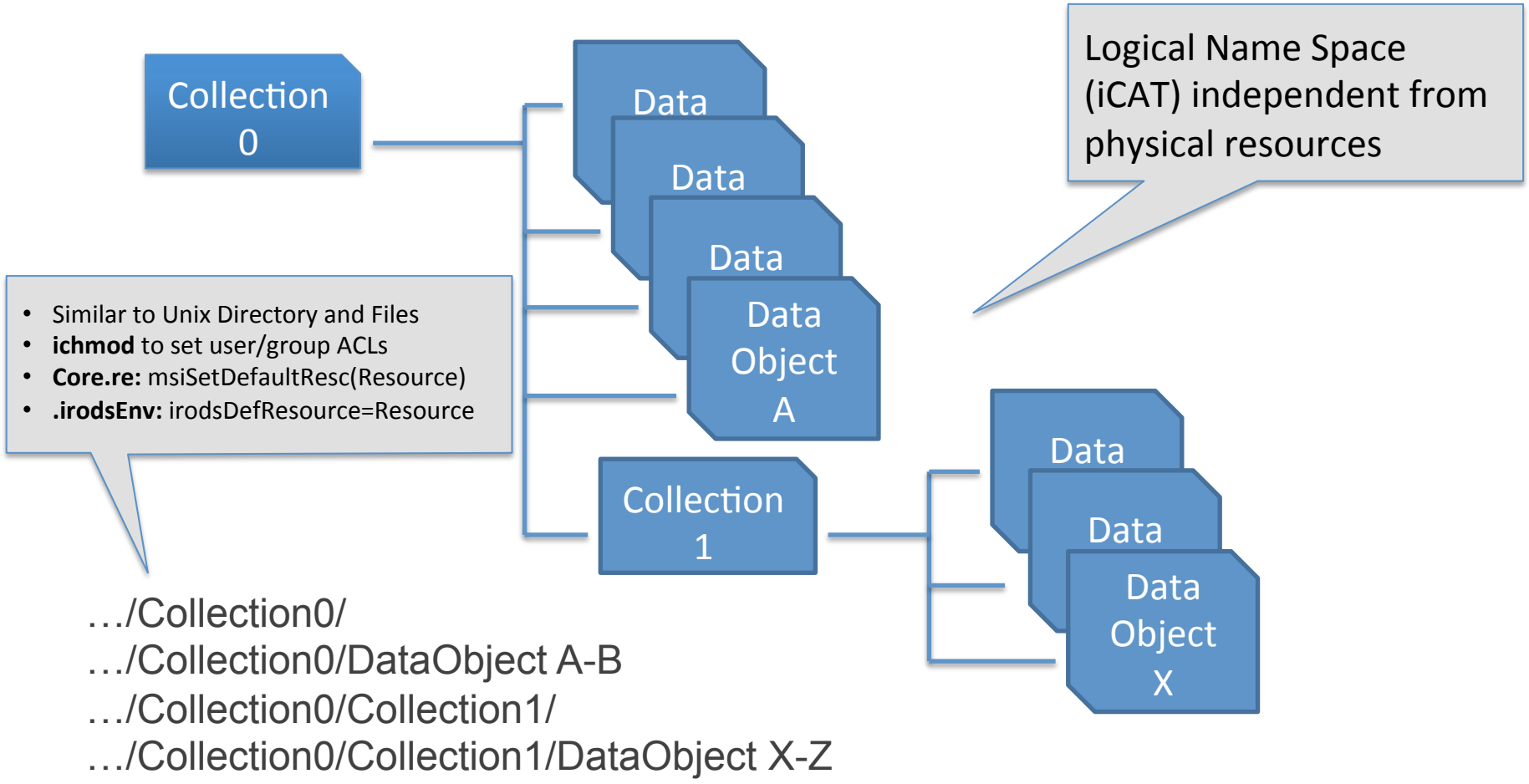
Is not used in EUDAT, communities have there own metadata repositories

Reading user-defined metadata

```
acGetDataObjAVU{
 msiMakeQuery("META_DATA_ATTR_NAME, META_DATA_ATTR_VALUE, COLL_NAME,
    DATA_NAME", "COLL_NAME = '*CollName'", *Query);
    msiExecStrCondQuery(*Query, *GenQOut);
forEachExec(*GenQOut){
  msiGetValByKey(*GenQOut, META_DATA_ATTR_VALUE, *AttrValue);
  msiGetValByKey(*GenQOut, META_DATA_ATTR_NAME, *AttrName);
  msiGetValByKey(*GenQOut, DATA_NAME, *name);
  writeLine(stdout,"*name has attribute *AttrName and value *AttrValue");
  }
}
INPUT *CollName="$/renci/home/rods"
OUTPUT ruleExecOut
```

This lists all of the user-defined metadata values for all of the files in the named collection

iRODS Data Organization



iRODS Client interfaces

- iRODS native client support
  - iCommands: UNIX like command line interface
  - iRODS Explorer for Windows
  - iRODS web browser
  - iDrop: client-side transfer and data management application
- iRODS API
  - Jargon – Java API
  - Prods – PHP API

iRODS Clients

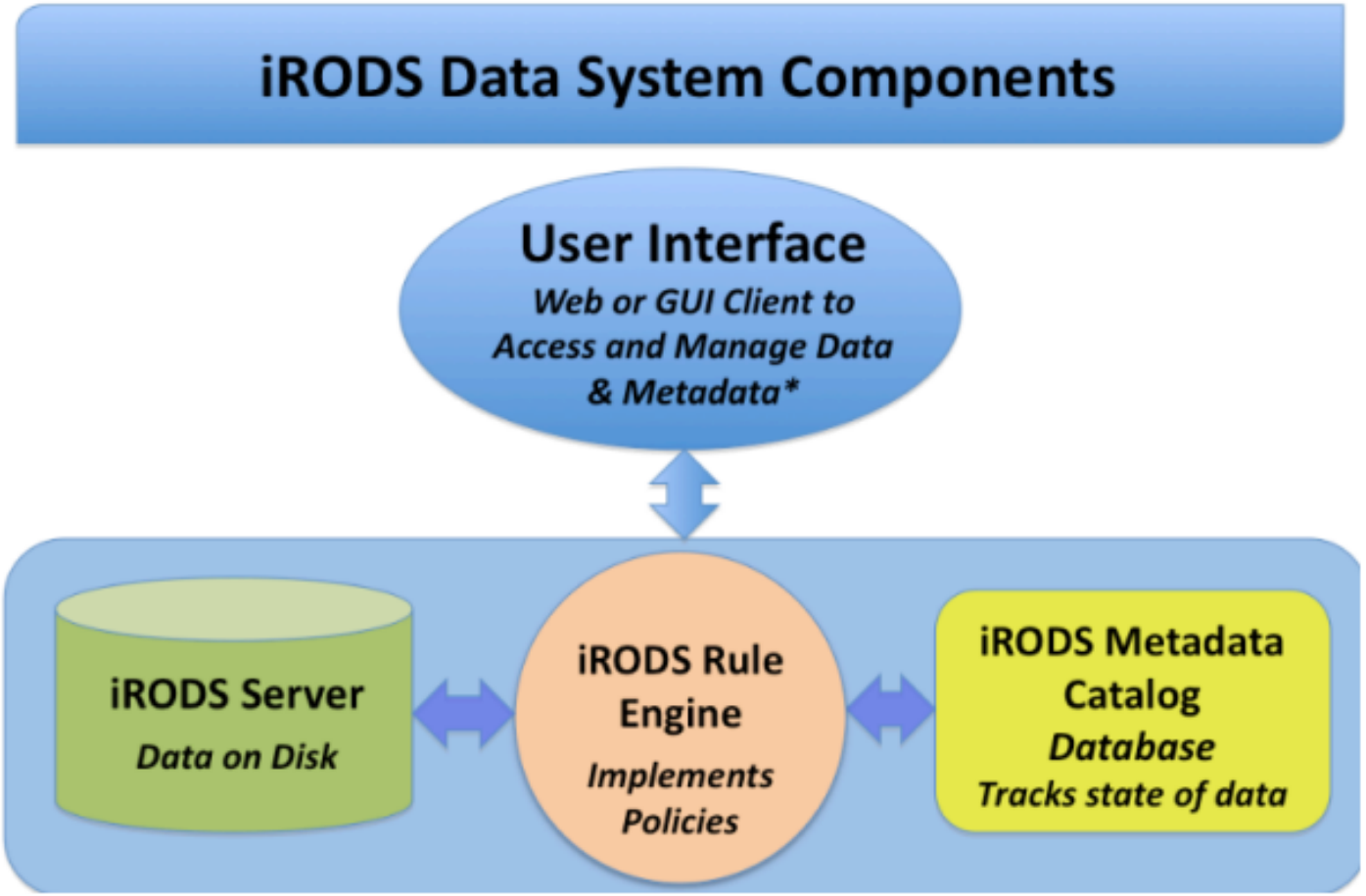
Many are community-supported

API	Client	Developer	Language
Browser			
	DCAPE	UNC	
	iExplore	RENCI	C++
	JUX	IN2P3	Jargon
	Peta Web browser	PetaShare	
	iDrop web browser	DICE	Java
	Davis web interface	ARCS	
	Rich web client	RENCI	
Digital Library			
	Akubra/iRODS	DICE	Jargon
	Dspace	MIT	
	Fedora on Fuse	IN2P3	FUSE
	Fedora/iRODS module	DICE	Jargon
	Islandora	DICE	Jargon
	Curators Workbench	CDR-UNC-CH	Jargon
File System			
	Davis - Webdav	ARCS	Jargon
	Dropbox / **iDrop**	DICE	Jargon
	FUSE	IN2P3, DICE	FUSE
	FUSE optimization	PetaShare	FUSE
	OpenDAP	ARCS	
	PetaFS (Fuse)	Petashare - LSU	
	Petashell (Parrot)	PetaShare	

iRODS Clients (Continued)

Grid	GridFTP - Griffin	ARCS	
	Jsaga	IN2P3	Jargon
	Parrot	UND	
	SRM	Academia Sinica	
	Saga	KEK	
I/O Libraries	PRODS - PHP	RENCI	PHP
	C API	DICE	C
	C I/O library	DICE	C
	Fortran	DICE	C
	Eclipse file system	CDR - UNC-CH	Jargon
	Jargon	DICE	Jargon
	Pyrods - Python	SHAMAN	Python
Portal	EnginFrame	NICE / RENCI	Java
	Petashare Portal	LSU	Jargon
Tools	Archive tools-NOAO	NOAO	
	Big Board visualization	RENCI	
	iFile	GA Tech	
	i-commands	DICE	
	Pcommands	PetaShare	
	Resource Monitoring	IN2P3	
	Sync-package	Academica Sinica	
	URSpace	Teldap - Academica Sinica	
Web Service	VOSpace	IVOA	
	Shibboleth	King's College	
Workflows	Kepler - actor	DICE	Jargon
	Stork - interoperability	LSU	
	Workflow Virtualization	LSU	
	Taverna - actor	RENCI	

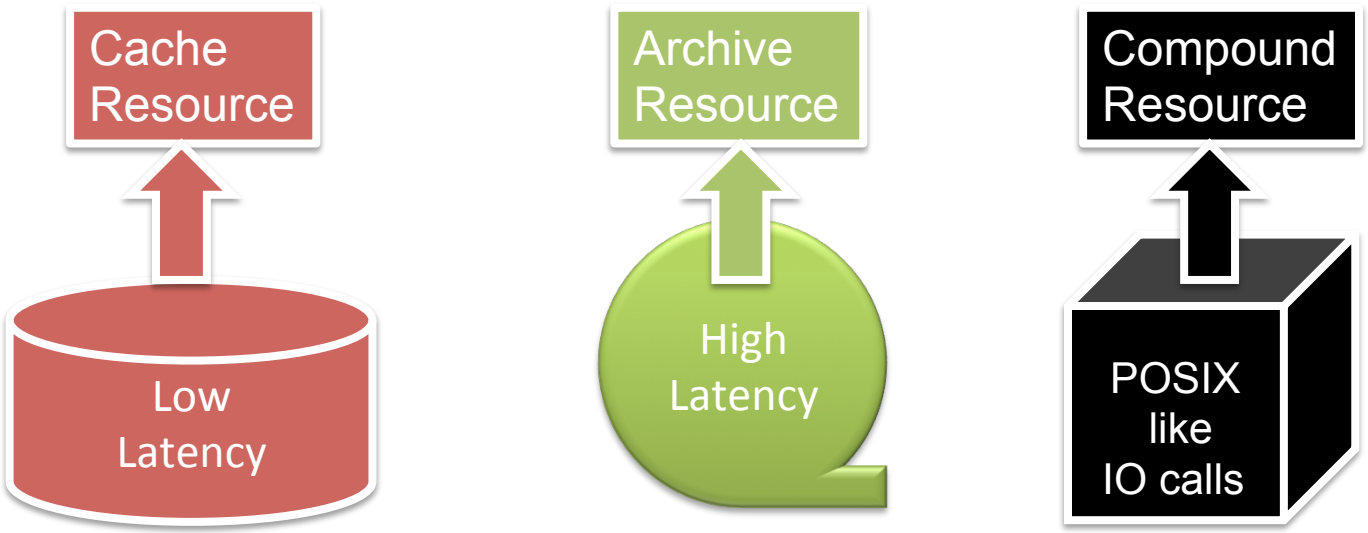
iRODS Scalable Architecture



iRODS Resources

- (Storage) Resource is a Software or Hardware system that stores data
- A resource is a logical mapping of a resource name to a number of physical attributes of a resource
- 3 Resource classes:
  - Cache, Archival and Compound
- Integration with existing Data Services via Mounted collections

iRODS Resource Classes



Compound Resource

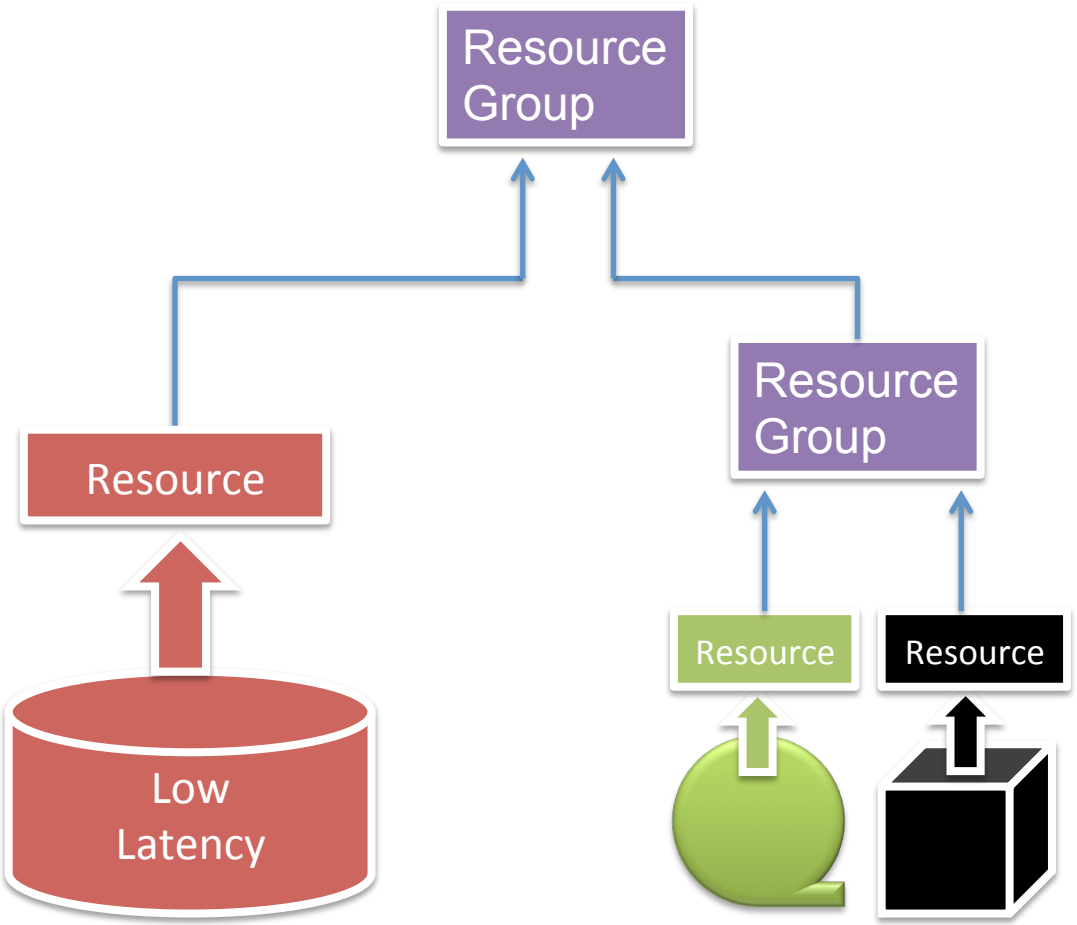
- POSIX like type calls:

syncToArch(), stageToCache(), mkdir(), chmod(),rm(),mv()

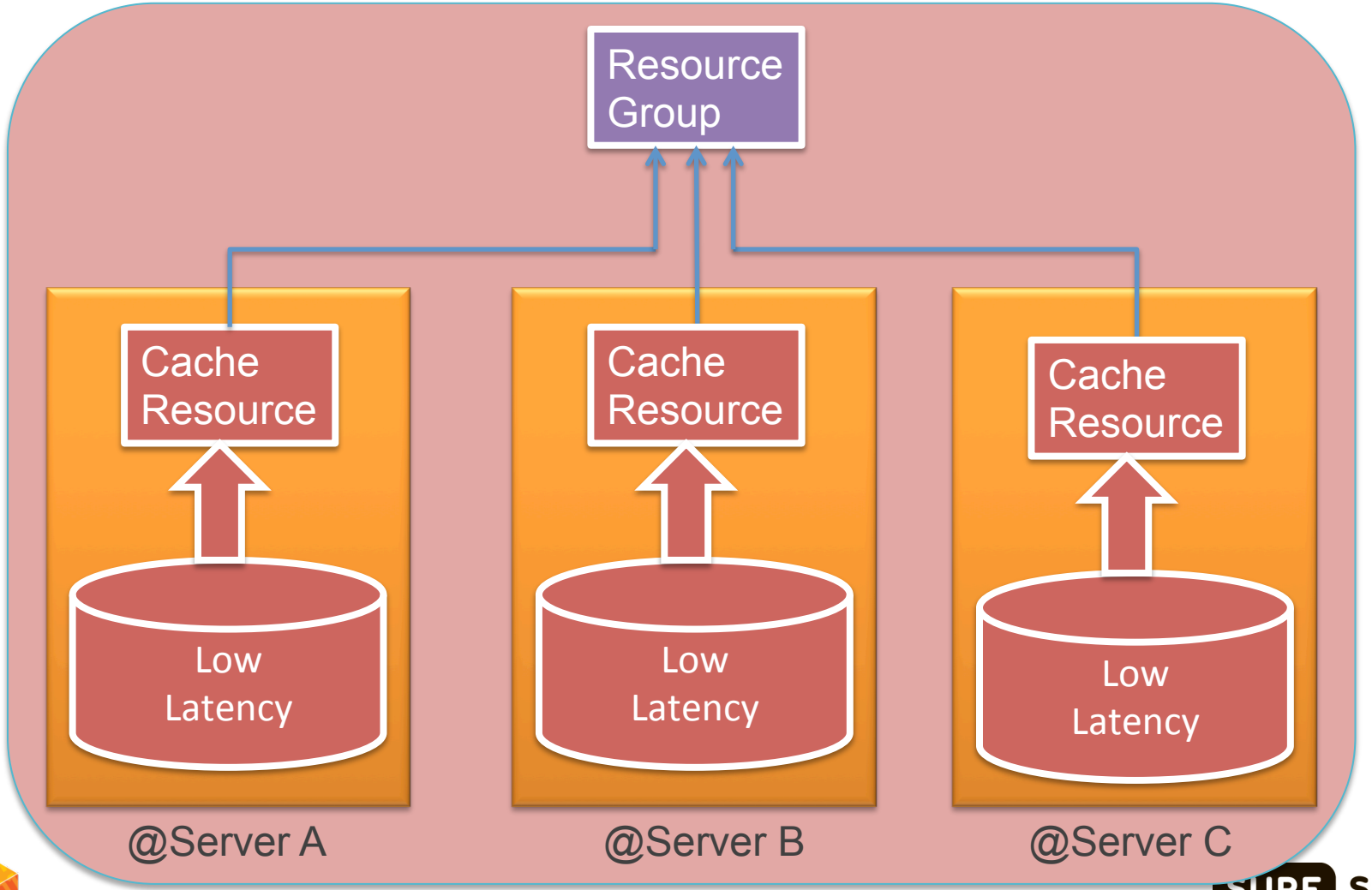
- Standard modules for: Unix FS, Windows FS, HPSS, S3, FTP, Database
- Universal MSS (Mass Storage Service) Driver

```
grep "()" /opt/iRODS/iRODS/server/bin/cmd/univMSSInterface.sh -A5
syncToArch () {
        # <your command or script to copy from cache to MSS> $1 $2
        # e.g: /usr/local/bin/rfcp $1 rfioServerFoo:$2
        /bin/cp $1 $2
        return
}
--
stageToCache () {
        # <your command to stage from MSS to cache> $1 $2
        # e.g: /usr/local/bin/rfcp rfioServerFoo:$1 $2
        /bin/cp $1 $2
        return
}
--
mkdir () {
        # <your command to make a directory in the MSS> $1
        # e.g.: /usr/local/bin/rfmkdir -p rfioServerFoo:$1
        /bin/mkdir -p $1
        return
}
--
chmod () {
        # <your command to modify ACL> $1 $2
        # e.g: /usr/local/bin/rfchmod $2 rfioServerFoo:$1
        /bin/chmod $2 $1
        return
}
--
rm () {
        # <your command to remove a file from the MSS> $1
        # e.g: /usr/local/bin/rfrm rfioServerFoo:$1
        /bin/rm $1
        return
}
--
mv () {
       # <your command to rename a file in the MSS> $1 $2
       # e.g: /usr/local/bin/rfrename rfioServerFoo:$1 rfioServerFoo:$2
       /bin/mv $1 $2
       return
}
```

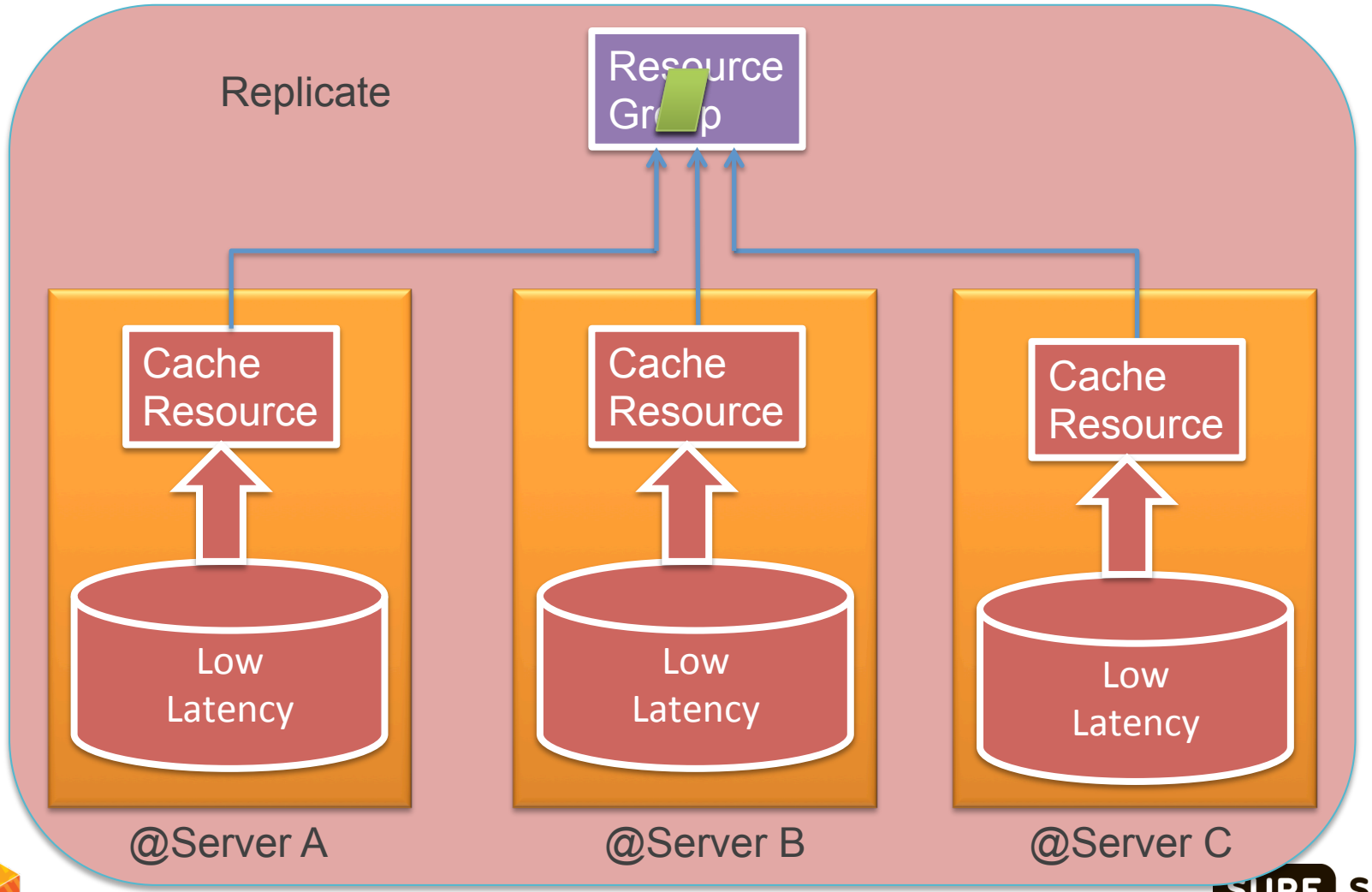
iRODS Resource Groups



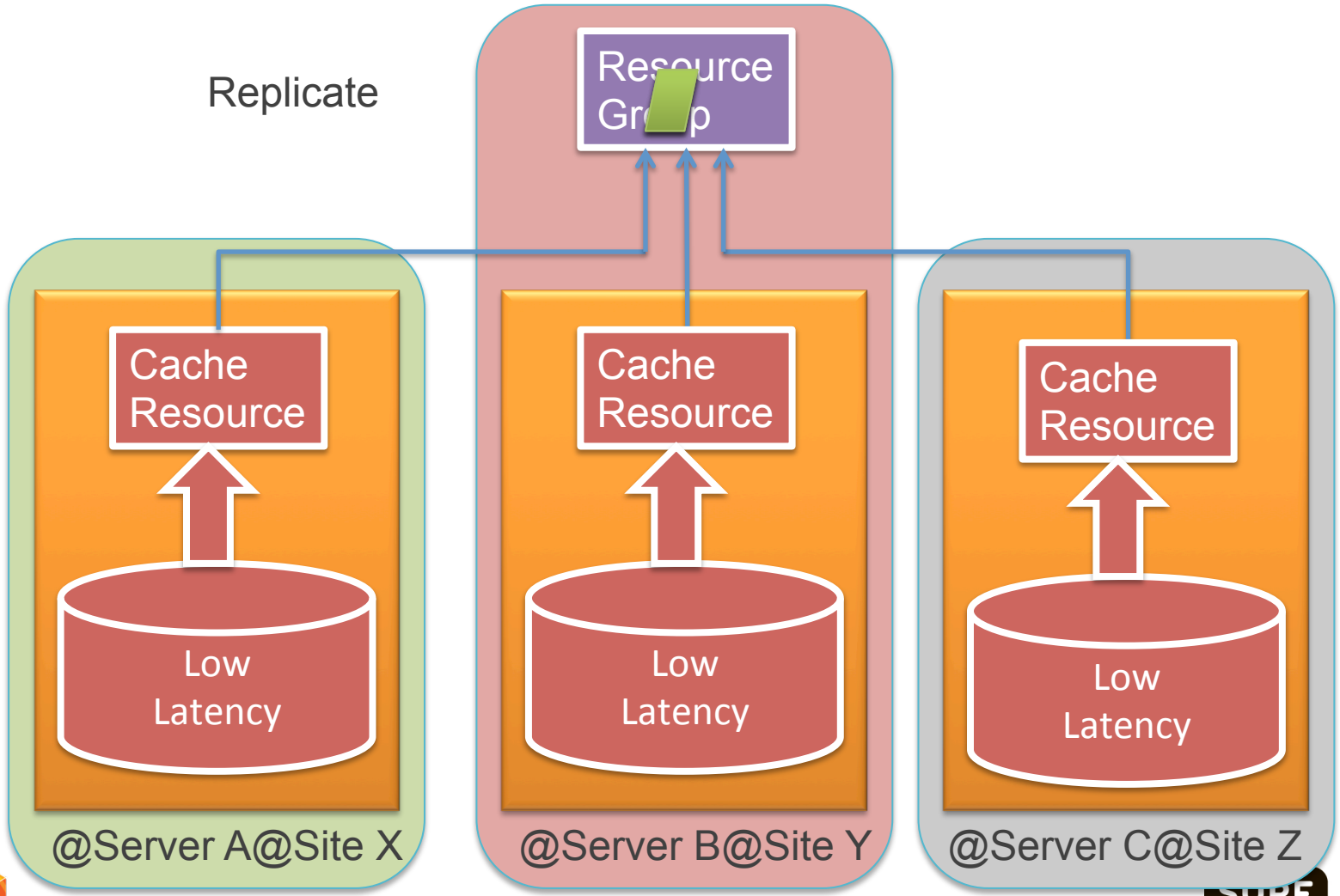
iRODS Resource Groups



iRODS Resource Groups



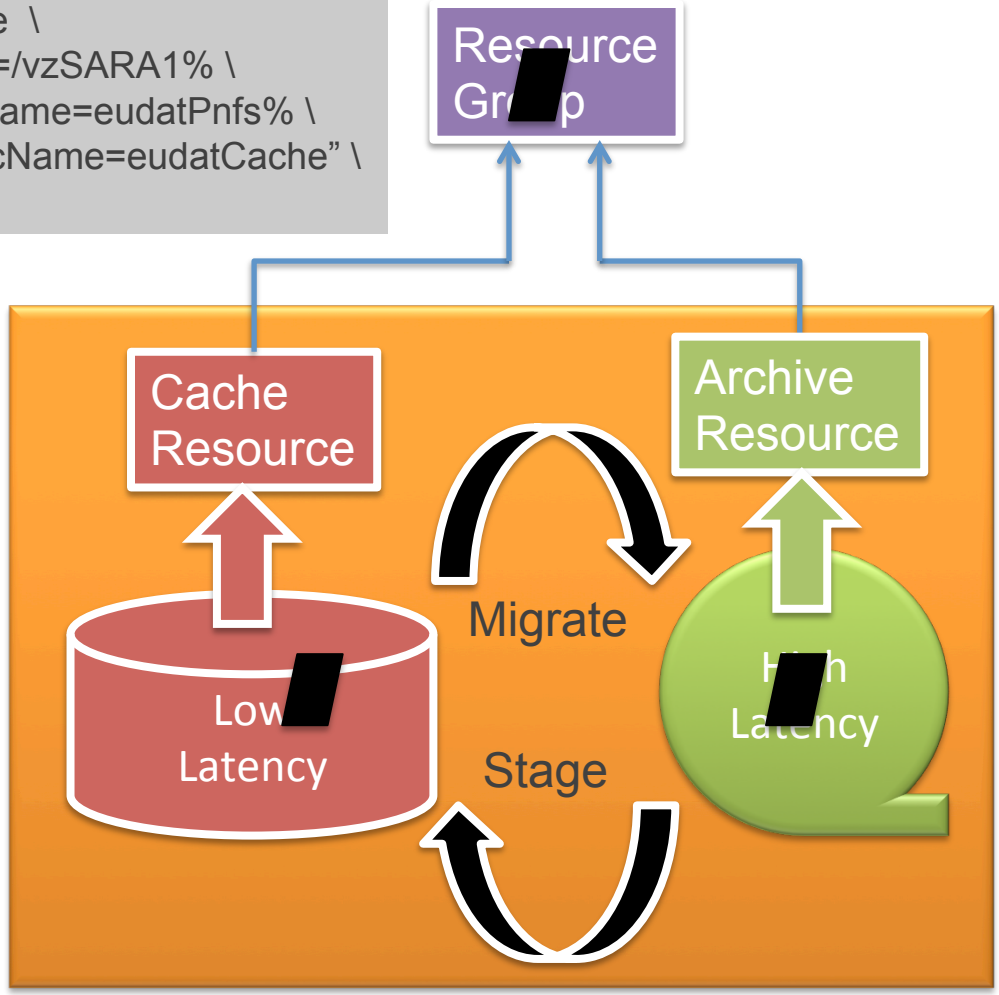
iRODS Resource Groups



iRODS Resource Groups

Implemented as a delayed rule:

```
irule replicateDiskCache  \
            "*Collection=/vzSARA1% \
            *PnfsRescName=eudatPnfs% \
            *CacheRescName=eudatCache" \
ruleExecOut
```



iRODS Mounted Collection

- Mount existing directory as a collection into iRODS
- IRODS works as an Proxy

Pro's

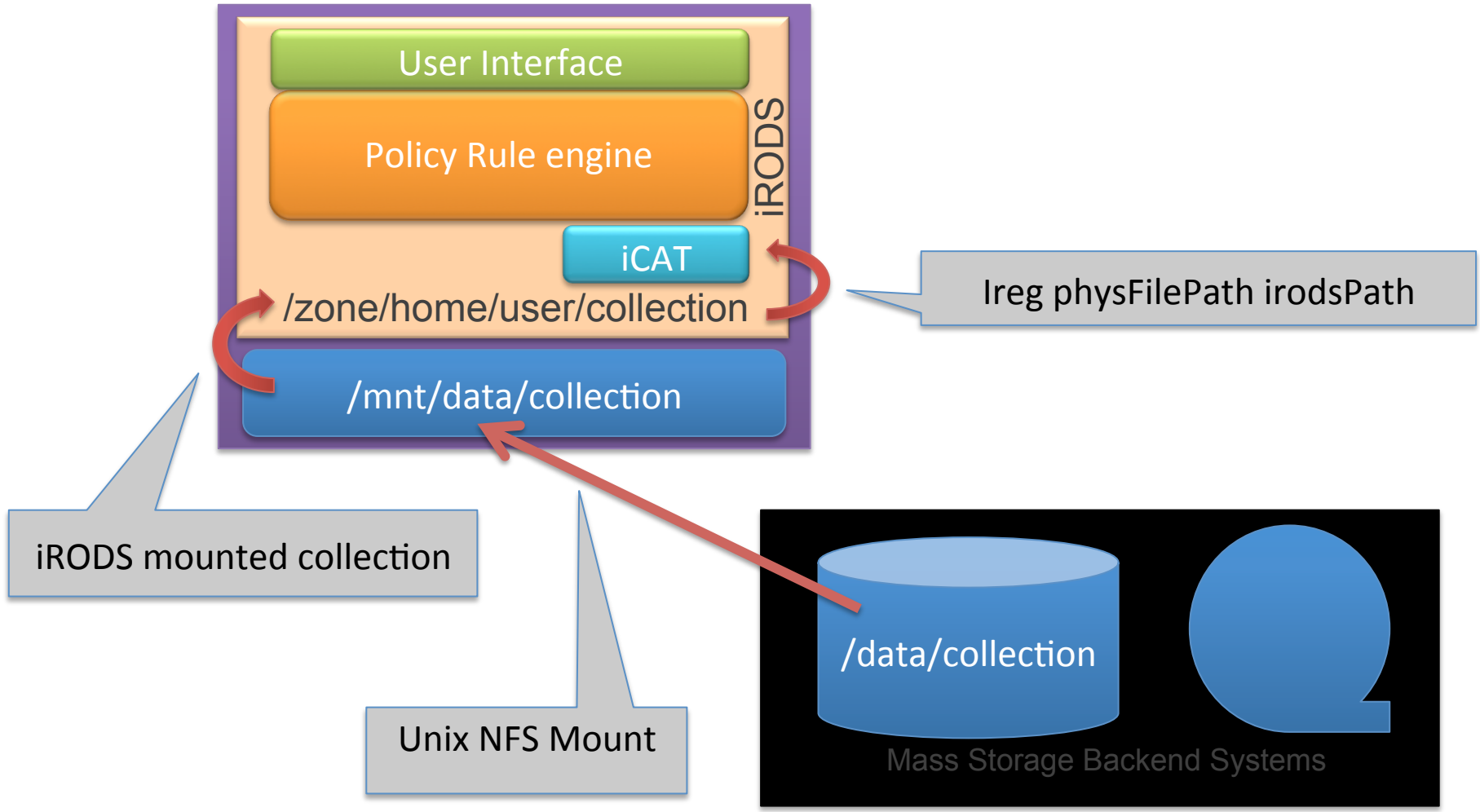
- Labour extensive way to integrate existing data collections
- Data can be modified via other means

Con's

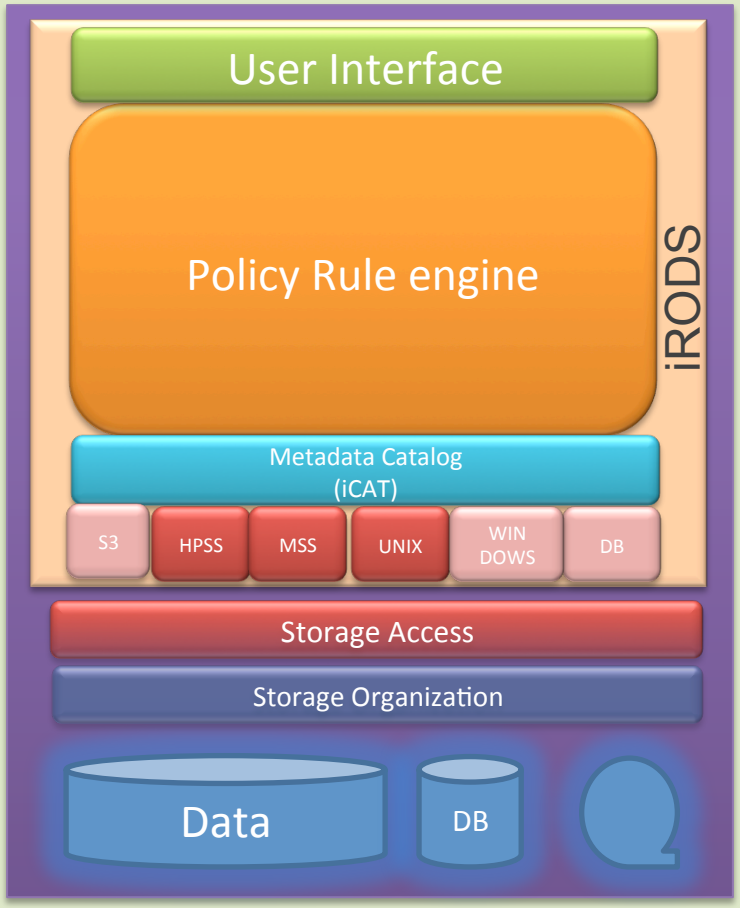
- No metadata is stored in the iCAT database
- Not aware of users
- Checksums are calculated on the fly

iReg to bridge the gap

iRODS Mounted Collection

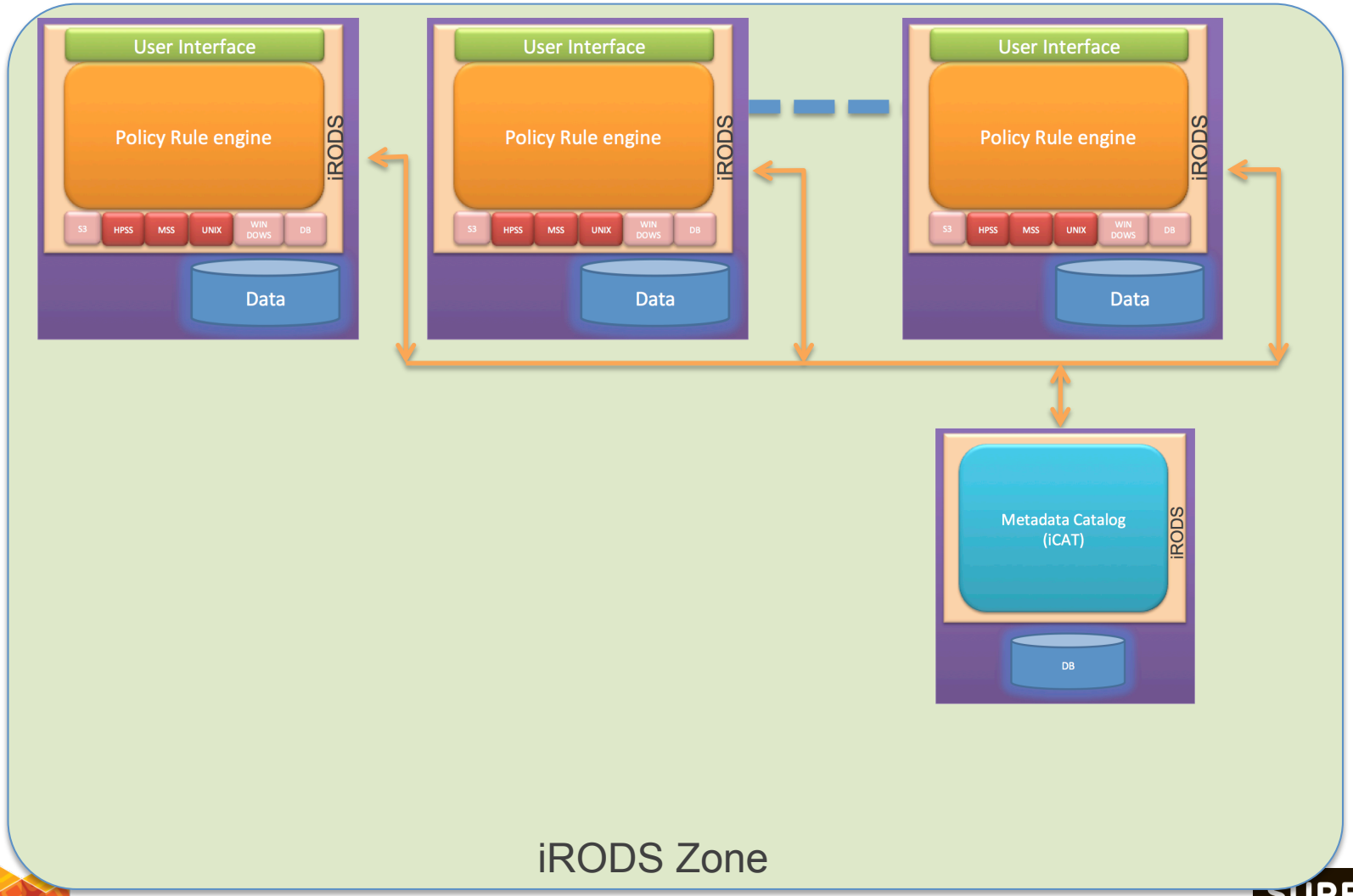


iRODS Scalable Architecture

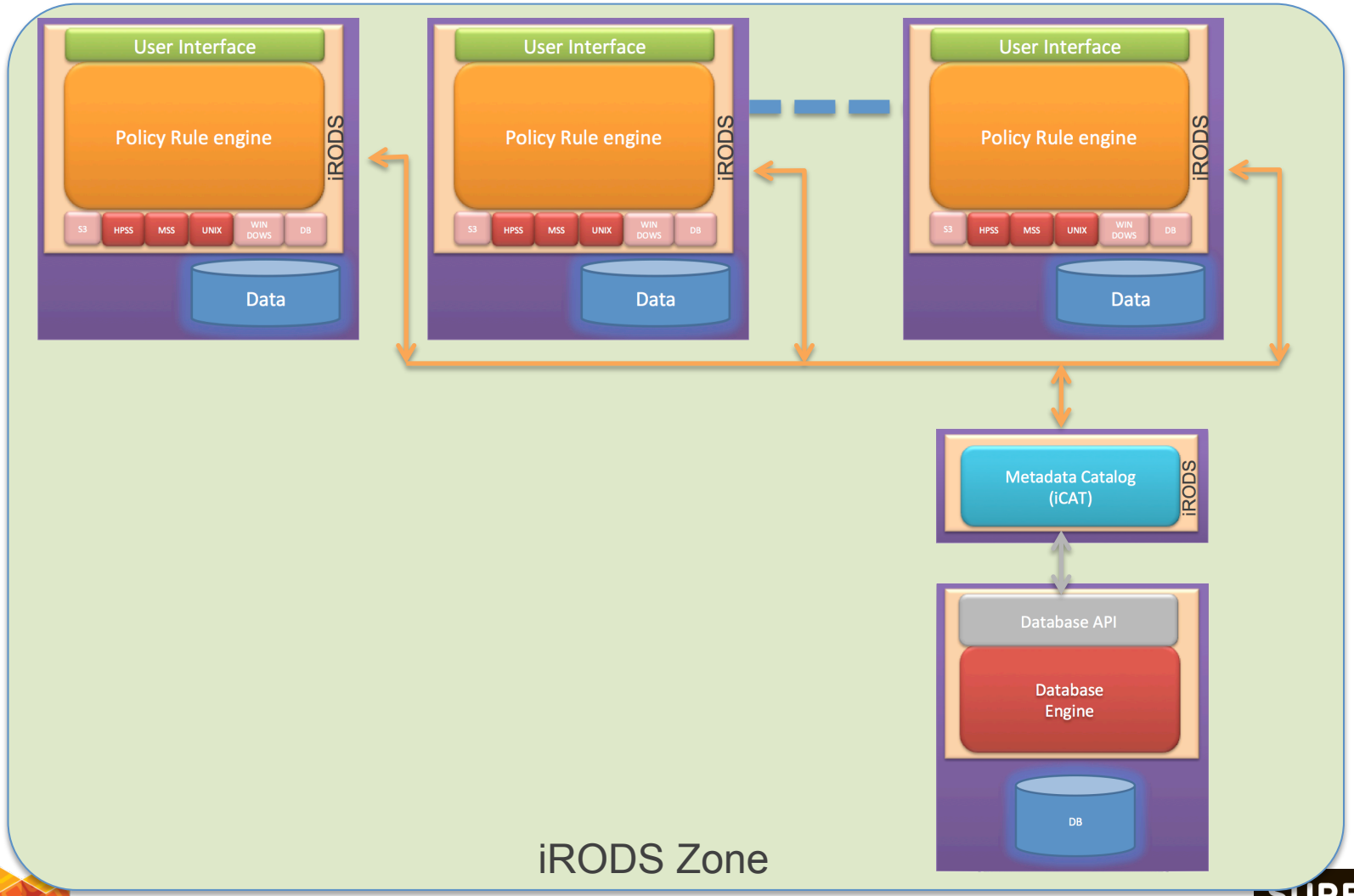


iRODS Zone

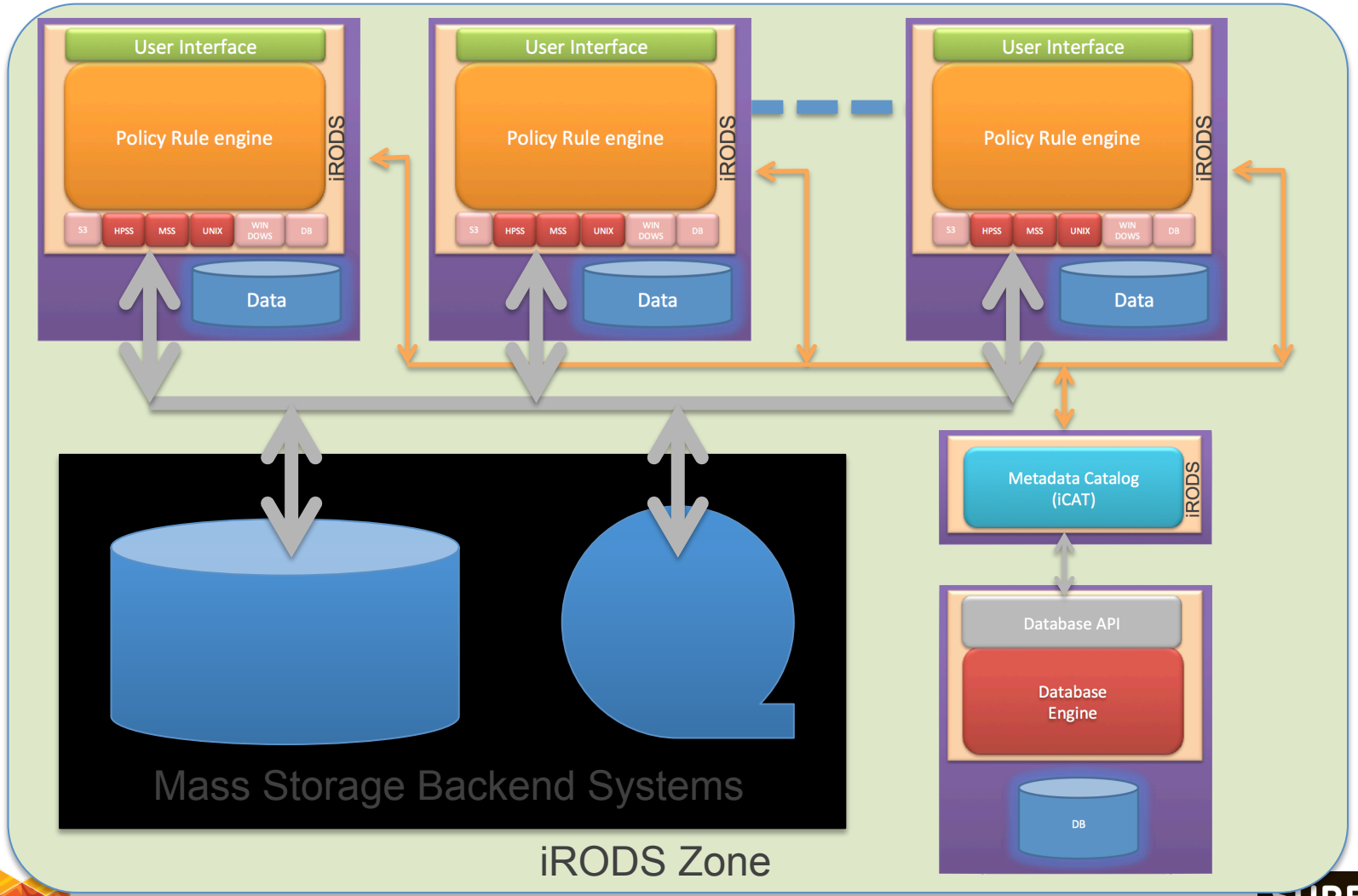
iRODS Scalable Architecture



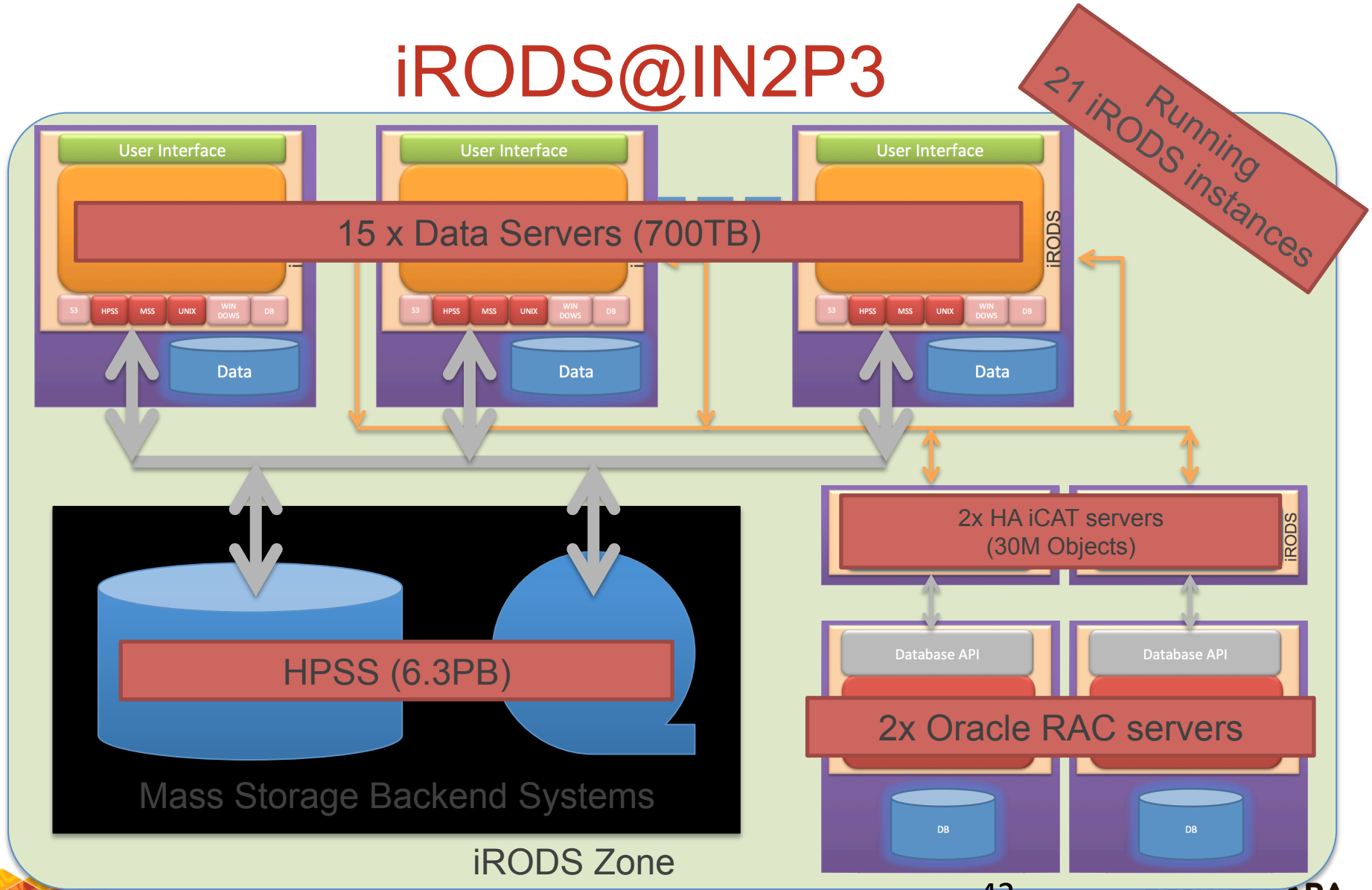
iRODS Scalable Architecture



iRODS Scalable Architecture

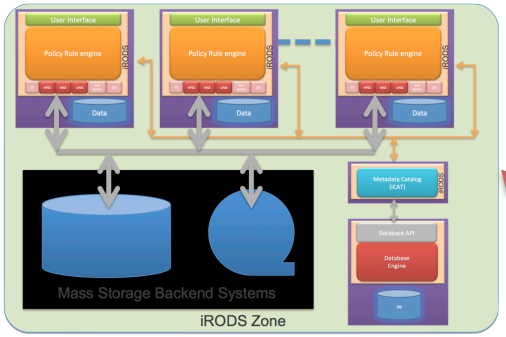


iRODS@IN2P3

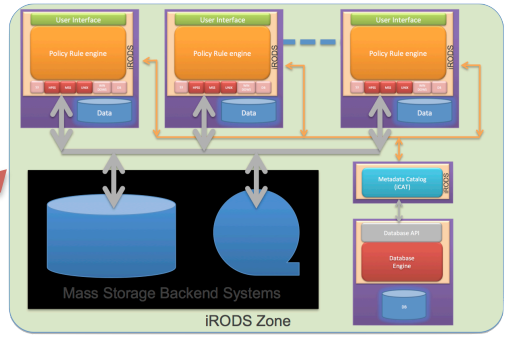

Running
21 iRODS instances
User Interface
User Interface
User Interface
15 x Data Servers (700TB)
iRODS
S3
HPSS
MSS
UNIX
WIN DOWS
DB
Data
Data
Data
2x HA iCAT servers
(30M Objects)
iRODS
HPSS (6.3PB)
Mass Storage Backend Systems
Database API
Database API
2x Oracle RAC servers
DB
DB
iRODS Zone

iadmin mkzone ZoneB remote irods.zoneb.org:1247
iadmin mkzone ZoneC remote irods.zonec.org:1247

iRODS Federation

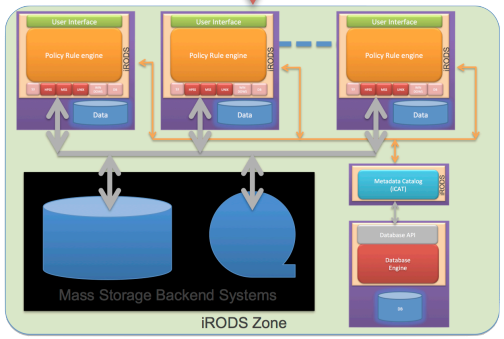


ZoneA



ZoneB

/ZoneA/Collection/DataObject
/ZoneB/Collection/DataObject
/ZoneC/Collection/DataObject



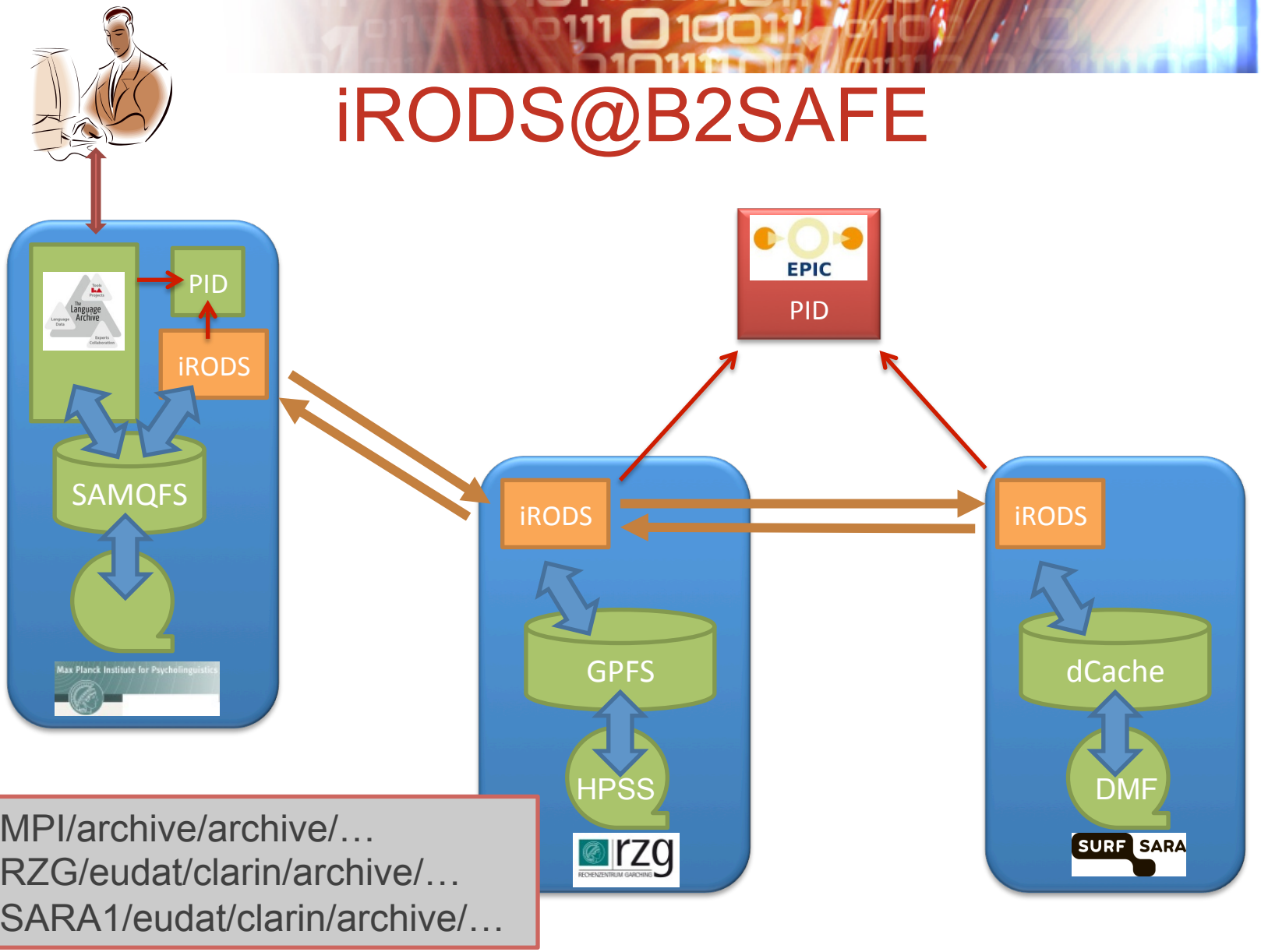
ZoneC

- Independent Administrative domains
- Users can access shared data and metadata in other zones (user#zone)
- No passwords needed, authenticated to local zone

What makes iRODS suitable for EUDAT

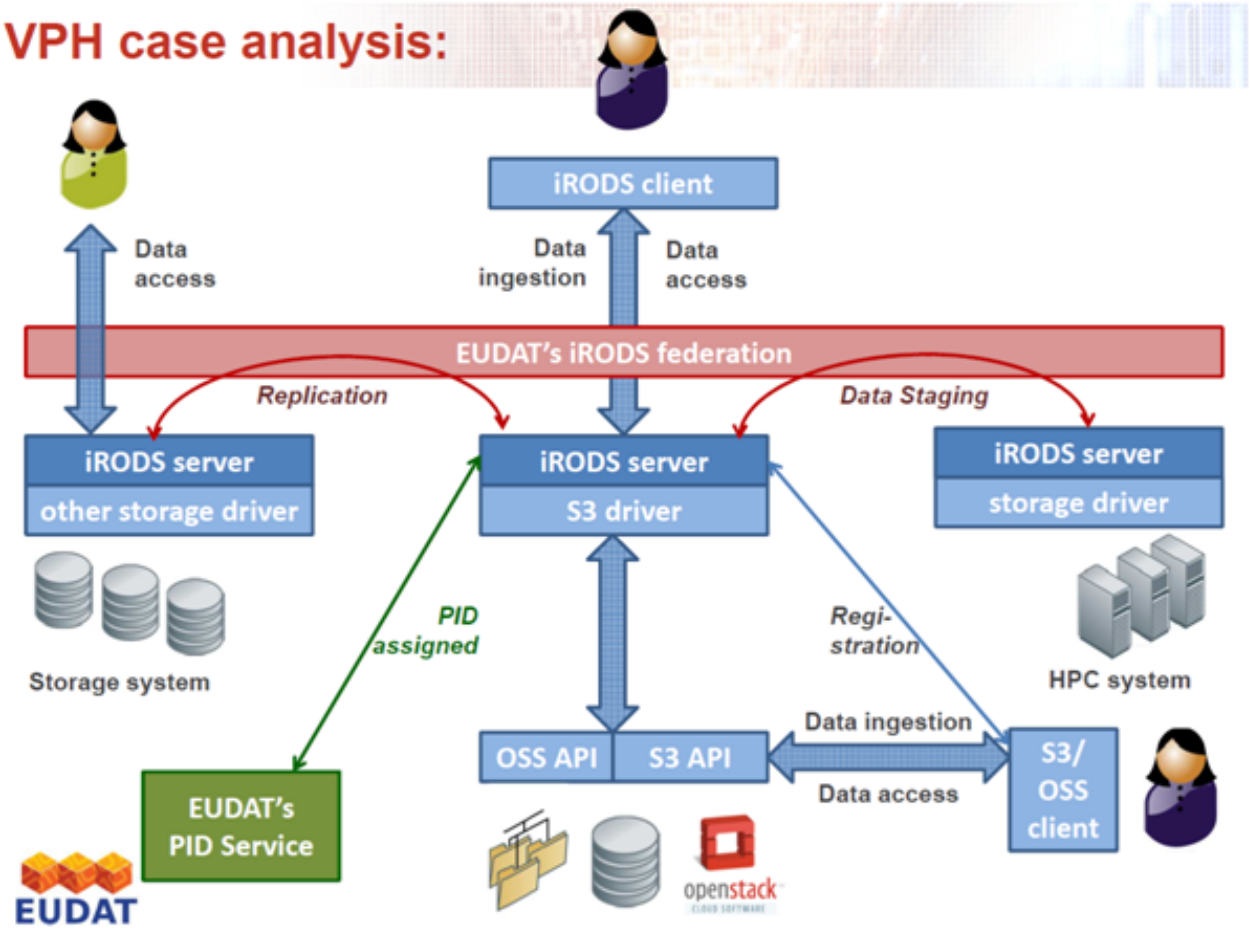
- Data management is policy drive by user and system level rules
  - EUDAT replication policies
- Highly extensible via user defined rules and micro services
  - EUDAT Replication and EPIC PID micro service
- Scalable architecture, from a single server to large scale storage clusters
- Integration with existing data repositories, MPI-PL “The Language Archive”, EPOS, ENES, VPH, INCF, and mass storage systems (HPSS, dCache/DMF, TSM, S3, …)
- iRODS is open source: BSD license
  - DICE, CC-IN2P3, EU SHAMAN, Australian ARCS, UK e-Science, King’s College and others

iRODS@B2SAFE

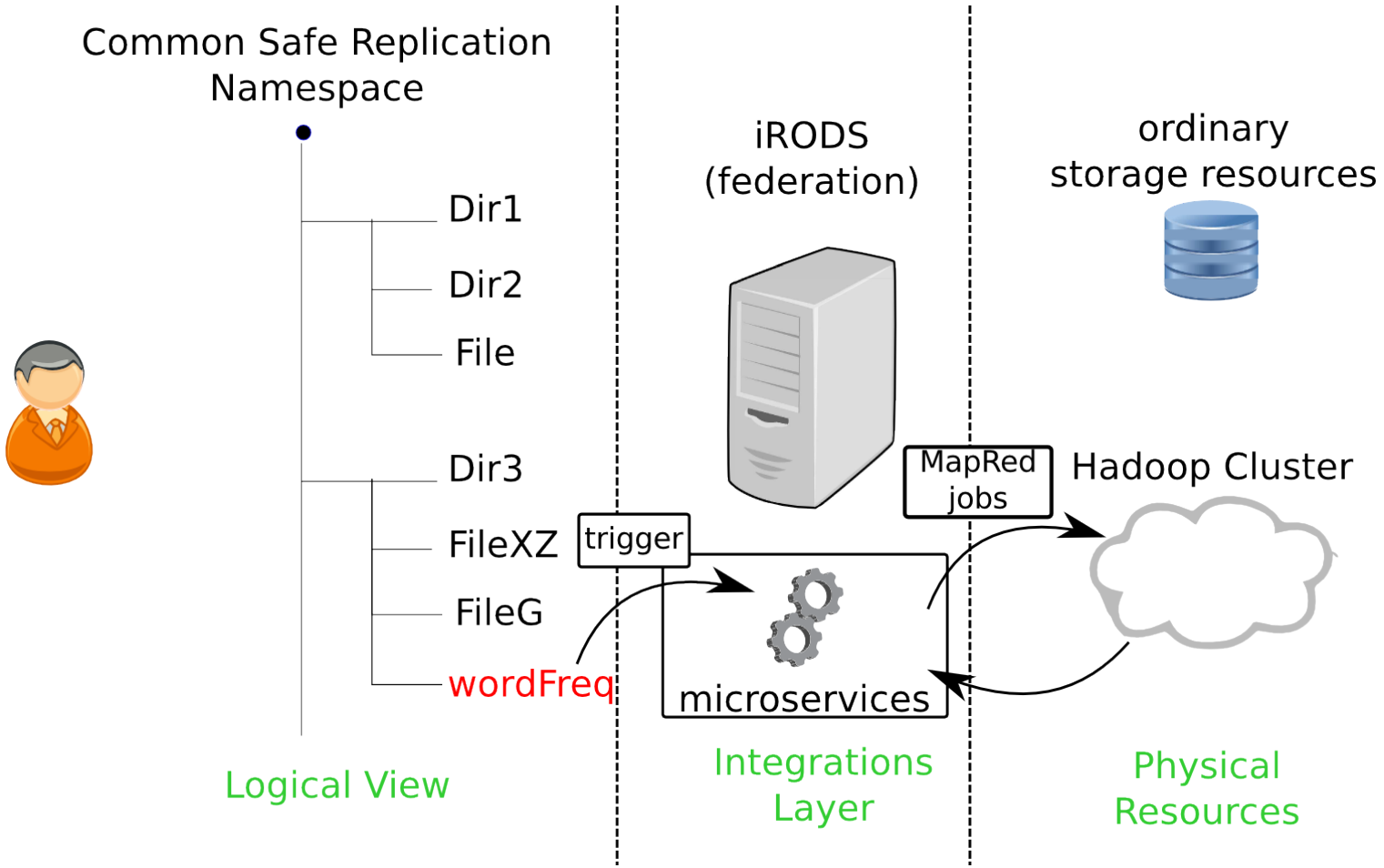


/vzMPI/archive/archive/…
/vzRZG/eudat/clarin/archive/…
/vzSARA1/eudat/clarin/archive/…

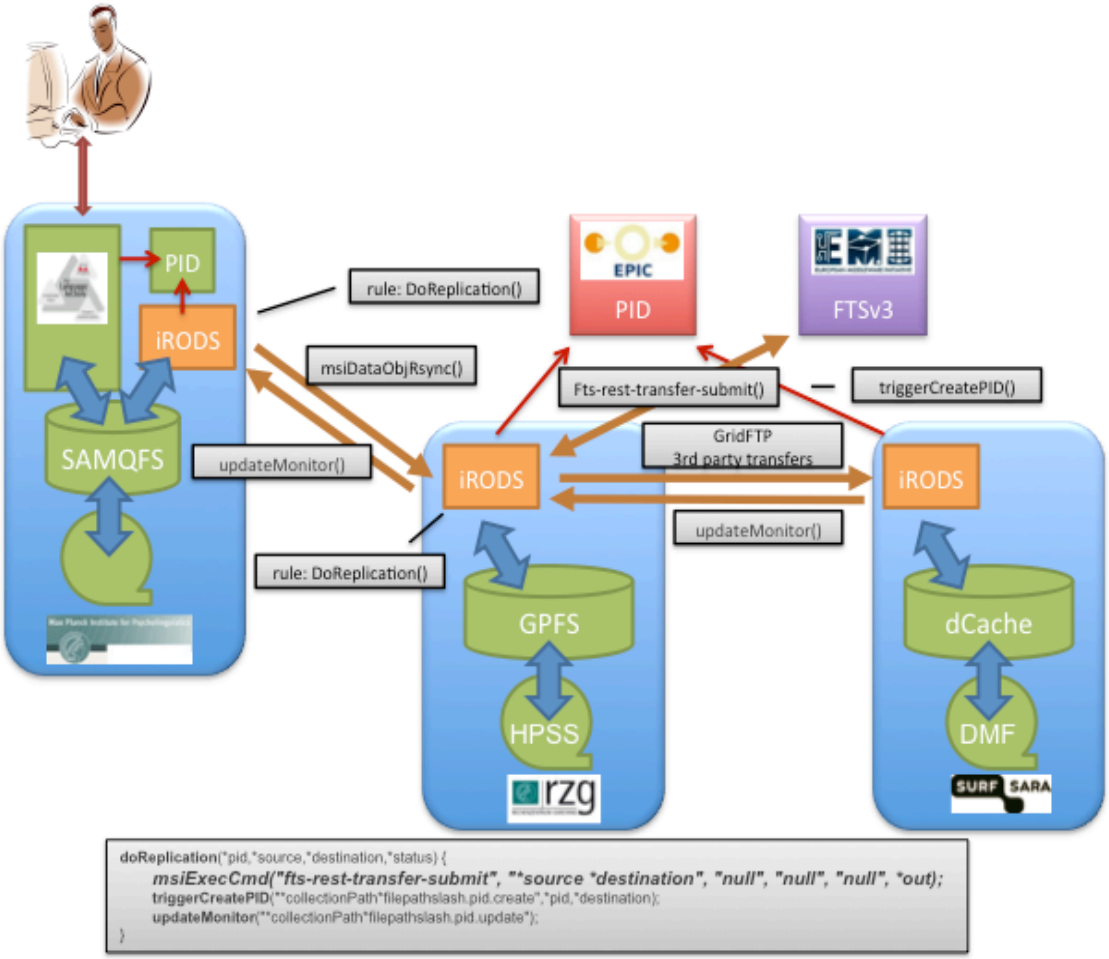
B2SAFE Integration with S3 Storage



B2SAFE Integration with Hadoop



B2SAFE with EMI File Transfer Service



iRODS Technologies: Community and Enterprise Editions

- Both are open-source code (BSD)
- Community edition – DICE; Enterprise edition – RENCI
- Enterprise edition: extensive testing on supported platforms, enhanced modularity with plug-ins, documentation and training, hardening,…
- The iRODS Consortium brings together DICE and RENCI support
- iRODS Version 4.0 – code coming in March 2014
  - A common code core for community and enterprise editions
  - Pluggable architecture
    - microservices
    - resources
    - DBs
- The real distinction between community and enterprise will be contained in plug-in modules.

Ongoing Support for iRODS

Supporting institutions

- Renaissance Computing Institute (*RENCI*) – a research unit of UNC Chapel Hill
- Data Intensive Cyber Environments (*DICE*) – another research center of UNC Chapel Hill, tightly coupled with RENCI; original developers from SDSC and UCSD
- The iRODS Consortium
  - Hosted at RENCI, co-founded with the Max Planck Society
  - Bringing support infrastructure together
  - Bringing iRODS community together – sustainability, support, roadmap

Website
https://www.irods.org/

Documentation
https://www.irods.org/index.php/Documentation

Downloads
https://www.irods.org/index.php/Downloads

iRODS Forum
https://groups.google.com/forum/#!forum/iROD-Chat

Email
irods@irods.org

Bug Tracking
https://www.irods.org/bugzilla/index.cgi

quaestio quaestio go < ? ? ? ?