



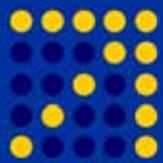
Architecture of the gLite Data Management System

AIFTIMIEI, Doina Cristina

INFN Padova

crisrina.aiftimiei@pd.infn.it





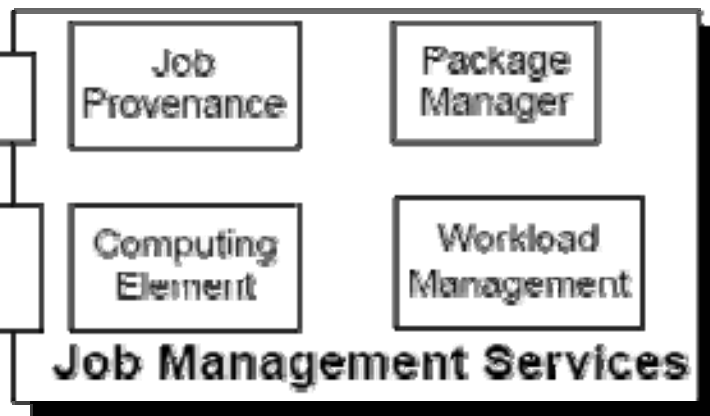
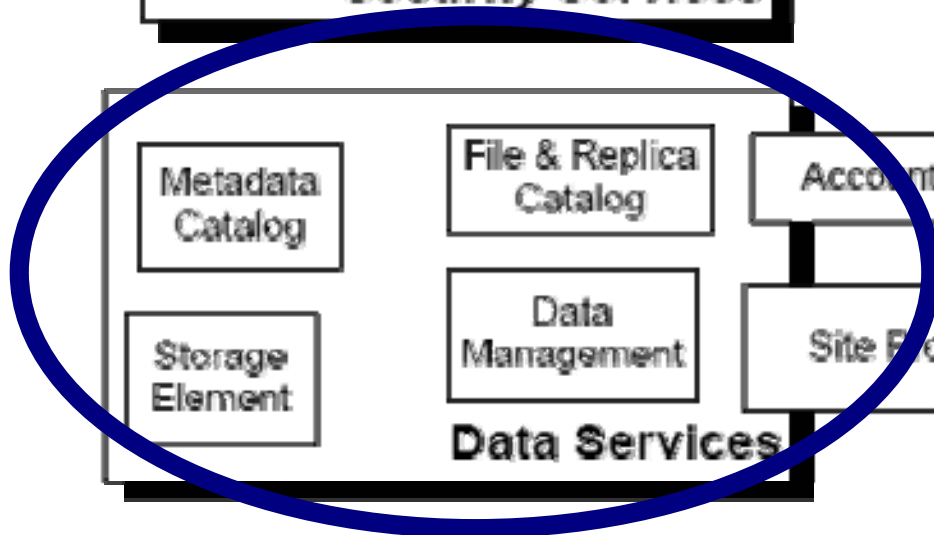
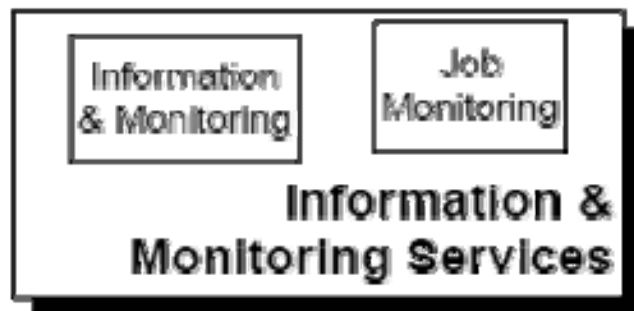
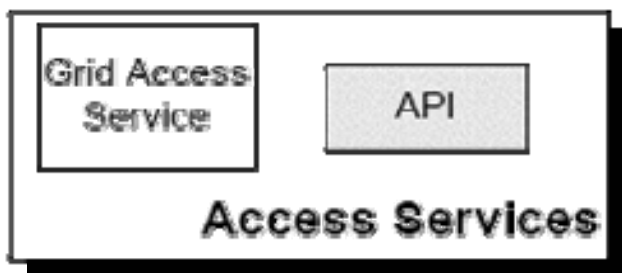
BioinfoGRID

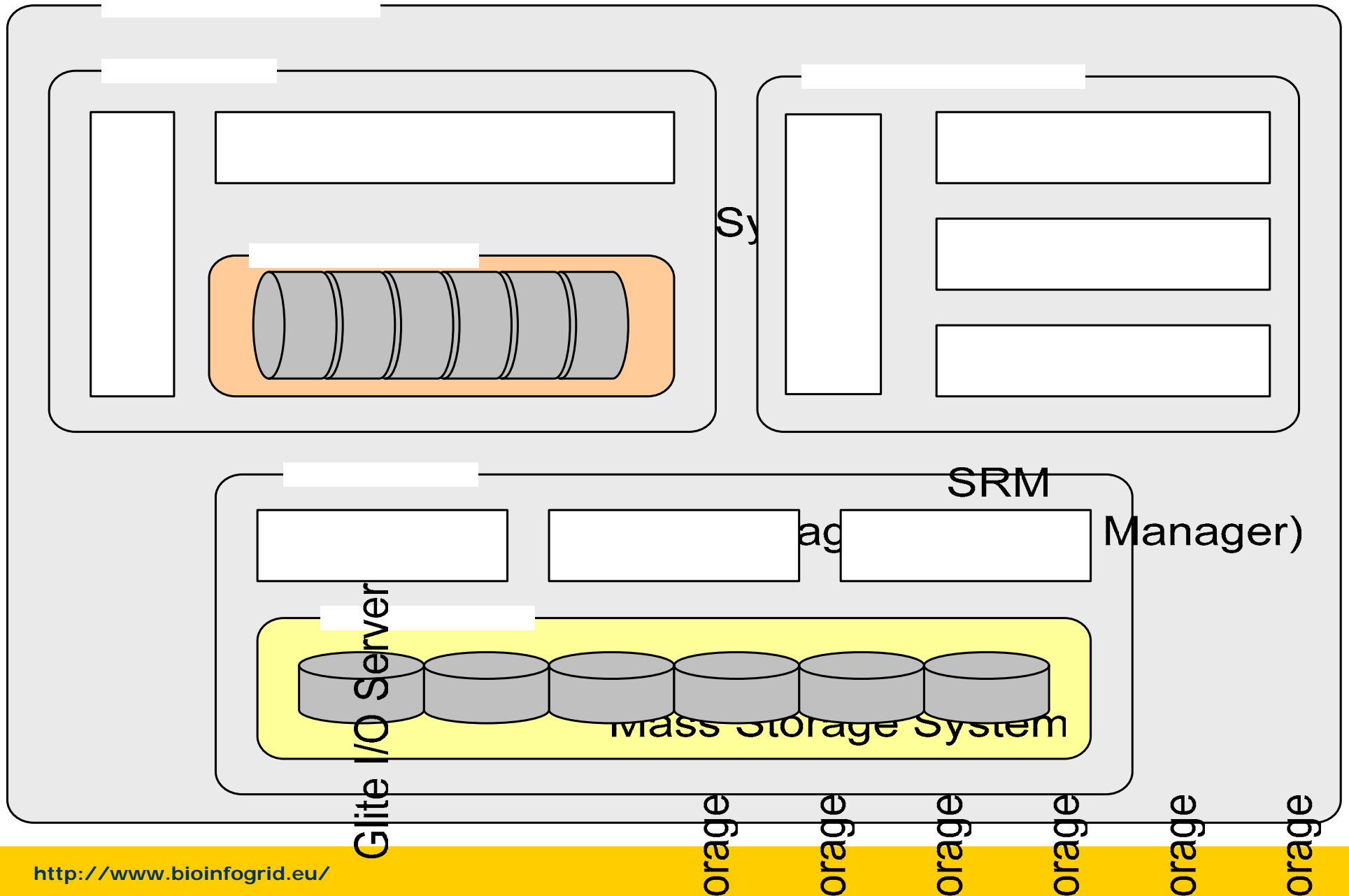
Bioinformatics Grid Application for life science



This presentation is based on previews presentations
from GILDA and EGEE Tutorials.

- Grid Data Management Challenge
- Storage Elements and SRM
- File Catalogs and DM tools
- File Transfer Service
- Metadata Service





- **Heterogeneity**
 - Data are stored on different storage systems using different access technologies
 - **Distribution**
 - Data are stored in different locations – in most cases there is no shared file system or common namespace
 - Data need to be moved between different locations
 - **Data Description**
 - Data are stored as files: need a way to describe files and locate them according to their content
- ***Need common interface to storage resources***
 - ***Storage Resource Manager (SRM)***
 - ***Need to keep track where data is stored***
 - ***File and Replica Catalogs***
 - ***Need scheduled, reliable file transfer***
 - ***File transfer service***
 - ***Need a way to describe file's content and query them***
 - ***Metadata service***

- **Assumptions:**
 - Users and programs produce and require data
 - the lowest granularity of the data is on the file level (we deal with files rather than data objects or tables)
 - Data = files
- **Files:**
 - Mostly, write once, read many
 - Located in Storage Elements (**SEs**)
 - Several replicas of one file in different sites
 - Accessible by Grid users and applications from “anywhere”
 - Locatable by the WMS (data requirements in JDL)
- **Also...**
 - WMS can send (small amounts of) data to/from jobs: Input and Output Sandbox
 - Files may be copied from/to local filesystems (WNs, UIs) to the Grid (SEs)



- The **Storage Element** is the service which allow a user or an application to store data for future retrieval
- Manage local storage (disks) and interface to Mass Storage Systems(tapes) like
 - HPSS, CASTOR, DiskeXtender (UNITREE), ...
- Be able to manage different storage systems uniformly and transparently for the user (providing an SRM interface)
- Support basic file transfer protocols
 - GridFTP mandatory
 - Others if available (https, ftp, etc)
- Support a native I/O (remote file) access protocol
 - POSIX (like) I/O client library for direct access of data

She is running a job which needs:
Data for physics event reconstruction
Simulated Data
Some data analysis files
She will write files remotely too

They are at CERN
In dCache

They are at Fermilab
In a disk array

They are at Nikhef
in a classic SE



dCache

Own system, own protocols
and parameters

gLite DPM

Independent system from
dCache or Castor

Castor

No connection with
dCache or DPM

SRM

I talk to them on your
behalf
I will even allocate space
for your files
And I will use transfer
protocols to send your files
there

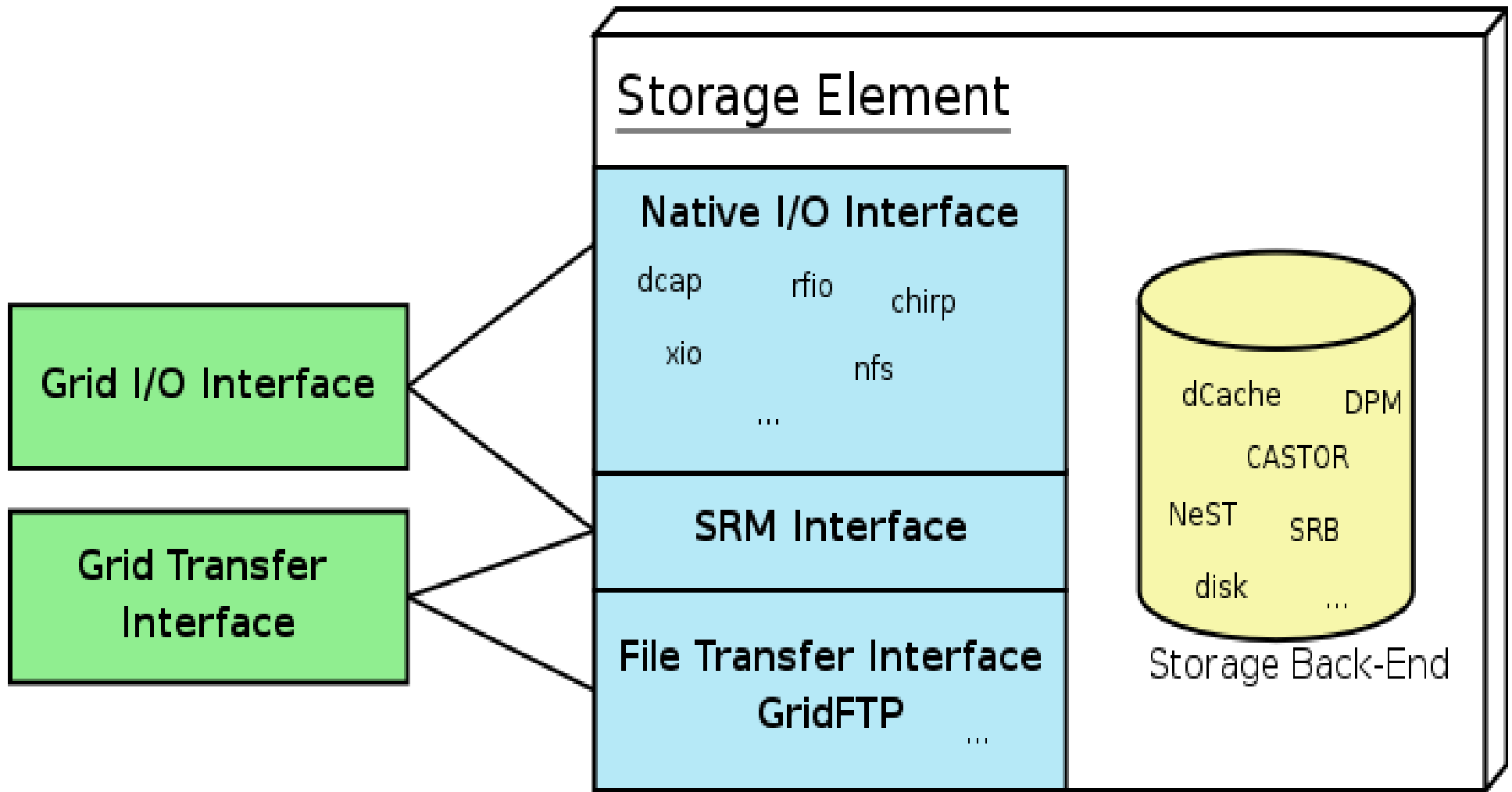


- **Data are stored on disk pool servers or Mass Storage Systems**
- **storage resource management needs to take into account**
 - Transparent access to files (migration to/from disk pool)
 - File pinning
 - Space reservation
 - File status notification
 - Life time management
- **The SRM (Storage Resource Manager) takes care of all these details**
 - The SRM is a single interface that takes care of local storage interaction and provides a Grid interface to the outside world
- **In gLite, interactions with the SRM is hidden by higher level services (DM tools and APIs)**

- gLite 3.0 data access protocols:
 - File Transfer: **GSIFTP** (GridFTP)
 - File I/O (Remote File access):
 - **gsidcap**
 - **insecure RFIO**
 - **secured RFIO (gsirfio)**
- **Classic SE:**
 - GridFTP server
 - Insecure RFIO daemon (rfiod) – only LAN limited file access
 - Single disk or disk array
 - No quota management
 - Does not support the SRM interf

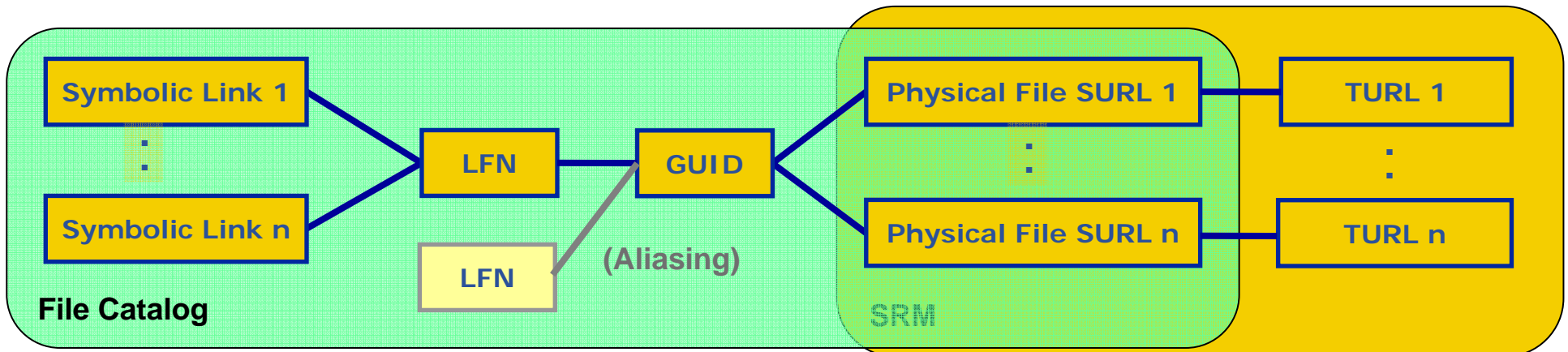


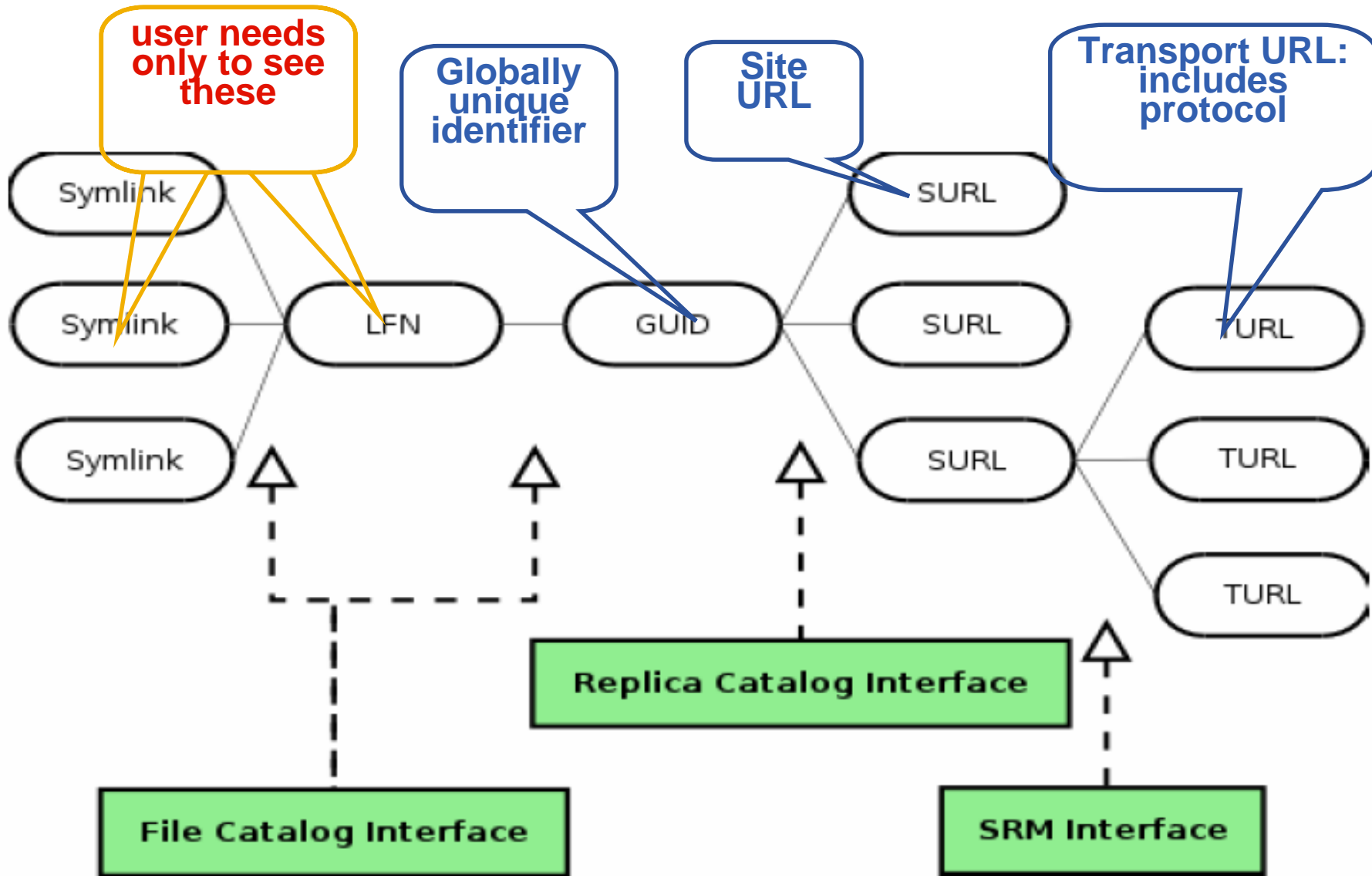
- **Mass Storage Systems (Castor)**
 - Files migrated between front-end disk and back-end tape storage hierarchies
 - GridFTP server
 - Insecure RFIO (Castor)
 - Provide a SRM interface with all the benefits
- **Disk pool managers (dCache and gLite DPM)**
 - manage distributed storage servers in a centralized way
 - Physical disks or arrays are combined into a common (virtual) file system
 - Disks can be dynamically added to the pool
 - GridFTP server
 - Secure remote access protocols (gsidcap for dCache, gsirfio for DPM)
 - SRM interface

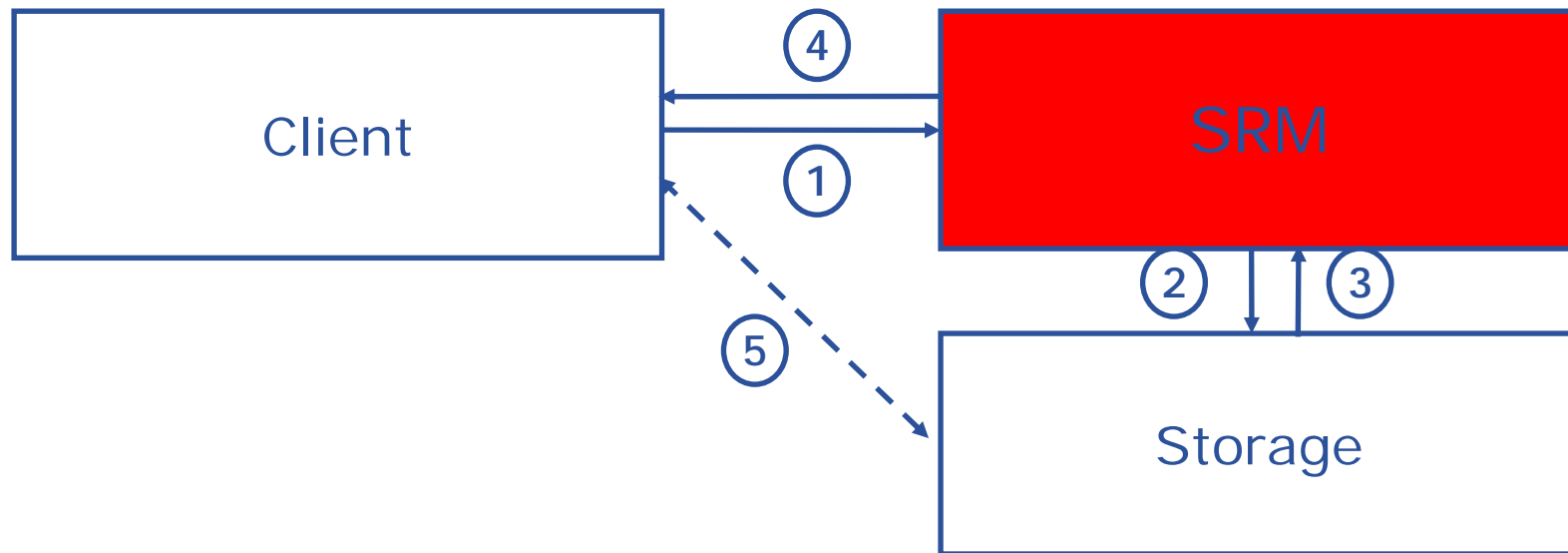




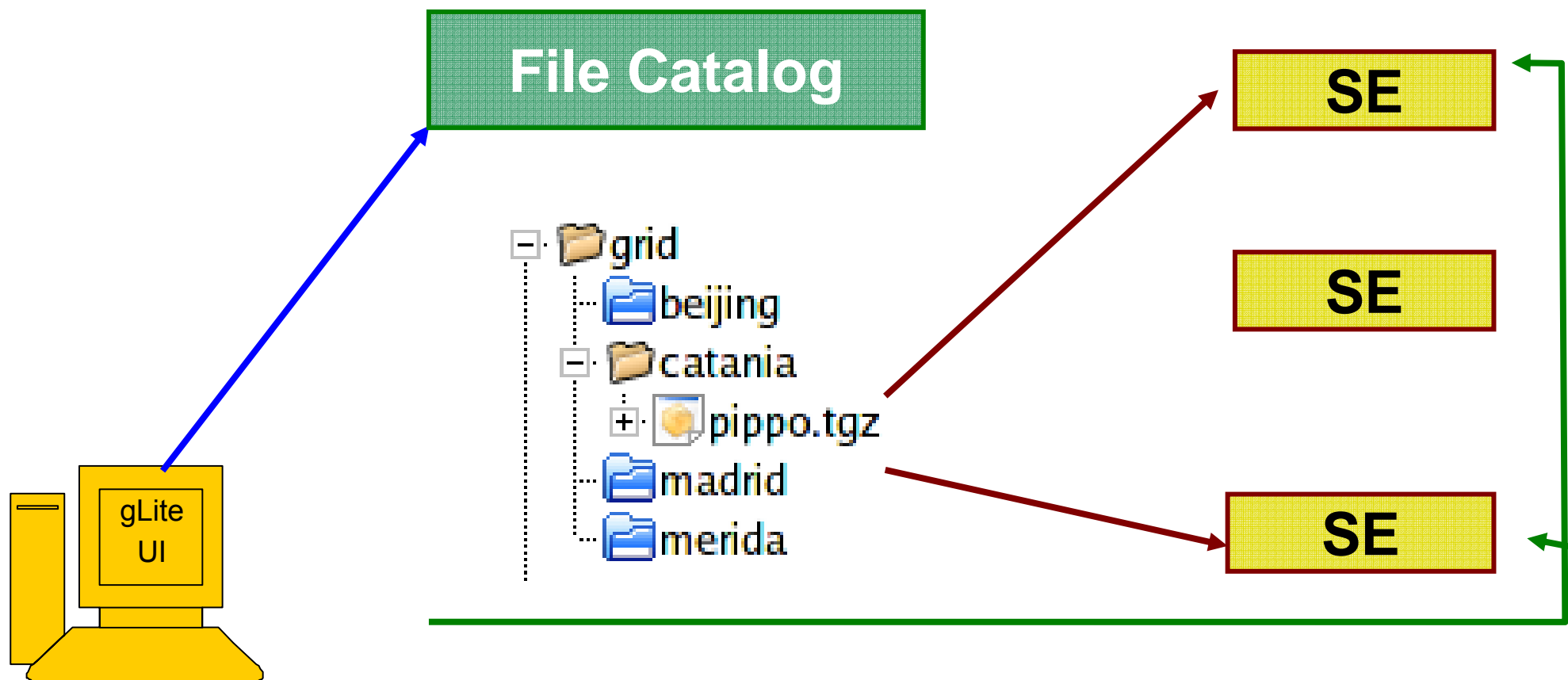
- **Logical File Name (LFN)**
- *An alias created by a user to refer to some item of data, e.g. “lfn:/grid/gilda/20030203/run2/track1”*
- **Globally Unique Identifier (GUID)**
- *A non-human-readable unique identifier for an item of data, e.g. “guid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6”*
- **Site URL (SURL) (or Physical File Name (PFN) or Site FN)**
- *The location of an actual piece of data on a storage system, e.g. “srm://grid009.ct.infn.it/dpm/ct.infn.it/gilda/output10_1” (SRM)
“sfn://lxshare0209.cern.ch/data/alice/ntuples.dat” (Classic SE)*
- **Transport URL (TURL)**
- *Temporary locator of a replica + access protocol: understood by a SE, e.g. “rfio://lxshare0209.cern.ch//data/alice/ntuples.dat”*







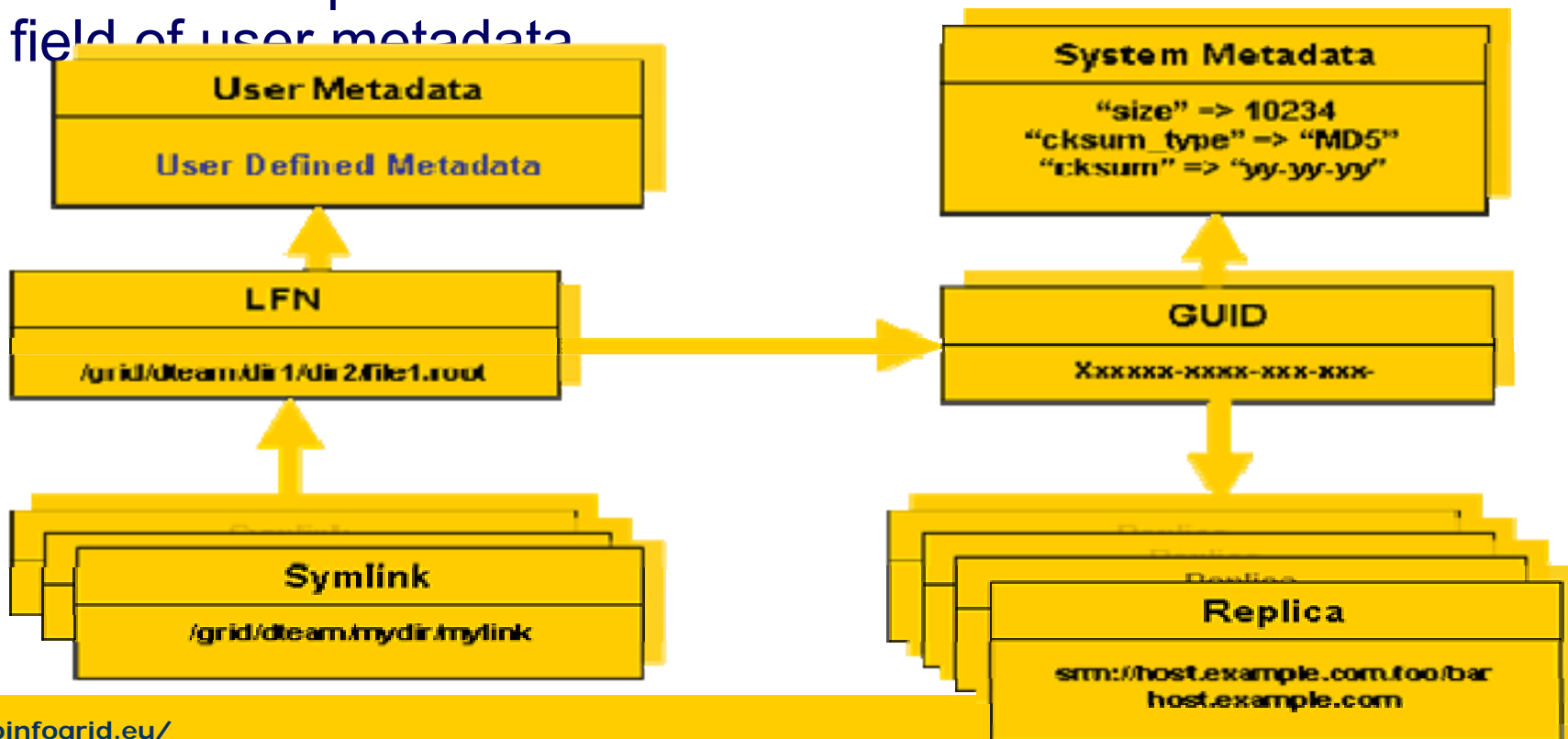
1. The client asks the SRM for the file providing an SURL (Site URL)
2. The SRM asks the storage system to provide the file
3. The storage system notifies the availability of the file and its location
4. The SRM returns a TURL (Transfer URL), i.e. the location from where the file can be accessed
5. The client interacts with the storage using the protocol specified in the TURL



- Users and applications need to locate files (or replicas) on the whole Grid. The File Catalog is the service which allows it and it maintains the mappings between LFNs, GUIDs and SURLS.
- In LCG-2, file cataloguing operations are provided by the LFC (LCG File Catalog).



- It keeps track of the location of copies (replicas) of Grid files
- LFN acts as main key in the database. It has:
 - Symbolic links to it (additional LFNs)
 - Unique Identifier (GUID)
 - System metadata
 - Information on replicas
 - One field of user metadata



The **L**CG **F**ile **C**atalog fixes the performance and scalability problems of EDG (European Data Grid) file catalogs.

Provides

- Bulk operations.
- Cursors for large queries.
- Timeouts and retries for client operations.

Added features :

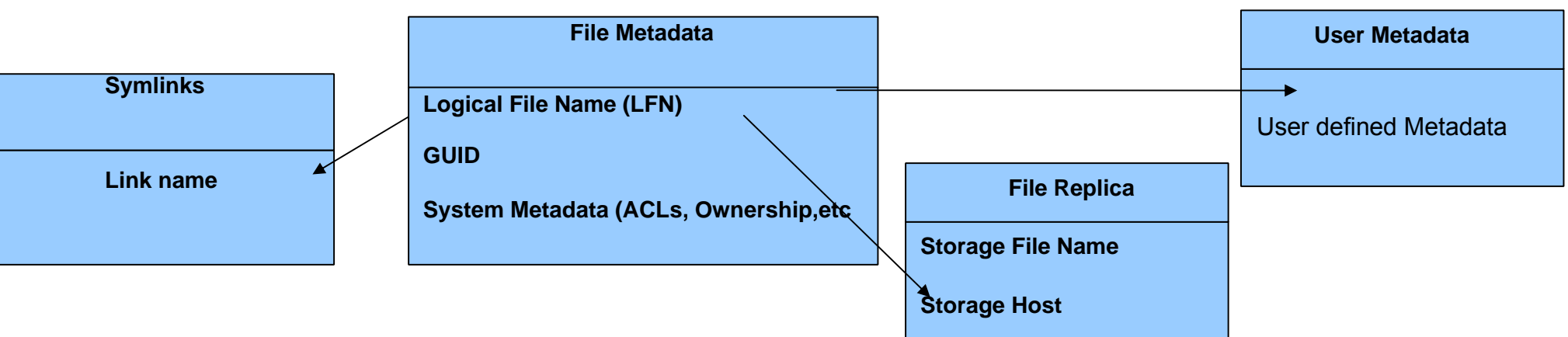
- User exposed transaction API.
- Hierarchical namespace and namespace operations.
- Integrated GSI Authentication and Authorization.
- Access Control Lists (Unix Permissions and POSIX ACLs).
- Checksums.
 - Integration with VOMS (VirtualID and Virtual GID)

Supported database backends: **Oracle** and **MySQL**

GFAL integration and support to lcg-* done by Grid Deployment group



- LFC stores both **logical** and **physical** mappings for the file in the same database → Speed up of operations
- Treats all entities as files in a **UNIX-like** filesystem.
- File **API** also similar to UNIX (create(), mkdir(), chown()....)
- Hierarchical namespace of **LFNs** mapped to the **GUIDs**
- **GUIDs** mapped to the physical locations of file **replicas** in the storage
- **System attributes** of files (creation time, file size and checksum...) stored as LFN attributes
- One field for **user-defined metadata**
- Multiple LFNs per GUID allowed as **symbolic links** to the primary LFN.





Summary of the LFC Catalog commands

lfc-chmod	Change access mode of the LFC file/directory
lfc-chown	Change owner and group of the LFC file-directory
lfc-delcomment	Delete the comment associated with the file/directory
lfc-getacl	Get file/directory access control lists
lfc-ln	Make a symbolic link to a file/directory
lfc-ls	List file/directory entries in a directory
lfc-mkdir	Create a directory
lfc-rename	Rename a file/directory
lfc-rm	Remove a file/directory
lfc-setacl	Set file/directory access control lists
lfc-setcomment	Add/replace a comment

Listing the entries of a LFC directory:

```
lfc-ls [-cdiLlRTu] [--comment] path
```

where *path* specifies the LFC pathname (mandatory)

- Remember that LFC has a directory tree structure
- */grid/<VO_name>/<you create it>*



- All members of a given VO have read-write permissions under their directory
- *-l* (it is a lowercase “L”) outputs long listing
- *-R* lists the contents of directories recursively (don’t use it so often!)
- You can set *LFC_HOME* to use relative paths:
LFC_HOME=/grid/gilda/myDir → */grid/gilda/myDir/myFile* becomes *myFile*



Creating directories in the LFC

```
lfc-mkdir [-m mode] [-p] path...
```

- Where *path* specifies the LFC pathname
- Remember that while registering a new file (using `lfc-cr`, for example) the corresponding destination directory must be created in the catalog beforehand.
- Examples:

```
> lfc-mkdir /grid/gilda/school/test1
```

You can just check the directory with:

```
> lfc-ls -l /grid/gilda/school
```

```
drwxr-xrwx 0 19100 1099 0 Jan 14 11:36 test
```

Creating a symbolic link – Equivalent to a lcg alias in LFC

```
lfc-ln -s file linkname  
lfc-ln -s directory linkname
```

Create a link to the specified *file* or *directory* with *linkname*

- *Examples:*

```
> lfc-ln -s /grid/gilda/school/test/dummyFile /grid/gilda/school/dummyLink
```



Let's check the link using `lfc-ls` with long listing (-l):

```
> lfc-ls -l
```

```
lrwxrwxrwx 1 19100 1099 0 Jan 14 11:58 dummyLink -> /grid/gilda/school/test/dummyFile  
drwxr-xrwx 1 19100 1099 0 Jan 14 11:39 test
```



Adding/deleting metadata information:

```
lfc-setcomment path comment  
lfc-delcomment path
```

lfc-setcomment adds/replaces a *comment* associated with a file/directory in the LFC Catalog

lfc-delcomment deletes a comment previously added

- **Example:**

```
lfc-setcomment /grid/gilda/school/test/dummyFile 'Hello  
World!'
```

- **Check your job with**

```
lfc-ls --comment /grid/gilda/school/test/dummyFile
```



Low level methods (many POSIX-like):

lfc_access	lfc_deleteclass	lfc_listreplica	lfc_setacl
lfc_aborttrans	lfc_delreplica	lfc_lstat	lfc_setatime
lfc_addreplica	lfc_endtrans	lfc_mkdir	lfc_setcomment
lfc_apiinit	lfc_enterclass	lfc_modifyclass	lfc_seterrbuf
lfc_chclass	lfc_errmsg	lfc_opendir	lfc_setfsize
lfc_chdir	lfc_getacl	lfc_queryclass	lfc_starttrans
lfc_chmod	lfc_getcomment	lfc_readdir	lfc_stat
lfc_chown	lfc_getcwd	lfc_readlink	lfc_symlink
lfc_closedir	lfc_getpath	lfc_rename	lfc_umask
lfc_creat	lfc_lchown	lfc_rewind	lfc_undelete
lfc_delcomment	lfc_listclass	lfc_rmdir	lfc_unlink
lfc_delete	lfc_listlinks	lfc_selectsrvr	lfc_utime
			send2lfc



- **lcg_utils**: lcg-* commands + lcg_* API calls
 - Provide (all) the functionality needed by the LCG user
 - Transparent interaction with file catalogs and storage interfaces when needed
 - Abstraction from technology of specific implementations
- Grid File Access Library (**GFAL**): API
 - Adds file I/O and explicit catalog interaction functionality
 - Still provides the abstraction and transparency of lcg_utils
- **edg-gridftp** tools: CLI
 - Complete the lcg_utils with low level GridFTP operations
 - Functionality available as API in GFAL
 - May be generalized as lcg-* commands



Interactions with SE require some components:

- File catalog services to locate replicas
- SRM
- File access mechanism to access files from the SE on the WN

GFAL does all this tasks for you:

- Hides all these operations
- Presents a POSIX interface for the I/O operations
 - Single shared library in threaded and unthreaded versions
libgfal.so, libgfal_pthr.so
 - Single header file
gfal_api.h
- **User can create all commands needed for storage management**
- **It offers as well an interface to SRM**

Supported protocols:

- file (local or nfs-like access)
- dcap, gsidcap and kdcap (dCache access)
- rfio (castor access) and gsirfio (dpm)

- **C API**

- The header file `gfal_api.h` needs to be included in the application source code to get the prototype of the functions.
- The function names are obtained by prepending `gfal_` to the Posix names, for example `gfal_open`, `gfal_read`, `gfal_close` ...
- The argument lists and the values returned by the functions are identical.
- The variable `errno` is set to the **Posix Error Codes** in the case of failure.

- **Java API (C API Wrapper)**

- It provides three main Java Objects that need to be imported in the java applications in order to hide the underlying C functions.
 - `GFalFile` : to handle and read/write files
 - `GFalDirectory` : to handle and manage directories (create, delete, list)
 - `GFalUtilities` : to manage file (rename, stat, lstat, delete)

- **Examples in gLite3 User Guide (Appendix F)**
 - <https://edms.cern.ch/file/722398//gLite-3-UserGuide.pdf>
- **GFAL C API Description:**
 - http://grid-deployment.web.cern.ch/grid-deployment/documentation/LFC_DPM/gfal/html/
- **GFAL JAVA API**
 - <https://grid.ct.infn.it/twiki/bin/view/GILDA/APIGFAL>
- **GFAL Java API code and libraries:**
 - https://grid.ct.infn.it/twiki/pub/GILDA/APIGFAL/GFAL_Java_API.zip
- **On-line JavaDoc of Java API:**
 - <https://grid.ct.infn.it/twiki/GFAL/>
- **GFAL Excercises (C/Java):**
 - <https://grid.ct.infn.it/twiki/bin/view/GILDA/UsingGFAL>



- **High level interface (CL tools and APIs) to**
 - Upload/download files to/from the Grid (UI,CE and WN <---> SEs)
 - Replicate data between SEs and locate the best replica available
 - Interact with the file catalog
- ***Definition:* A file is considered to be a **Grid File** if it is both physically present in a SE and registered in the File Catalog**
- ***lcg-utils*** ensure the consistency between files in the Storage Elements and entries in the File Catalog



Replica Management

lcg-cp	Copies a grid file to a local destination
lcg-cr	Copies a file to a SE and registers the file in the catalog
lcg-del	Delete one file
lcg-rep	Replication between SEs and registration of the replica
lcg-gt	Gets the TURL for a given SURL and transfer protocol
lcg-sd	Sets file status to “Done” for a given SURL in a SRM request

File Catalog Interaction

lcg-aa	Add an alias in LFC for a given GUID
lcg-ra	Remove an alias in LFC for a given GUID
lcg-rf	Registers in LFC a file placed in a SE
lcg-uf	Unregisters in LFC a file placed in a SE
lcg-la	Lists the alias for a given SURL, GUID or LFN
lcg-lg	Get the GUID for a given LFN or SURL
lcg-lr	Lists the replicas for a given GUID, SURL or LFN



Upload a file to a SE and register it into the catalog:

```
lcg-cr -d dest_file | dest_host -l lfn [-g guid] [-l lfn]  
[-v | --verbose] --vo vo_name src_file
```

where

- **dest_host** is the fully qualified hostname of the destination SE
- **dest_file** is a valid SURL (both sfn:// or srm:// format are valid)
- **guid** specifies the Grid Unique Identifier. If this option is not present, a GUID is generated internally
- **lfn** specifies the Logical File Name associated with the file
- **vo** specifies the Virtual Organization the user belongs to
- **src_file** specifies the source file name: the protocol can be *file:///* or *gsiftp:///*

The command help is misleading: -d -l and --vo are MANDATORY!

On the file protocol, only use 1 or 3 slashes!



Adding an alias for a given GUID

```
lcg-aa --vo vo guid lfn
```

where

- *vo* specifies the Virtual Organization the user belongs to
- *guid* specifies the Grid Unique Identifier of the file you want to add the alias to
- *lfn* specifies the new alias

- **Example:**

```
$ lcg-aa --vo gilda guid:402fc31a-b549-43ac-8272-5540fc24137d  
lfn:/grid/gilda/tutorial/aliasToNote.txt
```

- **To check if the previous command was successful, you can use lcg-la command to list the aliases for a given LFN, GUID or SURL**

```
$ lcg-la --vo gilda lfn:/grid/gilda/tutorial/note.txt  
lfn:/grid/gilda/tutorial/note.txt
```



Copying a file from one SE to another one and register it in the Catalog

```
lcg-rep -d dest_file | dest_host [-v | --verbose]  
--vo vo src_file
```

where

- ***dest_host*** is the fully qualified hostname of the destination SE
- ***dest_file*** is a valid SURL (both sfn:// or srm:// are valid)
- ***vo*** specifies the Virtual Organization the user belongs to
- ***src_file*** specifies the source file name: the protocol can be LFN, GUID or SURL. An SURL scheme can be sfn: for a classical SE or srm:



Listing of replicas for a given LFN, GUID or SURL

```
lcg-lr --vo vo_name file
```

where

- **vo_name** specifies the Virtual Organization the user belongs to
- **file** specifies the Logical File Name, the Grid Unique Identifier or the Site URL. An SURL scheme can be sfn: for a classical SE or srm:

Deleting replicas

```
lcg-del [ -a ] / [ -s se ] [ -v / --verbose ] --vo vo file
```

where

- **a** is used to delete all replicas of the given file
- **se** specifies the SE from which you want to remove the replica
- **vo** specifies the Virtual Organization the user belongs to
- **file** specifies the Logical File Name, the Grid Unique Identifier or the Site URL. An SURL scheme can be sfn: for a classical SE or srm:.

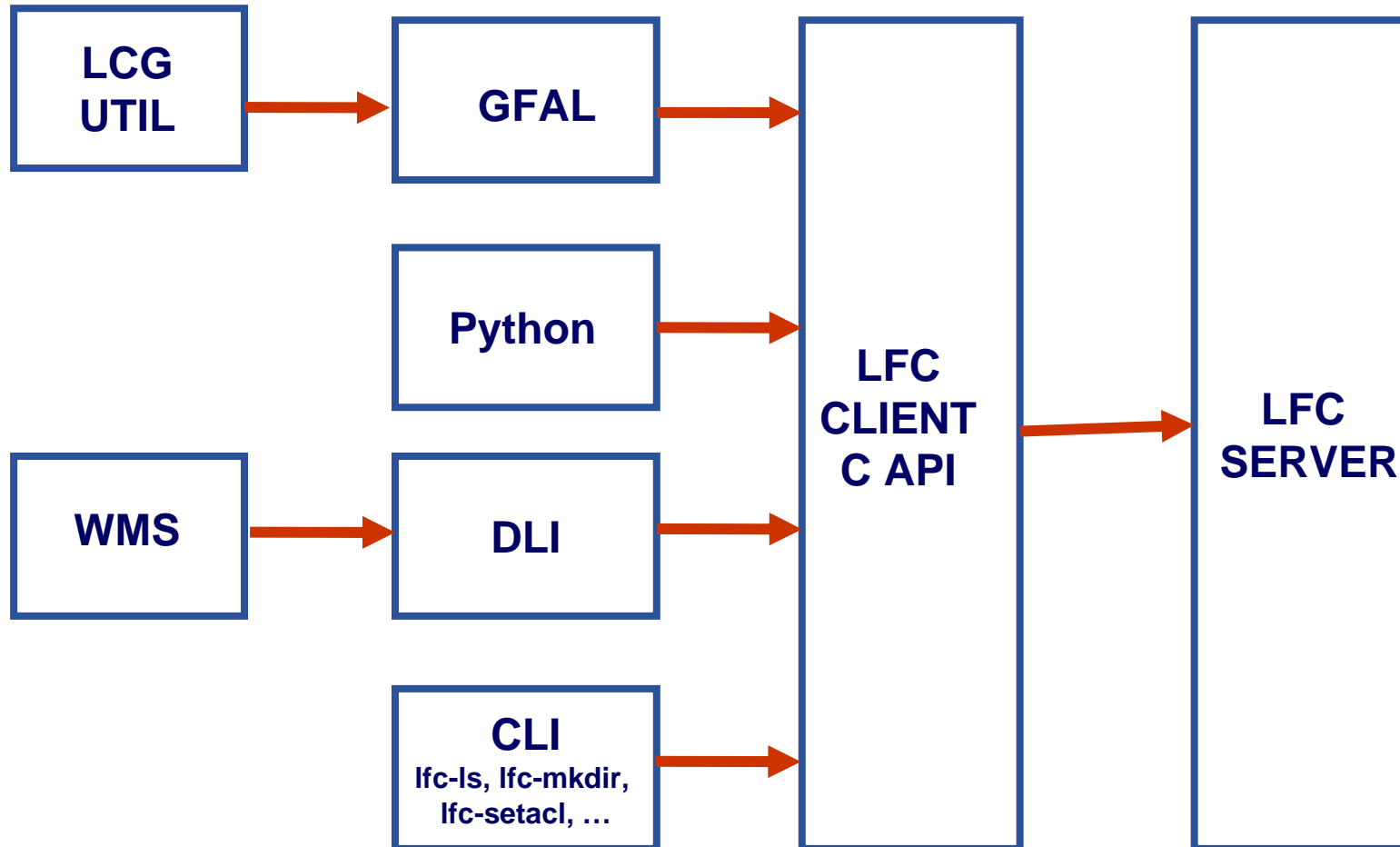


Downloading a Grid file in a SE to a local destination

```
lcg-cp [ -v | --verbose ] --vo vo src_file dest_file
```

where

- **vo** specifies the Virtual Organization the user belongs to
- **src_file** specifies the source file name: the protocol can be LFN, GUID, SURL or local file. An SURL scheme can be sfn: for a classical SE or srm:
- **dest_file** specifies the destination. The protocol can be file:/// or gsiftp://





- **LFC client commands**

- Provide administrative functionality
- Unix-like
- LFNs seen as a Unix filesystem (/grid/<VO>/ ...)

- **LFC C API**

- Alternative way to administer the catalog
- Python wrapper provided

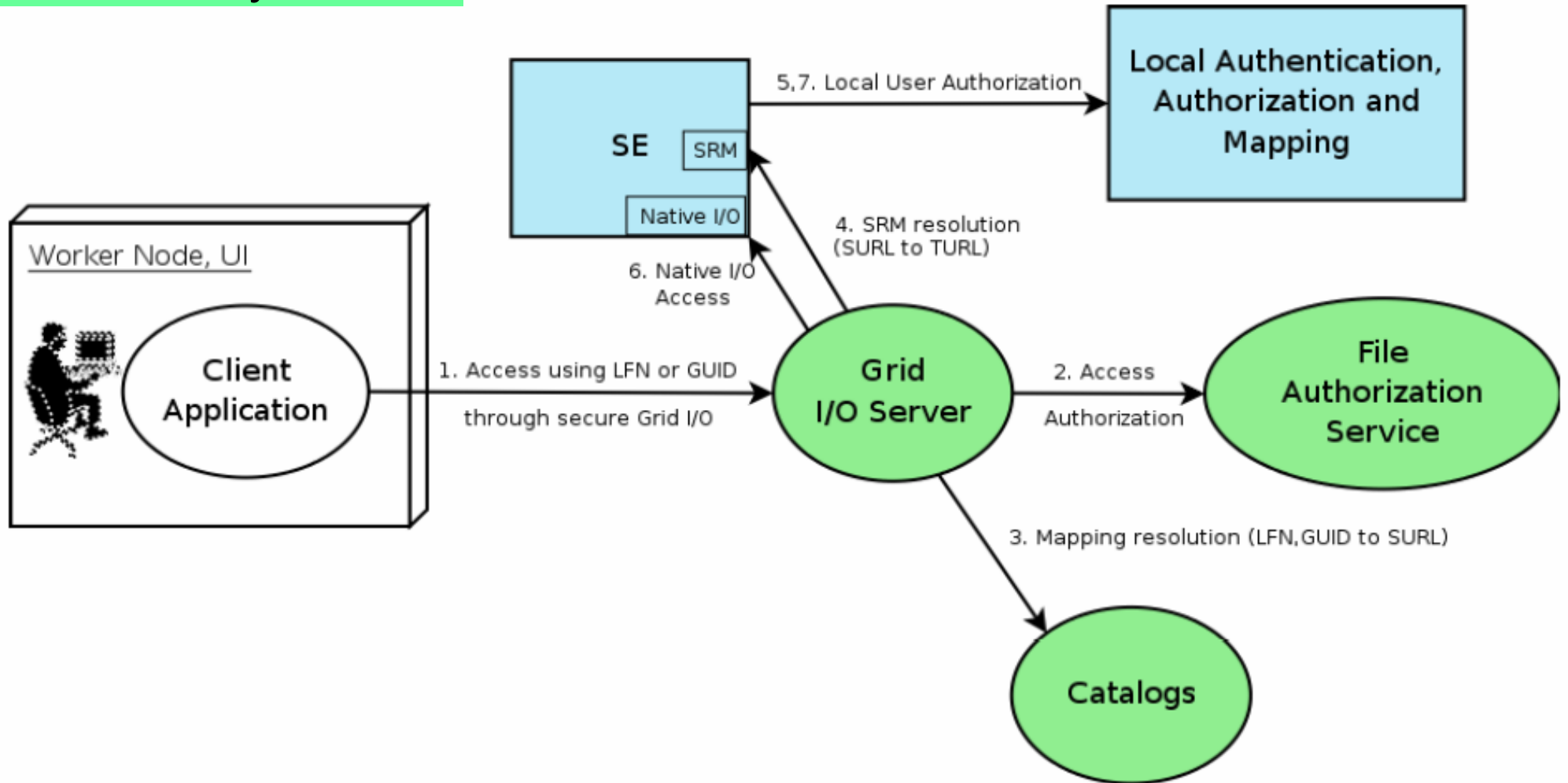
- **Integration with GFAL and lcg_util APIs complete**
→ lcg-utils access the catalog in a transparent way

- **Integration with the WMS completed**
 - The RB can locate Grid files: allows for data based match-making



Provided by site

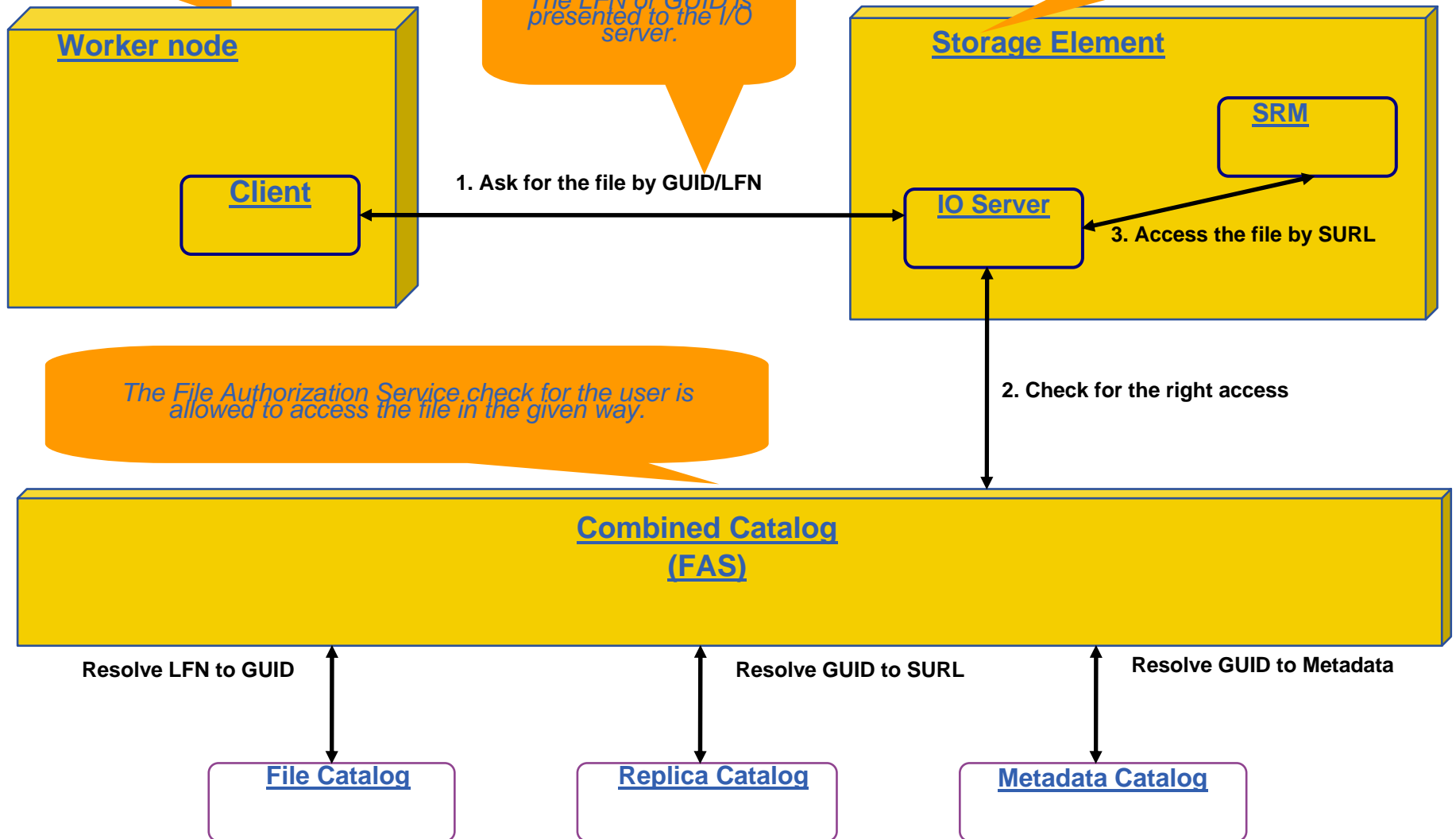
Provided by VO



The I/O client library accepts either LFN or GUID as an input to the API.

The LFN or GUID is presented to the I/O server.

The GUID or LFN is resolved into the SURL, which is used by the local SRM to access the file.





- Grids are naturally distributed systems
- The means that data also needs to be distributed
 - First generation data distribution mainly concentrated on copy protocols in a grid environment:
 - gridftp
 - http + mod_gridsite
- But copies controlled by clients have problems...



- Many Grid applications will distribute a LOT of data across the Grid sites
- Need efficient and easy way to manage File movement service
- **gLite File Transfer Service (FTS)**
 - Manage the network and the storage at both ends
 - Define the concept of a CHANNEL: a link between two SEs
 - Channels can be managed by the channel administrators, i.e. the people responsible for the network link and storage systems
 - These are potentially different people for different channels
 - Optimize channel bandwidth usage – lots of parameters that can be tuned by the administrator
 - VOs using the channel can apply their own internal policies for queue ordering (i.e. professor's transfer jobs are more important than student's)
- **gLite File Placement Service (FPS)**
 - It **IS** an FTS with the additional catalog lookup and registration steps, i.e. LFNs and GUIDs can be used to perform replication. Could've been called File Replication Service (**replica = managed/catalogued copy**)



- **File movement is asynchronous – submit a job**
 - Held in file transfer queue
- **Data scheduler**
 - Single service per VO – can be distributed
 - VO can apply policies (priorities, preferred sites, recovery modes..)
- **Client interfaces:**
 - Browser
 - APIs
 - Web service
- **“File transfer”**
 - Uses SURL
- **“File placement”**
 - Uses LFN or GUID, accesses Catalogues to resolve them



- **File movement is asynchronous – submit a job**
 - Held in file transfer queue
- **FPS fetches job transfer requests, contact File Catalogue obtaining source / destination SURLs**
- **Task execution is demanded to FTS**
- **User can monitor job status through jobID**
- **FTS maintains state of job transfers**
- **When job is done, FPS updates file entry in the catalogue adding the new replica**

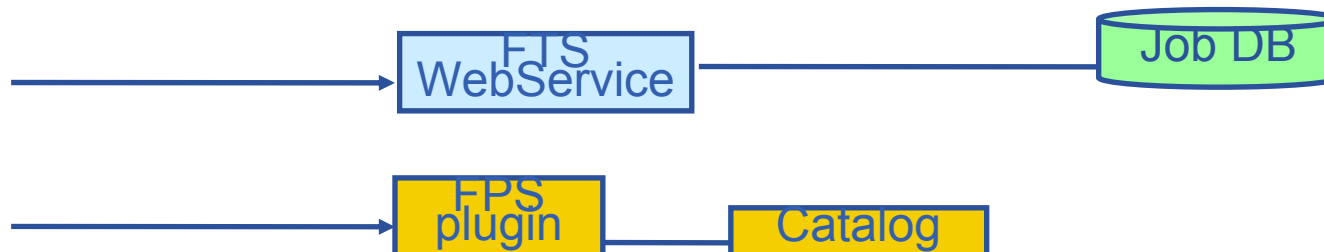


- **Data transfer and access protocol for secure and efficient data movement**
- **Standardized in the Global Grid Forum**
- **extends the standard FTP protocol**
 - Public-key-based **Grid Security Infrastructure (GSI)** or Kerberos support (both accessible via GSS-API)
 - **Third-party** control of data transfer
 - **Parallel data transfer**
 - **Striped data transfer** Partial file transfer
 - Automatic negotiation of TCP buffer/window sizes
 - Support for reliable and restartable data transfer
 - Integrated instrumentation, for monitoring ongoing transfer performance

- **GridFTP is the basis of most transfer systems**
- **Retry functionality is limited**
 - Only retries in case of network problems; no possibility to recover from GridFTP a server crash
- **GridFTP handles one transfer at a time**
 - No possibility to do bulk optimization
 - No possibility to schedule parallel transfers
- **Need a layer on top of GridFTP that provides reliable scheduled file transfer**
 - FTS/FPS
 - Globus RFT (layer on top of single gridftp server)
 - Condor Stork



- **File Transfer Service (FTS)**
 - Acts only on SRM SURLs or gsiftp URLs
 - `submit(source-SURL, destination-SURL)`
- **File Placement Service (FPS)**
 - A plug-in into the File Transfer that allows to act on logical file names (LFNs)
 - Interacts with replica catalogs (similar to gLite-I/O)
 - Registers replicas in the catalog
 - `submit(transferJobs)` (*transferJob = sourceLFN, destinationSE*)

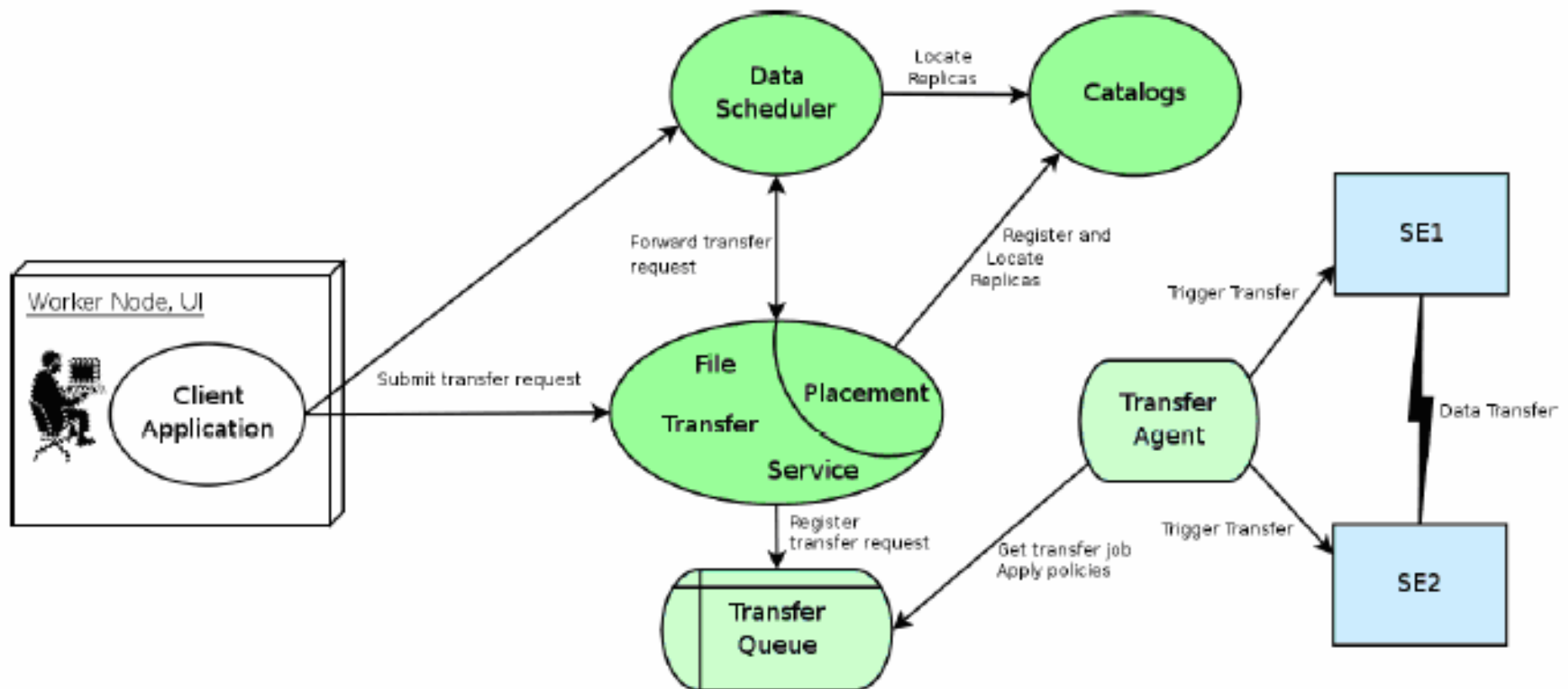


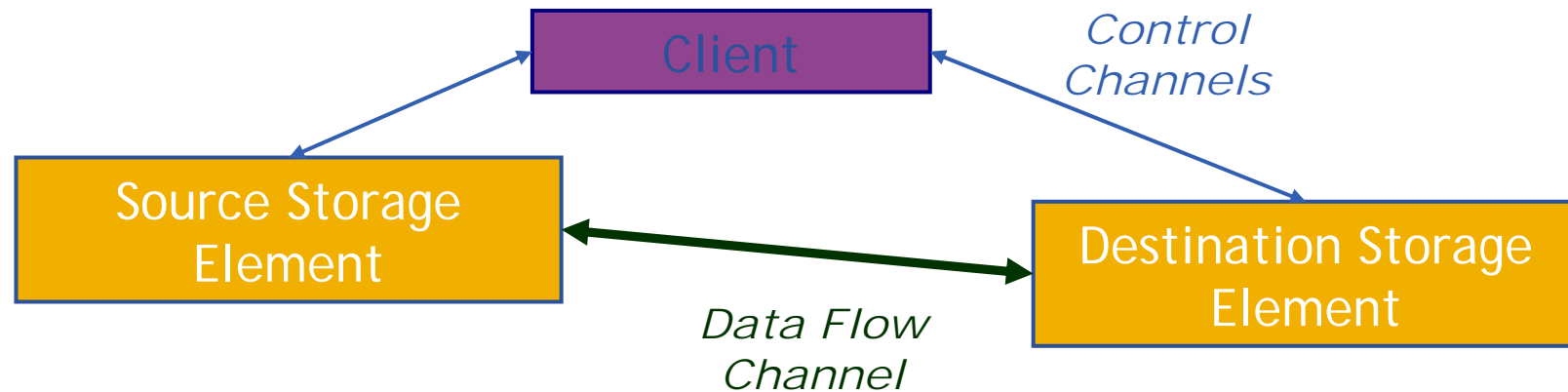


- **Using the File Transfer Service (FTS)**
 - Initiate and monitor transfer
 - Plugin takes care of catalog interactions
- **Using the File Placement Service (FPS)**
 - Lookup source SURL in replica catalog
 - Initiate and monitor transfer
 - After successful transfer register new replica in the catalog
- **FTS and FPS offer the same interface**
 - Difference only in input parameters to the submit command
 - Different configuration
 - SURLs vs. LFNs
 - FPS requires catalog endpoint



- Data Scheduler (**DS**) Keep track of user/service transfer requests
- File Transfer/Placement Service (**FTS/FPS**)
- Transfer Queue (Table)
- Transfer Agent (Network)



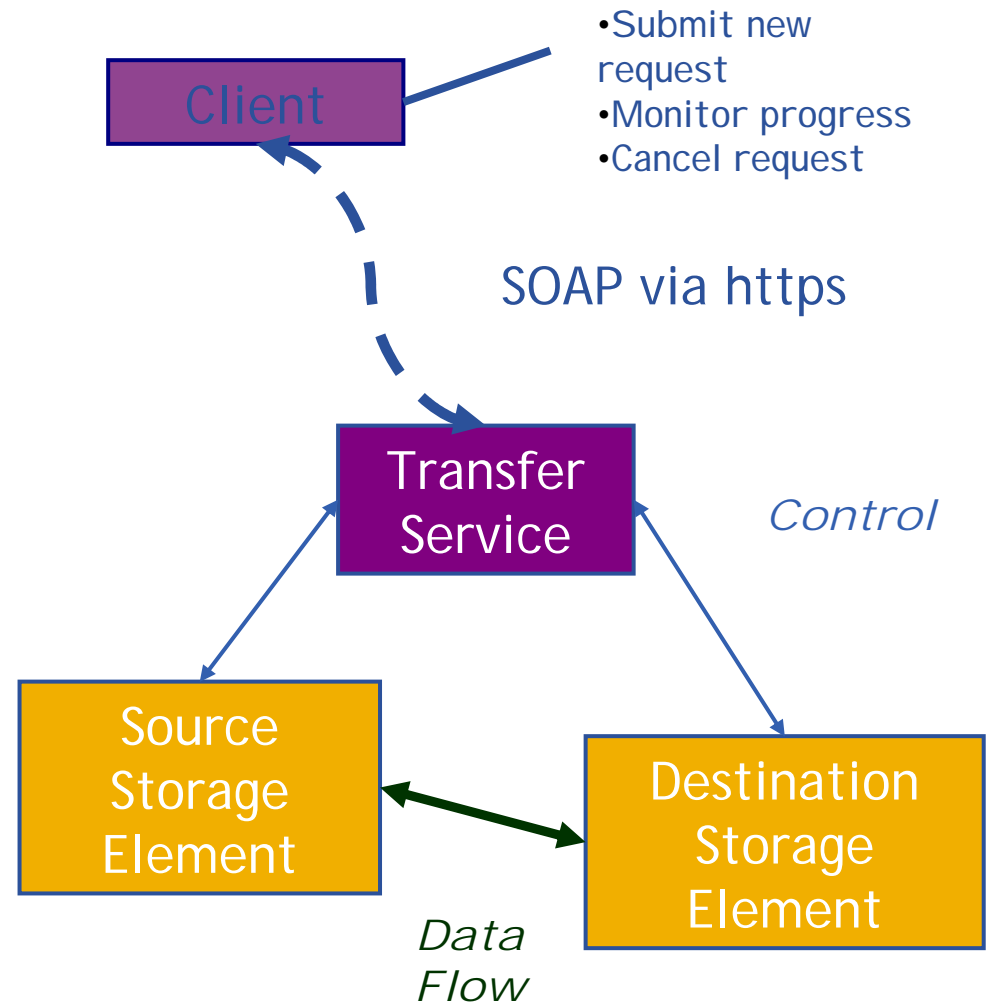


- **Although transport protocol may be robust, state is held inside client – inconvenient and fragile.**
- **Client only knows about local state, no sense of global knowledge about data transfers between storage elements.**
 - Storage elements overwhelmed with replication requests
 - Multiple replications of the same data can happen simultaneously
 - Site has little control over balance of network resources - DOS



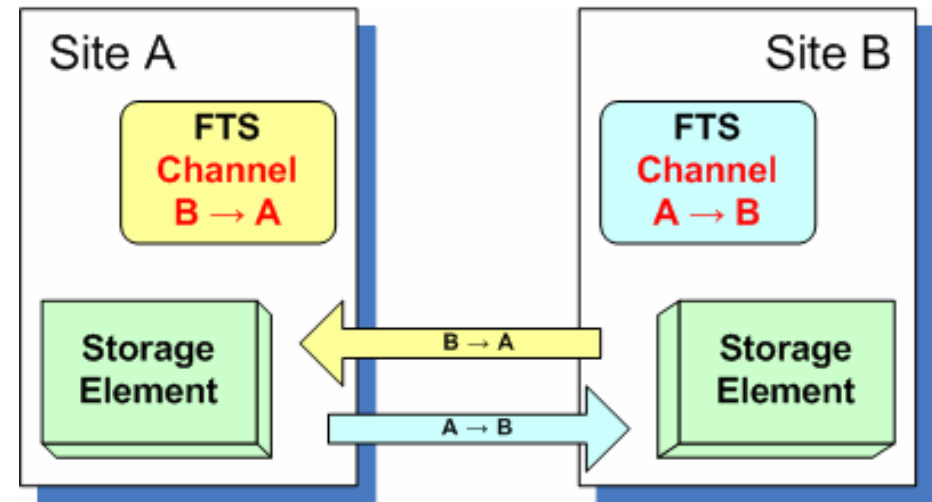
- **Clear need for a *service* for data transfer**

- Client connects to service to submit request
- Service maintains state about transfer
- Client can periodically reconnect to check status or cancel request
- Service can have knowledge of global state, not just a single request
 - Load balancing
 - Scheduling





- FTS Service has a concept of *channels*
- A channel is a *unidirectional* connection between two sites
- Transfer requests between these two sites are assigned to that channel
- Channels usually correspond to a dedicated network pipe associated with production
- But channels can also take wildcards:
 - * to MY_SITE : All incoming
 - MY SITE to * : All outgoing
 - * to * : Catch all



- Channels control certain transfer properties: transfer concurrency, gridftp streams.
- Channels can be controlled independently: started, stopped, drained.



- **Storage Element** – save data and provide a common interface
 - Storage Resource Manager (SRM) Castor, dCache, DPM, ...
 - Native Access protocols rfio, dcap, nfs, ...
 - Transfer protocols gsiftp, ftp, ...
- **Catalogs** – keep track where data are stored
 - File Catalog
 - Replica Catalog
 - Metadata Catalog

} LCG File Catalog (LFC)

AMGA Metadata Catalogue
- **Data Movement** – schedules reliable file transfer
 - File Transfer Service gLite FTS

(manages physical transfers)

- gLite documentation homepage
 - <http://glite.web.cern.ch/glite/documentation/default.asp>
- DM subsystem documentation
 - <http://egee-jra1-dm.web.cern.ch/egee-jra1-dm/doc.htm>
- LFC and DPM documentation
 - <https://uimon.cern.ch/twiki/bin/view/LCG/DataManagementDocumentation>
- FTS user guide
 - <https://edms.cern.ch/file/591792/1/EGEE-TECH-591792-Transfer-CLI-v1.0.pdf>