PROJECT WORK PLAN ADDENDUM

2004

Organization: CERN – LCG Project

LCG Persistency Framework Project

Document Revision #: 0.7

Date of Issue: 1.4.2004

Editors: Dirk Duellmann

Approval Signatures

Approved by: Project Leader

Approved by: LCG Project Leader

Prepared by: Project Manager

Prepared by: LCG Project Manager

Reviewed by: Quality Assurance Manager

Document Change Control

This section provides control for the development and distribution of revisions to the Project Work Plan up to the point of approval.

Revision Number	Date of Issue	Author(s)	Brief Description of Change
0.5	4.03.2004	D. Duellmann	Initial Draft
0.6	7.03.2004	D. Duellmann	Comments from AF, more task descriptive text, references and milestones sections added
0.7	5.04.2004	D. Duellmann	Comments from second AF, milestones changes

Table of Contents

1. Introduction

This document is an addendum to the POOL Project Work Plan document to describe with some detail the proposed work items for the period from January-2004 to December 2004.

The current work packages of the Persistency Framework project are listed in the following table.

Table 1 POOL Work Packages

Storage Manager	Object storage to persistent media
File Catalog	Handling of physical and logical file names and associated meta data
Collections and Meta Data	Handling of large collections of objects and associated meta data
Infrastructure and Testing	POOL development and testing infrastructure
Conditions Database	Version controlled access to condition data

2. POOL Tasks

The following task list has been derived from the functional request and current deployment schedules of all experiments planning to use POOL in their baseline software frameworks. The list is structured by the affected POOL work package and annotated with an estimated quarter of completion. More detailed estimates of the required development efforts are given in a table below.

2.1 POOL General Tasks and End User Utilities

2.1.1 Automated (re)creation of POOL file catalogs from a set of files in the local file system (Q1)

Based on physical file names and GUID information stored by POOL in each POOL file. This tool is mostly addressing small scale development use cases where a single user would like to get access to a small set of POOL files which are self contained. Even though this tool is very useful in simple cases it cannot replace proper catalog management. Eg all meta data and any logical file names can not be extracted just from a POOL file and need to be recreated by other means.

2.1.2 Consistent move and delete of files in a local file system (Q2)

ATLAS has asked to provide tools which could simplify the consistent handling of local file system operations like file renaming/moving with the associated catalog modification.

2.1.3 Use Relational Abstraction Layer for production versions of the POOL Collection and File Catalog component (Q2/Q3)

To achieve independence of the MySQL backend database we will remove direct MySQL++ use in the POOL code by an intermediate RDBMS abstraction layer which will allow to run these RDBMS based POOL component against a larger set of backend databases (at least MySQL, generic ODBC and ORACLE).

2.1.4 Integrate with SEAL Component Model (Q3)

Assuming that several experiments have picked up the SEAL component model we would change POOL to load and configure the internal POOL components also via the SEAL provided services. This would improve the consistency of component handling for the experiments and possibly simplify the management inside POOL.

2.1.5 Additional POOL ports (Q2/Q3)

We will complete the list of supported POOL platforms as defined by the Architect Forum. In particular CMS has requested explicitly support for icc 8.0, ecc (ia/64), Mac-OS platforms.

2.2 Storage Manager

2.2.1 Fix for CLHEP matrix problem (Q1)

The current POOL implementation does not properly support user defined allocator objects, which contain data elements for std containers. This results in a problem for storing CLHEP Matrix classes, which use such allocators to optimise their internal memory management. A workaround based on customised streamers for this problem has been developed in LHCb and communicated to ATLAS. Still POOL should reduce the storage manager assumptions on classes using std containers and allow a direct storage for this classes.

2.2.2 Add support for (remote) access to files catalogued in the LCG file catalog (including srm based file name syntax as defined together with LCG-GD and SURL to TURL translation) (Q3)

This could be done either in POOL or one level down in ROOT eg via a TFile plugin for ROOT based on ARDA abstraction of GRID file access.

2.2.3 Performance and Storage optimisation (Q1-Q4)

A significant focus of the POOL development will be on a systematic optimisation of the POOL performance and the minimisation of POOL resource usage. In particular we plan to address:

- Mass storage handling (minimise costly requests)
- Client side resource usage - memory, CPU, file handles

This will be an ongoing activity driven by requests from the experiments and the providers of lower level services.

2.2.4 POOL/ROOT interactivity (Ref<T> plugin) (Q2)

The internal review has requested POOL to allow the use of POOL object references inside ROOT as a interactive analysis framework. We expect to develop together with the ROOT team a plug-in module for ROOT, which will extend ROOT with the transparent navigation facilities provided by the POOL framework.

2.2.5 Schema Evolution (Q2/Q3)

As requested by the internal review we will define together with the experiments the requirements for schema evolution and confirm that POOL does not significantly constrain the support for schema evolution, which is implemented in ROOT. For this purpose we will start from the supported schema evolution cases listed in the ROOT manual and document which subset is supported with each POOL version. We will implement tests suite which confirms that all required use cases are supported and introduce these tests into the POOL build procedure.

A RDBMS schema evolution policy needs to be defined after the Object-to-RDBMS mapping implementation has been clarified.

2.2.6 Support for automated translation of doubles to floats (Q3)

ATLAS and LHCb have asked POOL to support a mapping of double attributes on the transient side to float attributes on the persistent side and back (as provided eg in ROOT). The same functionality will be provided by POOL eg via an annotation of affected classes/attributes in the LCG dictionary. Numerical instabilities resulting from the platform dependent conversion between float and double will not be handled by POOL.

2.2.7 Follow the proposed evolution of SEAL dictionary

The SEAL work plan has outlined a proposed evolution of SEAL and ROOT to achieve a common dictionary implementation. POOL will continue to follow this evolution assuming that the changes introduced are compatible with the POOL commitment to support existing persistent data.

2.2.8 Object storage on the RDBMS layer (Q2/Q3)

POOL so far provides support for object storage only via ROOT. For application meta data which is frequently updated, may need transaction control or other relational functionality like automatic replication or server side query we foresee to re-implement the existing POOL interface using our Relational Abstraction Layer. This would allow storing objects in any of the backend databases supported by RAL.

We propose to proceed in two steps.

- **RDBMS table -> Object mapping (Q2/Q3)**
  We support the transparent mapping of only C++ objects (rather structures) which can be trivially mapped to a SQL table definition (only elementary attribute types, no or only trivial inheritance)

- **Complex Object -> RDBMS mapping (Q1 2005)**
  At a later stage this could be extended to allow for more complex C++ types including complex attribute types and inheritance handling if required by the experiments.

A more detailed description of this work item can be found at:
/afs/cern.ch/project/lcg/app/www/pool/relational_urd.pdf

2.2.9 On demand Dictionary loading (Q2)

Based on the new SEAL service for automated, type based dictionary loading POOL will together with ROOT team integrate new class definitions into running ROOT storage service also after the file requiring the new types has already been opened and used. This functionality needs to be addressed already before the POOL migration to ROOT v4.x.

2.2.10 Storage Manager scripting interface in Python (Q3)

To complete the set of python interfaces to POOL components we plan to also provide a python binding to the POOL storage / persistency service.

2.2.11 ROOT v4.0 integration (Q3)

POOL will during 2004 move to the next major ROOT release 4.0. This move will need careful preparation as ROOT provides more complete STL support, which is hoped to replace in the longer term the POOL developed code. As POOL needs to stay compatible to data written with older POOL releases we expect to support the current POOL STL streaming in parallel for quite some time to allow access to existing files. We expect to work closely together with the ROOT team in order to minimise the impact of changes in ROOT 4.0 on POOL users with existing data.

2.3 File Catalog

2.3.1 Support the LCG-2 RLS (Q1)

This task has been completed in POOL V1.5.

2.3.2 Composite Catalog support (Q1)

Including support for a prioritized list of file catalogs. Separation between read and write catalogs. Basic support completed with POOL V1.6. Some remaining consistency issues (eg catalog entries for files registered in several catalogs) still need to be clarified.

2.3.3 Support for pluggable replica optimisation (Q1/Q2)

2.3.4 Insure that POOL and LCG replica manager fully interoperate and share a common representation for file names in the LCG file catalog (Q1/Q2)

2.3.5 Integrate POOL with ARDA provided file catalog components (Q3)

Implement the POOL File Catalog interface based on the ARDA catalog and meta data components.

2.3.6 Participate in ARDA/EGEE requirement gathering discussions (Q2)

Make sure that ARDA/EGEE deliverables allow the integration with POOL. In example catalog, meta data, security and I/O models are compatible with POOL assumptions.

2.3.7 Support GLOBUS RLS developers in their POOL Catalog implementation (Q1-4)

On demand by the GLOBUS developers implementing the POOL catalog interface.

2.4 Collections and Meta Data

2.4.1 Support for Tokens in Attribute Lists (Q1)

POOL should allow in addition to elementary types also POOL Object Tokens to be kept as file or collection level meta data (in V1.6). At this point we do not expect to extend the meta data support much further as genuine support for more complex meta data is coming up via the RDBMS-to-Object mapping.

2.4.2 Provide POOL collection plug-in for ROOT (Q2/Q3)

POOL collection functionality should be made available inside ROOT via ROOT plug-in. This task needs a close collaboration between the POOL and ROOT teams.

2.4.3 Separation of logical and physical collection identification (Q2)

Collections should be catalogued and referred to similar as files via a logical collection name (and possibly via a unique immutable GUID). The implementation of this catalog could be shared with the file catalog.

2.4.4 Collection Extraction and publishing tools (Q2)

To extract complete (or partial) POOL collections based on event or collection level meta a set of tools similar to those of the file catalog is required.

2.4.5 Integrate POOL collections with ARDA provided services (Q3)

This task includes the collection cataloguing, integration with middleware support for the concept of file sets and POOL collection integration with a ARDA provided parallel analysis back end (eg based on ROOT/PROOF).

2.4.6 Collection end user interface (eg AIDATupel) (Q3)

POOL so far provides only a developer level interface to its collection. The provision of the real end-user integration was expected to be part of the PI or ARDA projects, to be addressed once ARDA is clearly defined, which is still to happen. As POOL files and collections will soon be used for analysis the end-user interface must be addressed now. Therefore until the final organization is established, the POOL collections WP will host the work on end-user interface to collections, with participation broadened to include

end-user analysis interests.

2.5 Infrastructure and Testing

2.5.1 Review documentation structure (Q1)

This issue has been addressed in V1.6 with a move to a more modular documentation based on DocBook modules.

2.5.2 Complete automated data format regression test (Q1)

The file format regression tests in POOL need to be fully integrated into the automated POOL build and need to be extended to also to check for relational database data regressions.

2.5.3 Incorporation of experiment defined test suites into the POOL release test procedure (Q2)

The persistency of data structures crucial for LHC experiment software systems should be checked already during the POOL internal testing. We therefore propose to include key test classes from all main experiments together with experiment provided functional tests.

2.5.4 Evaluate/Move to AppWorks (Q1)

Given the unclear support for SCRAM - POOL should evaluate AppWorks and possibly migrate to it.

2.5.5 Complete move to QmTest (Q1)

All POOL tests should be executed under QmTest the proposed test environment of LCG- SPI

2.5.6 Parallel build and test machinery (Q2)

We would like to extend the POOL build system to allow for parallel builds on different platforms.

2.5.7 Extend POOL online tutorial (Q2)

This task is part of the preparation of the POOL hands-on exercises for the CERN School of Computing 2005.

2.6 Conditions DB

2.6.1 First IOV interface and implementation (Q2)

This task includes the provision of a first release of the the Interval Of Validity (IOV) service with both the MySQL and the ORACLE backend. Only the common subset of the interface is provided with same semantics for both backends.

2.6.2 Connection to POOL Data (Q2)

This release provides example programs which show how to combine objects stored in a POOL store (ROOT or relational) with the IOV service above.

2.6.3 Condition catalog (review of CondDB folders) (Q3)

In this task we will address the possibility of common meta data handling for the condition db with other meta data components (File Catalog, Collection Catalog).

3. Summary

3.1 Resource Requirements

Table 2 Summary of the required resources for each major work item. The estimated FTE is calculated assuming an effective fulltime year of 40 weeks

	Work Item	FTEweek	FTE
Common Tasks		22	0.55
	Automated (re)creation of POOL file catalogs	2	
	Consistent move and delete of files in a local file system	4	
	Use Relational Abstraction Layer for production FC/Col	10	
	Integrate with SEAL Component Model	2	
	Additional POOL port	4	
Storage Manager		74	1.85
	Fix for CLHEP matrix problem	2	
	Remote LCG SE access	6	
	Performance and Storage optimisation	12	
	POOL/ROOT interactivity (Ref<T> plugin)	4	
	Schema Evolution	10	
	Support for automated translation of doubles to floats	2	
	Follow the proposed evolution of SEAL dictionary	8	
	Object storage on the RDBMS layer (Q2/Q3	8	
	On demand Dictionary loading	10	
	Storage Manager scripting interface in Python	2	
	ROOT v4.0 integration	10	
File Catalog		32	0.8
	Support the LCG-2 RLS	4	
	Composite Catalog support	4	
	Support for pluggable replica optimisation	4	
	POOL and LCG replica manager fully interoperate	8	
	Integrate POOL with ARDA provided file catalog	8	
	Participate in ARDA/EGEE requirement gathering	2	
	Support GLOBUS developers in their catalog integration	2	
Collections		37	0.925
	Support for Tokens in AttributeLists	1	
	Provide POOL collection plug-in for ROOT	8	
	Separation of logical and physical collection identification	8	
	Collection Extraction and publishing tools	6	
	Integrate POOL collections with ARDA provided services	8	
	Collection end user interface (eg AIDATupel)	6	
Infrastructure and Testing		38	0.95
	Review documentation structure	4	
	Complete automated data format regression test	4	
	Incorporation of experiment defined test suites	10	
	Evaluate/Move to AppWorks	8	
	Complete move to QmTest	4	
	Parallel build and test machinery	6	
	Extend POOL online tutorial	2	
Total		203	5.1

3.2 Milestones

The following is the list of proposed Level-2 milestones for the POOL project.

3.2.1 POOL RDBMS abstraction layer completed (31 May)

This milestone completes the work on the interface definition of the relational abstraction layer inside POOL. At least two different backend implementations (eg MySQL and Oracle) should exist and be validated against an interface test suite.

3.2.2 RDBMS independency achieved for POOL relational components (30 June)

The relational abstraction layer is used to implement the production versions of the POOL file catalog and POOL collections.

3.2.3 Common interface for Conditions DB defined (30 June)

Conclusion of the review of the conditions database interfaces and the extensions provided by some of the existing implementations.

3.2.4 First release of the POOL Relational Storage Manager (31 August)

Adding the capability to store objects via the standard POOL interface into relational back ends supported by the RDBMS abstraction layer.

3.2.5 Condition DB production release (31 July)

IOV service implemented based on the relational abstraction layer. Payload data storage is using POOL Storage Service or intrinsic storage capabilities of the IOV implementations.

3.2.6 POOL meets scalability requirements (31 October)

Need a level 3 milestone together with the experiments to fully define the scalability requirements.

3.2.7 POOL integrates ROOT 4 (31 October)

Exact timescale determined by outcome of feasibility study performed right now and feedback on backward compatibility issues from experiments.

3.3 Feedback and Priorities from Experiments

ATLAS

1. Features needed for persistence of DC2 event model, CLHEP matrix classes, etc. RD had filed a bug report with POOL and was asking for a

bug fix release of POOL 1.5.0 to be accommodated ASAP in ATLAS releases. I'm assuming we'll be able to say what this means more precisely with the mid-March DC2 ESD/AOD definition from the event model task force.

2. Separation of logical and physical collection identification. (We're ambivalent about the corresponding cataloging.)

3. Tools for keeping files and catalogs synchronized. Note that Ioannis, who knew about this need, has already delivered a command line utility that helps.

4. Support for Refs, or at least file GUIDs, in AttributeLists and collections. (This work is now underway.)

5. Clearer definition of what "relational backend to POOL" means, with prioritization to follow.

LHCb

The LHCb list is in more or less priority order:

- File catalog implementation based on the Alien file catalog
- Support for prioritized list of file catalogs. Separation between read and write catalogs.
- CPU Performance improvement (from x7 to < x2)
- Support for storing doubles as floats. Requires also changes in LCG dictionaries.
- Schema evolution support. We need to be able to read "old" data for at least a year.

CMS

in order of priority
(Q1 = now, Q2 = june, Q3 = sometime before fall)

Condition DB
  IOV interface and implementation Q2
   its relationship with Pool   Q2
  POOL RDBMS back-end   Q3 (Q2?)
  condition catalog (was folders)   Q2/Q3

Catalog
  RLS saga   Q1
  composite catalog   Q1
  bestPfn (see also next point) Q1

File Access
  SURL->TURL   Q1
  generic Root TFile/TSystem plugin   Q1

Root storage manager
  handling missing classes in dictionary Q2 (Q1?)
  performance improvements Q2/Q3
  schema evolution   Q3

Collections:
  integration in CMS (cms job) Q2

```
   collection catalog          Q2/Q3

Bindings other than C++:
  mainly for Catalog, collections and DataService (and conditions?)
  integration in user environment (root/cint, python, unix shell) Q2/Q3

Testing:
  regression tests covering cms "use cases"  Q2

Porting:
  icc      Q2
  ia/64    Q?
  mac-os   Q?
```

4. References

[1] *POOL project plan*, D. Duellmann et al., CERN-LCGAPP-2002-05
http://pool.cern.ch/talks/pool2002workplan.pdf

[2] Applications area internal review report 2003, CERN-LCGAPP-2003-16
http://lcgapp.cern.ch/project/mgmt/rev200310/aa_review_report_2003.doc

[3] Architecture Blueprint RTAG report, T. Wenaus et al., CERN-LCGAPP-2002-09
http://lcgapp.cern.ch/project/blueprint/BlueprintReport-final.doc

[4] Proposal to bring conditions DB into applications area scope, P. Mato, CERN-LCGAPP-2003-08
http://lcgapp.cern.ch/project/mgmt/CONDB_Proposal.doc