Athena/POOL integration

ATLAS Software Workshop

Database Session

David Adams

BNL

March 4, 2003

Contents

Goal

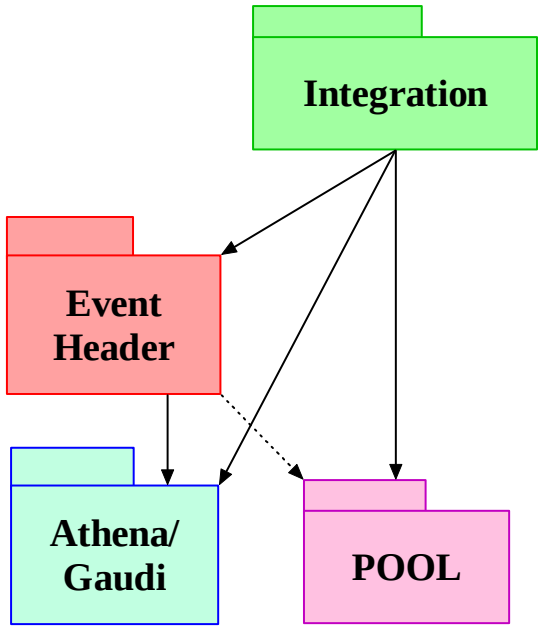
Integrated Athena/POOL will enable:

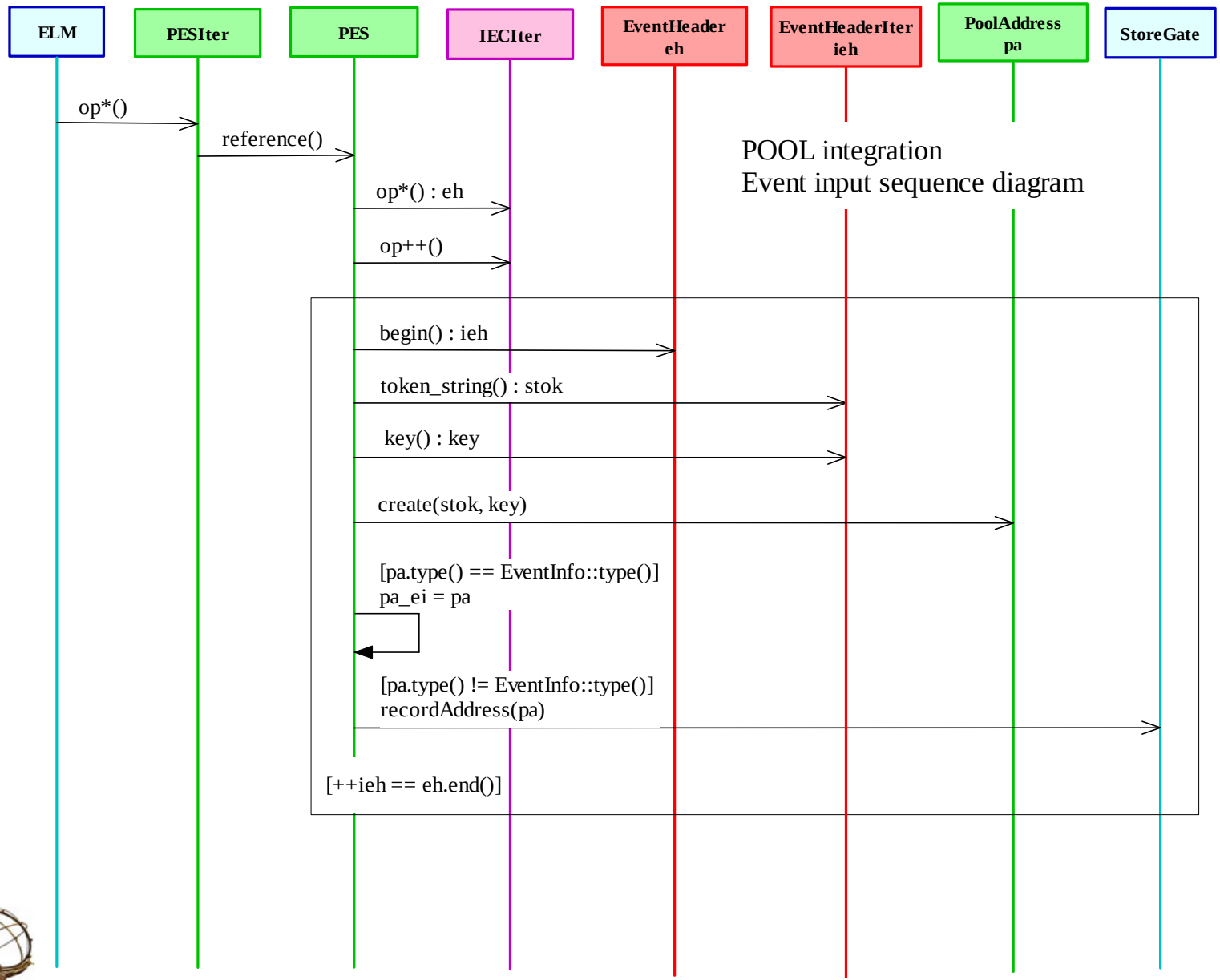
- Writing event data from StoreGate to POOL
- Reading event data from POOL into StoreGate
- User interface similar to that for Oby and AthenaRoot
- Event data organized into ADB EventHeader
- Access via POOL Collection<EventHeader>
- Type definitions in SEAL dictionary
- POOL conversions based on this dictionary

Design

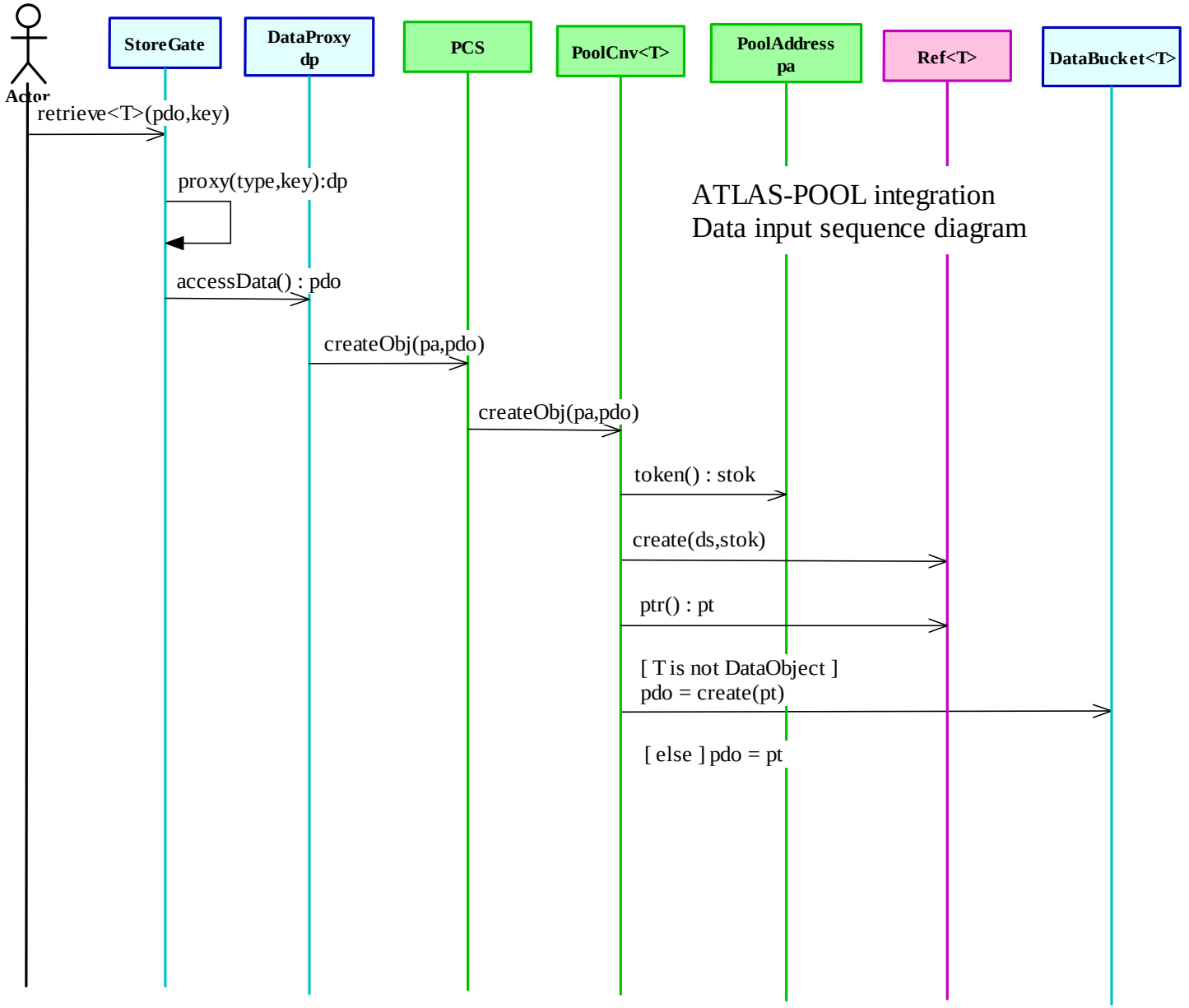
See http://www.usatlas.bnl.gov/~dladams/poolint/

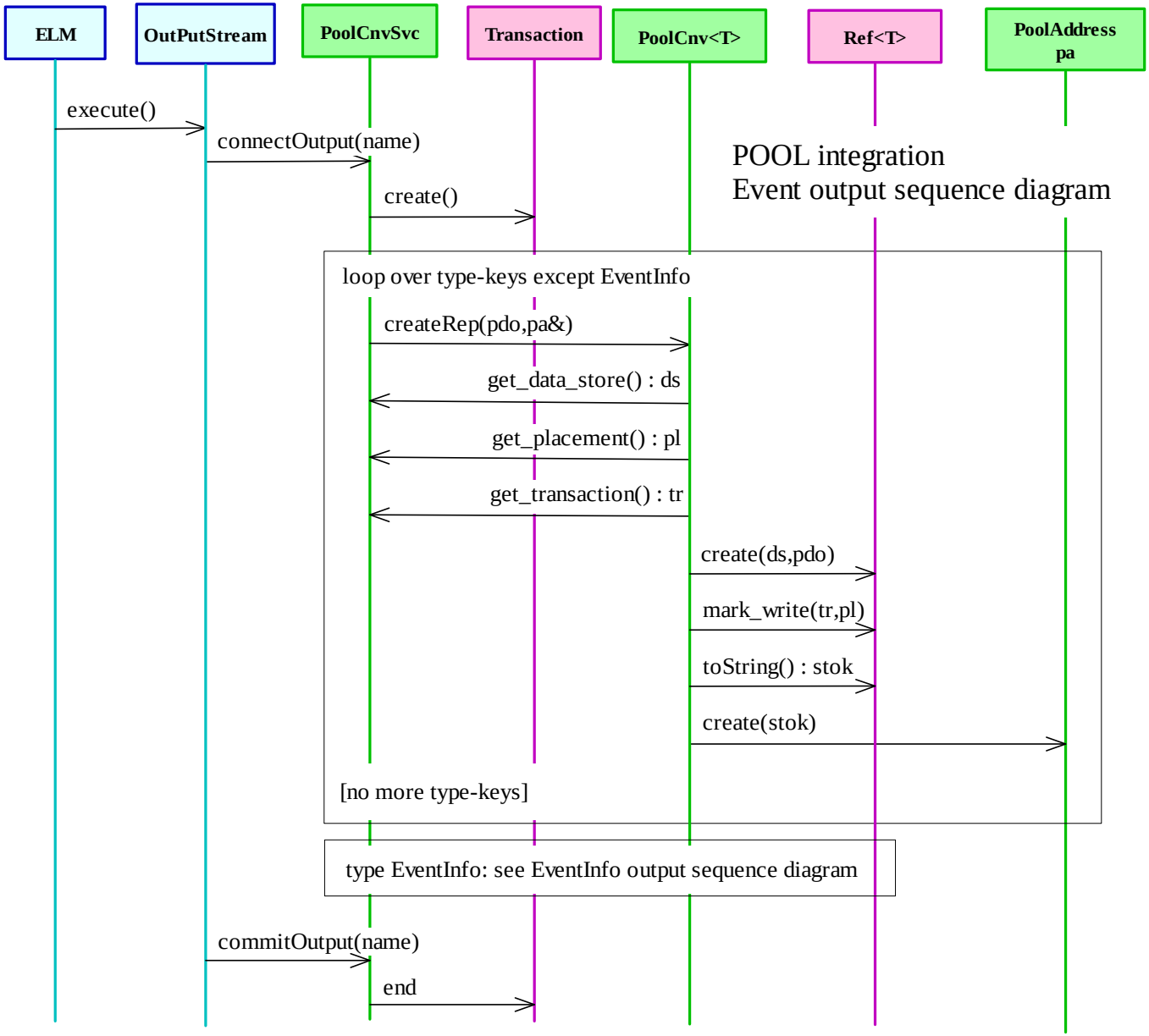
- Identify systems and their dependencies →
  - Color coded
- Sequence diagrams show interactions between components
- Class diagram illustrates dependencies between components

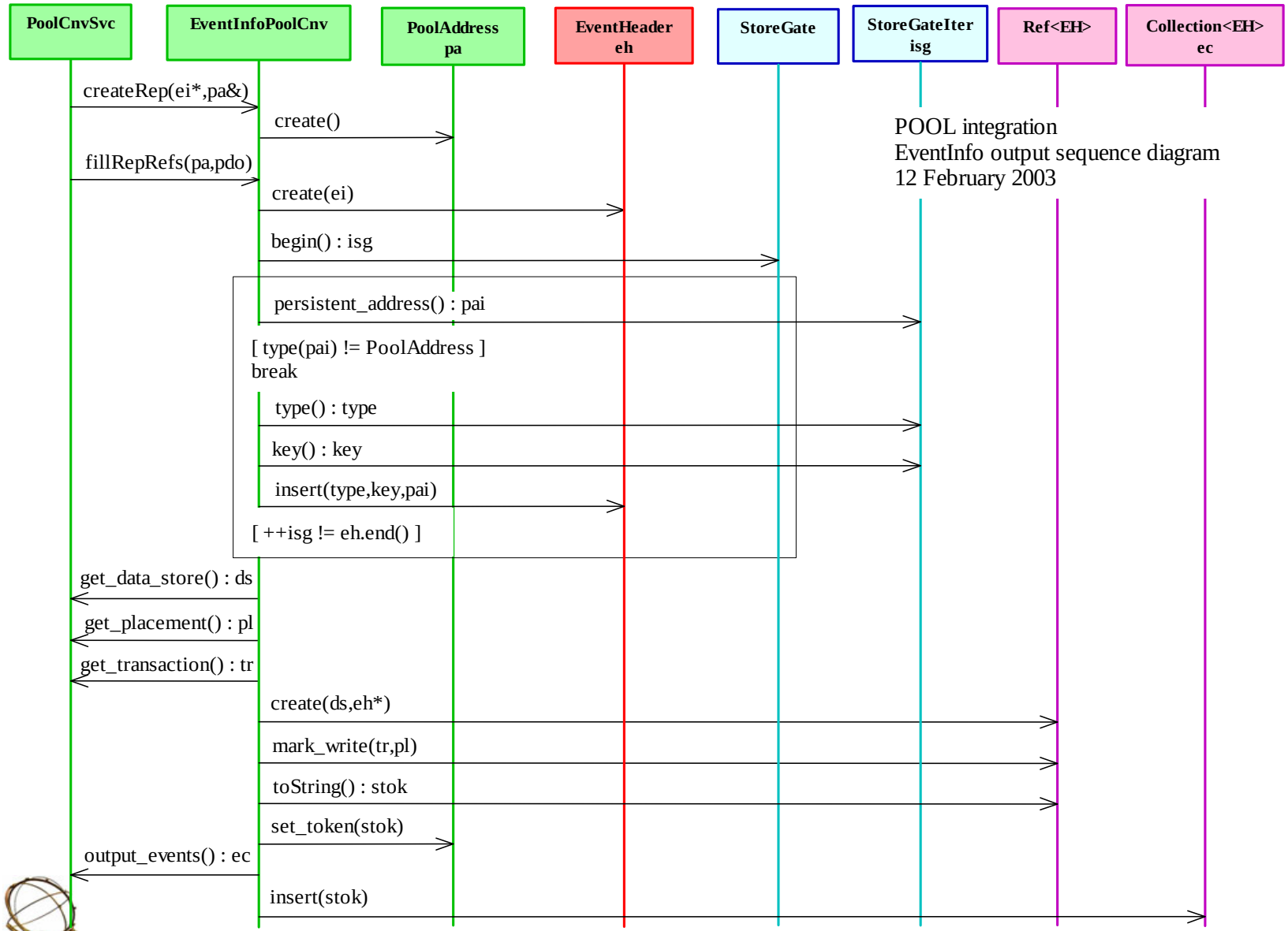


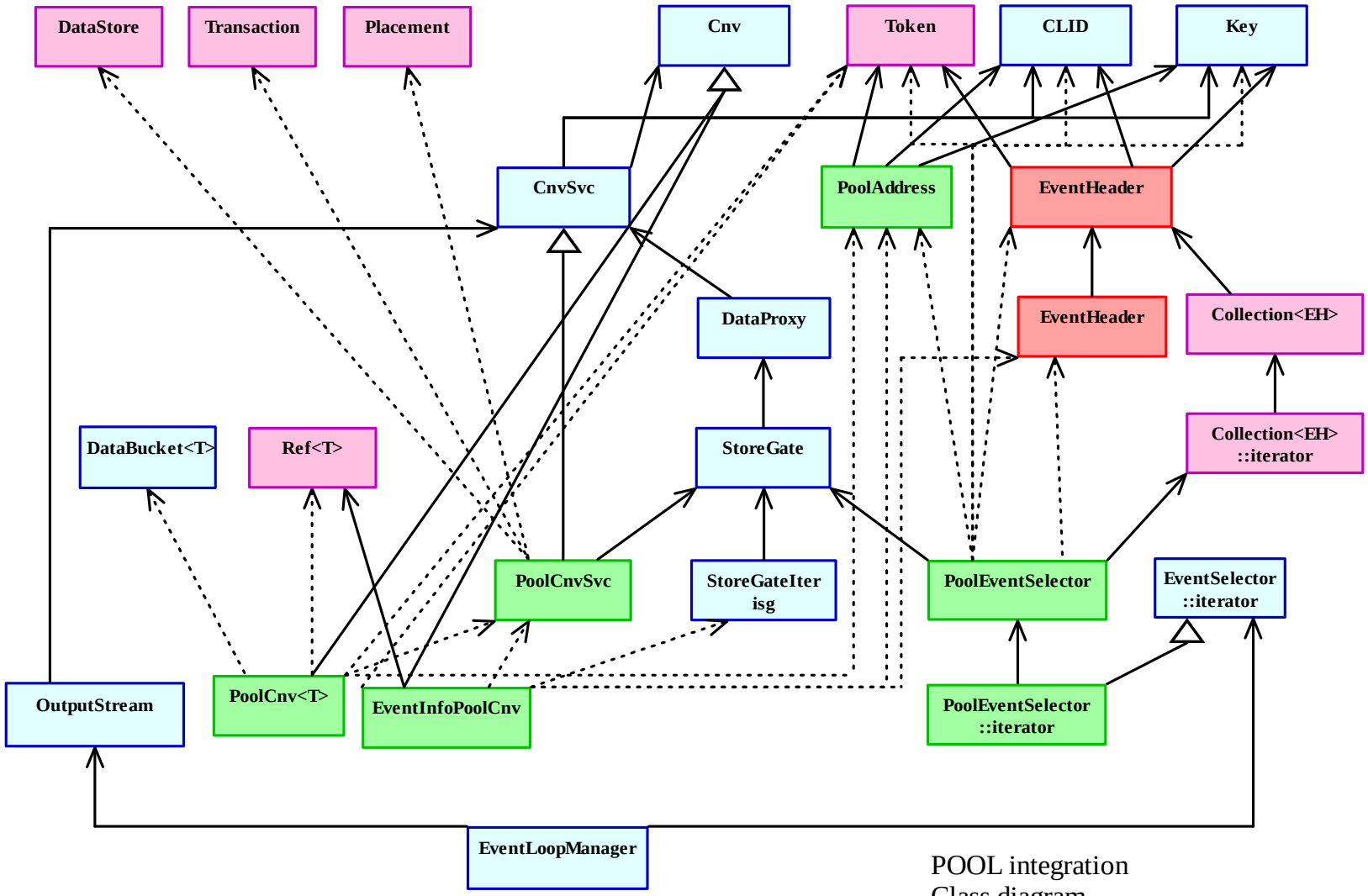


POOL integration
Event input sequence diagram


Actor
StoreGate
DataProxy
dp
PCS
PoolCnv<T>
PoolAddress
pa
Ref<T>
DataBucket<T>
retrieve<T>(pdo,key)
proxy(type,key):dp
accessData() : pdo
createObj(pa,pdo)
createObj(pa,pdo)
token() : stok
create(ds,stok)
ptr() : pt
[T is not DataObject]
pdo = create(pt)
[else] pdo = pt
ATLAS-POOL integration
Data input sequence diagram


ELM
OutPutStream
PoolCnvSvc
Transaction
PoolCnv<T>
Ref<T>
PoolAddress
pa
execute()
connectOutput(name)
create()
POOL integration
Event output sequence diagram
loop over type-keys except EventInfo
createRep(pdo,pa&)
get_data_store() : ds
get_placement() : pl
get_transaction() : tr
create(ds,pdo)
mark_write(tr,pl)
toString() : stok
create(stok)
[no more type-keys]
type EventInfo: see EventInfo output sequence diagram
commitOutput(name)
end


PoolCnvSvc
EventInfoPoolCnv
PoolAddress
pa
EventHeader
eh
StoreGate
StoreGateIter
isg
Ref<EH>
Collection<EH>
ec
createRep(ei*,pa&)
create()
fillRepRefs(pa,pdo)
create(ei)
begin() : isg
persistent_address() : pai
[type(pai) != PoolAddress]
break
type() : type
key() : key
insert(type,key,pai)
[++isg != eh.end()]
get_data_store() : ds
get_placement() : pl
get_transaction() : tr
create(ds,eh*)
mark_write(tr,pl)
toString() : stok
set_token(stok)
output_events() : ec
insert(stok)
POOL integration
EventInfo output sequence diagram
12 February 2003



POOL integration
Class diagram
February 13 2003

Plan

Development has three phases

1. Write and read simple event data
   - See following
2. Full dictionary
   - Automatic population in ATLAS
   - Support complicated types
   - Automatic conversion
3. Support references between data objects

Phase 1

Phase 1 is to first deliver the ability to write and read back

- Collection<EventHeader>
- Including “EventInfo”
- One or two very simple data objects/event

Do not require

- Complicated objects
- Automatic dictionary population
- References between objects

Phase 1 status (cont)

Design – DONE

- 4 access sequence diagrams – DONE
- Initialization sequence diagrams – SKIP?
- Class diagram – DONE

Event Header classes

- EventHeader
- EventHeaderIterator

Phase 1 status (cont)

Integration classes

- PoolAddress - STARTED
- PoolSvc - STARTED
- PoolCnvSvc - STARTED
- BasePoolCnv - STARTED
- PoolCnv<T> - STARTED
- EventInfoPoolCnv

Phase 1 status (cont)

Example/demo programs

- Write POOL file from Athena with one simple object/event - SOON
- Write Collection<SimpleClass>
- Write and read Collection<EventHeader>
- Previous with one data object in event
- Previous with multiple objects taken from StoreGate (Phase 1 complete)

Issues

POOL documentation

- Difficult to understand how to use POOL
- Example programs are a great help
  - (after you find them)

GCC 3.2 required for POOL

- E.g. StoreGate compiles but does not work properly

Xerces 2.1 required for POOL

Issues (cont)

CMT version 13

- Required for GCC 3.2
- Problems working in tcsh

ROOT

- Where to initialize?
  - POOL or ATLAS bridge package
  - Can AthenaRoot and AthenaPool be used together?
- Which ROOT libraries to link?
  - At present AthenaPool has explicit ROOT dependency

Issues (cont)

Providing access to POOL in ATLAS releases

- Alex has created AtlasPool package
  - Will need update for POOL 0.4
- Also need SEAL
  - Same strategy (AtlasSeal)?

EventHeader

- We will write this. OK?

Issues (cont)

POOL

- library dependencies
  - Need all (including test and example)?
- Collection<T>
  - Typed collection not present in 0.3
    - Present in 0.4?
  - Use untyped collection as workaround

POOL 0.4

- Switch right away or after phase 1?

Longer term issues

Dictionary integration (phase 2)

References between objects (phase 3)

POOL ROOT file format

- Data directly accessible in ROOT?
  - Without POOL?
- Or other non-athena applications?

Longer term issues

Long term data storage

- When do we require that data written to POOL be accessible in the far future?

Conditions data

- Can/should we store this data in POOL?

Schedule

Difficult to estimate

- GCC 3.2 problems
- And other problems not yet encountered

Try to get most or all of examples/tests working by the end of this month.

- Working release without all our local patches and workarounds may take longer

Conclusions

POOL appears to offer a sensible persistence strategy for ATLAS

- Share development and maintenance effort with other experiments
- Meets all ATLAS requirements
- No insurmountable problems yet

Phase 2 (dictionary integration)

- discussed by David Q yesterday

Phase 3 (references)

- Needs discussion