Athenaeum allows to access (remote) Athena from (local) JAS.
Any Python script can be send directly from JAS to Athena.
Results (usually in XML) are send back and can be processed within JAS.
Special Python scripts are provided to automatically present standard Athena data within JAS.

J.Hrivnac, LAL/Orsay
CHEP'06/Mumbai, Feb'06
Athenaeum

Using **Java** Analysis **Studio**

as an interface to the

Atlas Offline Framework

➢ **Athenaeum** allows to access (remote) Athena from (local) JAS.
➢ Any (Athena) Python script can be send directly from JAS.
➢ Results (usually in XML) are send back and can be processed within JAS.
➢ Special Python scripts are provided to present Athena data within JAS.
Java Analysis Studio

JAS is a GUI based on FreeHEP library. FreeHEP is Java equivalent of CERNLIB, Root, OpenScientist,...

Most Functionality implemented by Plugins. They can be loaded dynamically (over network).

see http://jas.freehep.org/jas3 for details
Open Connection to Athena

$ athena.py -i -s jobOptions.py

........

XML-RPC server 'atldbdev01.cern.ch:48966' created
method 'process()' registered
Waiting for requests...

On Client
(Any platform with JAS + Athenaeum Plugin)

On Server
(Linux with Athena)

execfile ("InteractiveServer.py")

Server script written
by Atlantis team
(http://cern.ch/atlantis)
Interact with Athena

- Execute a Python script on Athena Server, get results back
- Steer Athena Event Loop from JAS
- Access Cool (Conditions DB) data
- Access Particle data (prototype)
- Get Information about Athena Server environment (loaded dictionaries, paths,...)

Registered Proxies are implementing concrete handling of connection to specific (Athena) functionality.
User can mix Python running within JAS and Python running in a (remote) Athena. Athena Python scripts could be moved to JAS.
Steer Athena Event Loop

Athena interpreted as a set of Records (Events)

Python script executed on each Event
Results analyzed locally

Next Event,...

Output Console

Connecting to http://lxplus003.cern.ch:48966
On http://lxplus003.cern.ch:48966 executing:
theApp.initialize()
Remote Proxy

Registered **Proxies** are implementing concrete handling of connection to specific (Athena) functionality. They are implemented by:
- *Athena Python script* to extract data from Athena
- *JAS wrapper* to present/handle data inside JAS
- *XML schema* to describe data

When implementing pre-defined interfaces from Athenaeum, those Proxies will make themselves automatically available inside JAS system in an organic way.
Construction of Proxy

AthenaObject
- encapsulates the result of the Python script run on the Athena Server.

 ProxyAdapter provides an interactive functionality of AthenaObject in JAS3 Tree.

DefaultTreeNodeAdapter

AthenaObject
- askAthenaTXT(athena: AthenaClient): StringTokenizer
- askAthenaXML(athena: AthenaClient, contextPath: String): Object

ProxyAdapter
- object(athena: AthenaClient): AthenaObject

MyObject.py
- execute on Athena Server

Python
- Python script communicate TXT or XML to AthenaObject. XML is represented by JAXB model.

MyObject

MyObjectAdapter
- Green classes should be implemented for each Athena entity represented in JAS3.

AthenaAdapter
- register(adapter: ProxyAdapter, clazz: Class): void

All AthenaObjects with their ObjectAdapters should be registered in AthenaAdapter.
XML Schema Representations

➢ The same data are shared between different Frameworks/Applications implemented in different languages.
➢ All data representations are derived (generated) from the XML Schema.
Interact with Cool

➢ Open connection to Cool DB
➢ Interpret data (as AIDA NTuples)
➢ Show data as HTML
➢ Show data as XML
➢ Analyse data
➢ Show Python script used to get data
Data can be represented as
- XML
- Objects
- Tree
- (AIDA) NTuples
- HTML

and accessed
- via GUI
- using scripting interface (Java, Python, Pnuts)
- using API (Java, Python)
Work with Cool (2)

- Data can be represented as
  - XML
  - Objects
  - Tree
  - (AIDA) NTuples
  - HTML

- and accessed
  - via GUI
  - using scripting interface (Java, Python, Pnuts)
  - using API (Java, Python)
Only user code + access layer in clients
Data access and standard processing in servers
Orchestration and optimization in virtual servers
Passed data described by common (XML) Schema

➢ Athenaem
➢ SQLTuple/ColMan (see poster 331)
➢ Sequoia (see poster 331)
Architecture Advantages

➢ **Light local client**
  ➢ Running on any platform, any release
  ➢ Fully interactive GUI, scripting and API in several languages
  ➢ Easily extensible by modular plugins

➢ **Server on a powerful machine**, close to data, replicated and hierarchised when useful

➢ **Standard communication protocols**
  ➢ XML-RPC for the Control Flow and small data
  ➢ Eventually performant protocols (JDBC, xrootd,...) for big data
Problems

- **PyAthena** (Python API to Athena)
  - **Incomplete** (only a subset of C++ API is available via Python)
  - **Undocumented** (C++ Doxygen is not enough for documentation of its Python API; it is not easy to guess the meaning of weakly-typed methods; code fragments on Web/Wiky are often out-of-date)
  - **Unstable** (too many things change too often)
  - The **C++ Framework** is **still there**, it just hidden (its problems will pop up from time to time)

- **Data**
  - **No abstract data definition** is available, the actual data model is hidden very deep in the C++ header files forest
    - Athenaeum **XSD Schema** has been written for data passed around; XML, Java, Python and C++ incarnations can be created from them
To Do Next

➢ Generalization for other Monolithic Frameworks
  ➢ there is nothing special about Athena/Gaudi, any Framework with functional XML-RPC server would work fine
➢ Lazy & Compressed data transport (to speed up)
  ➢ XML-aware compression, MPEG-7 compression, binary XML,... can give size down to about 2x compressed Root files size of the same data
➢ User-customizable XSLT
➢ More Proxies (Analysis objects, Generic StoreGate access, ...)
➢ Athena (remotely) startable from Athenaeum (so that user does not have to start the server herself)
➢ Deployment of a network of hierarchical Athena Servers
How To Start

➢ Within CERN AFS:
  ➢ . /afs/cern.ch/sw/java/share/bin/setjdk sun 1.5.0_02
  ➢ /afs/cern.ch/atlas/offline/external/JAS/jas3/jas3

➢ Elsewhere (any platform):
  ➢ Get Java 1.5
  ➢ Get JAS from http://jas.freehep.org/jas3 (Linux, MS, MacOSX, ...)
  ➢ Set Plugin Server (View - Preferences...)
  ➢ Get Plugin (View – Plugin Manager...)

![Preferences](image1.png)

![Plugin Manager](image2.png)
Help

➢ http://home.cern.ch/hrivnac/Activities/Packages/Athenaeum
➢ https://uimon.cern.ch/twiki/bin/view/Atlas/HowToUseJAS
➢ JAS integrated Help (with executable examples)