Control Room Accelerator Physics

Day 1
Introduction to Open XAL
Overview

Open XAL is an extensible application framework for developing accelerator physics applications, scripts and services.
Collaboration

Open XAL is a collaboration among SNS, CSNS, ESS, GANIL, TRIUMF and FRIB.
Official Website

http://xaldev.sourceforge.net
Features

• Open Source collaboration with dozens of developers across several sites
• Pure Java for cross platform development and deployment
• Application Framework for rapidly developing modern applications
• Toolbox of Java packages
• Collection of applications (over four dozen) and services
• EPICS Channel Access support
• Ant based build system independent of IDE
Development Requirements

- Java J2SE 7 with JDK
- Git 1.7.5
- Ant 1.8
Runtime Requirements

- Java J2SE 7
- JRuby 1.6 (for JRuby scripts)
- Jython 2.1 (for Jython scripts)
- EPICS Channel Access client libraries (optional - native Channel Access)
Reference Release Source Code
Anonymous Access

Raw / IDE Independent
git clone http://git.code.sf.net/p/xaldev/openxal

Eclipse Configured
git clone http://git.code.sf.net/p/xaldev/project.eclipse

Xcode Configured
git clone http://git.code.sf.net/p/xaldev/project.xcode
## Primary Constructs

<table>
<thead>
<tr>
<th>Construct</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>Common Open XAL library</td>
</tr>
<tr>
<td>Extension</td>
<td>Optional addition to the Core, Core has no dependency</td>
</tr>
<tr>
<td>Plugin</td>
<td>One of each type of plugin is required by Core at runtime</td>
</tr>
<tr>
<td>Service</td>
<td>Runs continuously, headless, includes extension</td>
</tr>
<tr>
<td>Application</td>
<td>Launched by user, Graphical Interface</td>
</tr>
<tr>
<td>Script</td>
<td>JRuby or Jython based script which may have an graphical interface</td>
</tr>
</tbody>
</table>
Core Overview

- Common packages shared across sites
  - Online Accelerator Physics Model
  - Accelerator Object Graph
  - Channel Access abstraction
  - Database abstraction
  - Messaging
  - Concurrency Dispatch
  - Math packages
Project Layout - Core

- apps
- build
- build.xml
- config
- core
- extensions
- plugins
- scripts
- services
- test

Ant Build File

External Libraries

Images, properties, etc.

Java Source Code

JUnit Tests
Extension Overview

- Mechanism to add capabilities to Open XAL without changing the core
- May depend on core, extensions and plugins
- Core has no dependency on extensions
- Apps and Services may depend on extensions
- May include libraries, resources and source code
  - Included libraries (e.g. jmdns) should be completely wrapped by their extension
- Two types
  - Pure
  - Service
Pure Extension

- Placed under top level extensions directory
- Examples: application (framework), bricks (runtime), fit, scan, solver, service (framework), widget
- Package prefix: `xal.extension.<extension-name>`
Service Extension

- Associated with a service
- A service’s protocol is an extension
- A service’s other supporting code may be an extension
- Placed at extensions directory under its service’s directory
- Package prefix: xal.service.<service-name>
- Example: pvlogger
Project Layout - Pure Extensions

- apps
- build
- build.xml
- config
- core
- extensions
- plugins
- scripts
- services
- test
- application
- bricks
- build.xml
- extlatgen
- fit
- orbit
- scan
- service
- resources
- src
Plugin Overview

- Mechanism to provide custom implementation for some abstracted core components (e.g. Channel)
- May depend on core, extensions and plugins
- Core has runtime only dependency on plugin families
- Apps and Services may depend directly on plugins
- May include libraries, resources and source code
  - Libraries should be completely wrapped (e.g. jca)
- Two types
  - Solitary
  - Family Member
Solitary Plugin

- Only one plugin for a given family may be included
- Core references a plugin family class to be implemented by just one plugin (e.g. channel factory)
- Two source code package prefixes to supply (abstract and implementation)
  - xal.<core package tree>
  - xal.plugin.<plugin-name>
- Example JCA Plugin
Family Member Plugin

- Multiple plugins for a given family may be included
- Core references a plugin family indirectly through configuration files (e.g. database configuration)
- One source code package prefix to supply
  - xal.plugin.<plugin-name>
- Database Adaptor Plugins
  - oracle, mysql
Project Layout - Plugins

- apps
- build
- build.xml
- config
- core
- extensions
- plugins
- scripts
- services
- test
- jca
- mysql
- oracle
- lib
- src
- xal
- plugin
- ca
- Readme.html
Services Overview

• Headless executable
• Runs 24/7
• Supports multiple clients via remote messages using the service framework
Project Layout - Services

- apps
- build
- build.xml
- config
- core
- extensions
- plugins
- scripts
- services
- test

- build.xml
- common.xml
- mpstool
- pvlogger
- extension
- src
- build.xml
Applications Overview

• Executable with a user interface
• Built upon the application framework
• Inherits a common extensible menubar
• Shares a common look and feel
• Model - View - Controller architecture
Scripts Overview

- JRuby or Jython Script
- May have a graphical user interface
- Allows for rapid development and testing
Project Layout - Scripts

- apps
- build
- build.xml
- config
- core
- extensions
- plugins
- scripts
- services
- test
- build.xml
- waveform_monitor
- gui.bricks
- waveform-monitor.rb
Default Build Phases

- Core
- Extensions
  - Service Extensions
  - Plugins
- Service
  - ...
- Application
  - ...


Build Options

- You can build everything at the top level by simply typing: ant
- You can get the available build options by typing: ant help
- Each absolutely independent component has its own build file
  - At component root, simply type: ant
  - You can build an individual application
  - You cannot build an individual extension
Top Level Build Options

> ant help

Buildfile: /Users/t6p/Projects/OpenXAL/Development/Base/openxal/build.xml
help: [echo] Build the XAL project [echo] Usage: ant [ant options] target1 [target2 | target3 | ... ] [echo] [echo] where target(s) can be: [echo] help .............................. Print this message. [echo] all ............................. Build the XAL core, services and applications. Copy scripts to the build directory. [echo] apps ....................... Compile the applications and assemble the jar products. [echo] clean ........................ Clean compiled classes and build products [echo] core ......................... Compile the core XAL classes and assemble the jar products. [echo] doc .......................... Build the javadoc for the core, extensions and plugins. [echo] info .......................... Post build information. [echo] install ..................... Install all build products which allow batch installation. [echo] install-apps .............. Install all apps which allow batch installation. [echo] install-doc ................ Install the javadoc. [echo] install-scripts .......... Install all scripts. [echo] install-services ........ Install all services which allow batch installation. [echo] install-shared ........... Install the core. [echo] jar-resources .............. Archive resources for the core plus all batch enabled applications and services. [echo] purge-build ............... Purge all build products. [echo] purge-install ............. Purge all installed products. [echo] purge-intermediates ...... Purge the build intermediates.
Top Level Build Options Continued

[echo] purge-shared-intermediates . Purge the shared build intermediates.  [echo] run-tests ................. Build and run unit tests.  [echo] scripts .................... Copy scripts to the build directory.  [echo] services .................... Compile the services and assemble the jar products.  [echo] shared ...................... Build the shared library including the core and any extensions and plugins.  [echo] standalone-apps ........ Build the applications which allow batch building and assemble the jar products as standalone applications.  [echo] standalone-services ..... Build the services which allow batch building and assemble the jar products as standalone services.