# USER'S MANUAL

*DIMSIM Portal and CIMA Extensions*

**Australian ResearCH Enabling enviRonment**
**- ARCHER Project**

December 2008

# Revision Sheet

| Release No. | Date | Revision Description |
|---|---|---|
| Rev. 0 | 5/12/08 | Initial Draft |
| Rev. 1 | 18/12/08 | Release 1.0 |
| | | |
| | | |
| | | |
| | | |
| | | |

# USER'S MANUAL

## TABLE OF CONTENTS

Page #

# 1.0 GENERAL INFORMATION

# 1.0 GENERAL INFORMATION

## 1.1 Background

Australian Federal Government's *National Collaborative Research Infrastructure Strategy (NCRIS)* Roadmap for 2006 recognizes e-research and availability of *platforms for collaboration* as vital tools for is vital for Australian researchers. According to NCRIS: *seamless access enables researchers to carry out their research more creatively, efficiently and collaboratively across long distances, regardless of location and time, and disseminate their research outcomes with greater effect.*

A core requirement for many of the NCRIS capabilities is the ability to capture data coming from a range of instruments. Current and future user requirements in area of scientific instrument data capture can be summarized as below:

1. Instrument monitoring in real-time (state plus images)
2. Instrument programmatic control (will vary by instrument type and owner requirements)
3. Data acquisition
4. Ability to add new instruments to system
5. Secure instrument management/control/acquisition

Archer project's *Distributed Instrument and Multi-Sensor Integrated Middleware* (DIMSIM) work package aims to identify infrastructure requirements and develop software solutions to the above user requirements.

## 1.2 Overview

Dimsim work package leverages existing work in the area of data collection and monitoring. In particular

1. CIMA : JCU (Ian Atkinson's group) and Indiana (Ric McMullen's group) Universities
2. JAINIS work at JCU as part of the DART project
3. Instrument management work at University of Sydney by Peter Turner's group

### 1.2.1 Common Instrument Middleware Architecture (CIMA)

Research instruments vary widely in their design, construction, and interfaces. The Common Instrument Middleware Architecture is designed to provide a single virtualisation layer to hide this complexity, and offer a relatively simple Web service interface to the rest of the data pipeline. CIMA was initially proposed and used by Ric McMullen's group at the Indiana University. Ian Atkinson's group at James Cook University recognized that adapting CIMA to the Australian context, in particular to reef sensor grids and X-ray crystallography labs at leading universities in Australia will further NCRIS goal to develop *platforms for collaboration*.

CIMA framework is focused primarily on providing a virtualisation layer to capture data from a variety of instruments. Key features of CIMA include

- linking network instruments with remote data store
- exchange messages using XML parcels
- subscription based, clients request data from producers

- key components : Instrument Representative, Data Manager & Plug-ins

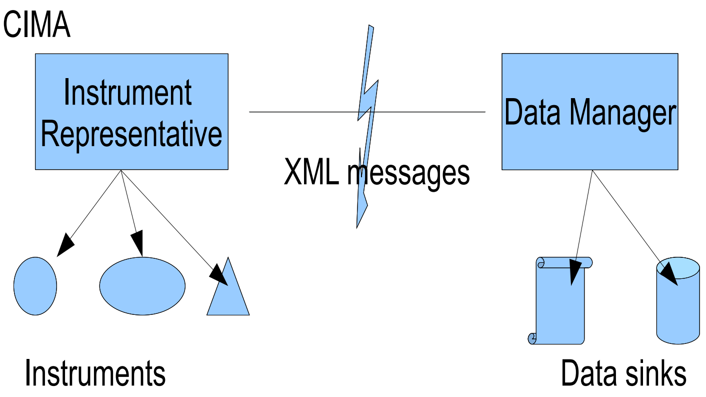

*Figure 1: CIMA - Original Concept*

### 1.2.1 CIMA redesign

Concurrent to the Dimsim work package, JCU and Univ of Sydney collaborated to design a CIMA implementation that was modular and intuitive to use.

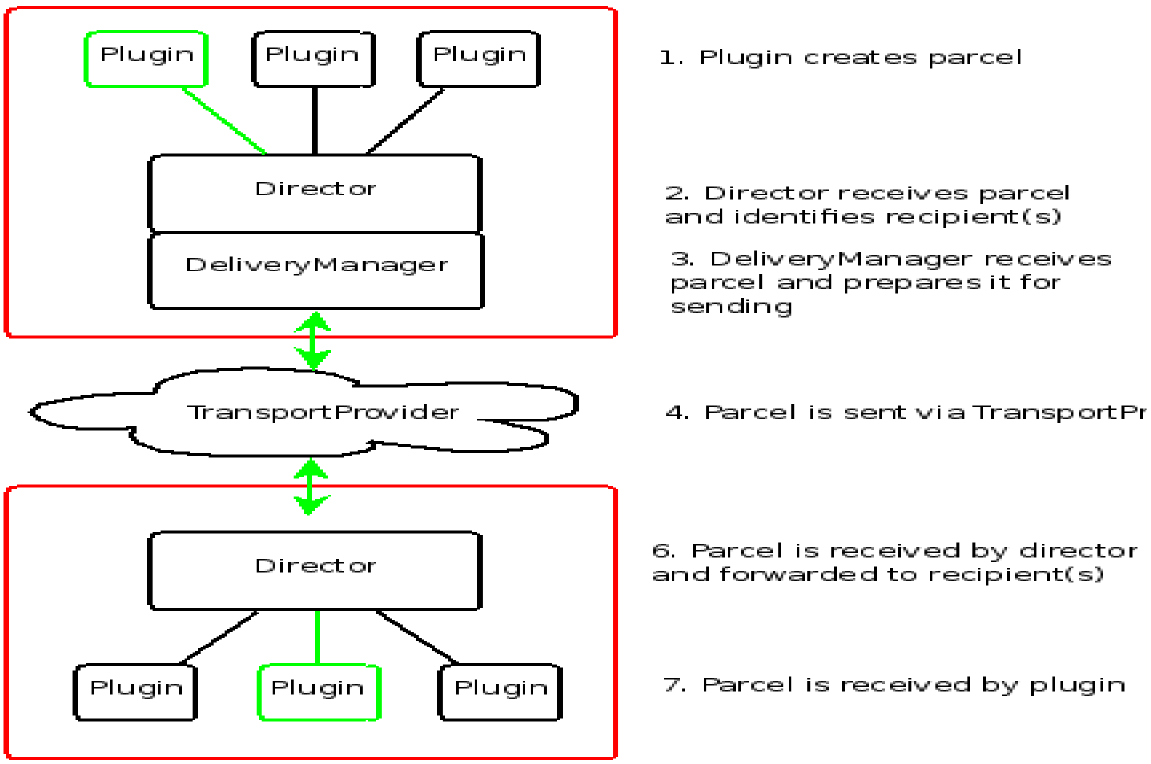
Plugin
Plugin
Plugin
1. Plugin creates parcel
Director
2. Director receives parcel and identifies recipient(s)
DeliveryManager
3. DeliveryManager receives parcel and prepares it for sending
TransportProvider
4. Parcel is sent via TransportPr
Director
6. Parcel is received by director and forwarded to recipient(s)
Plugin
Plugin
Plugin
7. Parcel is received by plugin

*Figure 2: CIMA redesign : Modular approach to parcel management*

Figure 2 describes the flow of data from producer to consumer. Comparing with Figure 1.(Cima Original Concept) we find that the new CIMA implementation provides us with the following advantages:

- Parcel handling is modularized. Each module (Director, Transport Provider, Delivery Manager) encapsulates a particular phase in parcel management and is independent of other modules.

- Modules are pluggable. For example the default transport provider shipped with CIMA is Memory Transport. Dimsim deliverables includes a SOAP transport that replaces it.

API doc for the new CIMA implementation is provided with Dimsim deliverables. More details on CIMA redesign can be found at JCU https://www.hpc.jcu.edu.au/projects/DIMSIM/wiki/ReDesign. The source code for this work is not covered under the Archer project grant. For those interested in reviewing the code, please refer to JCU repository https://www.hpc.jcu.edu.au/projects/DIMSIM/browser/CIMA.

### 1.2.2 CIMA : Quality Issues

While CIMA provides a scalable, instrument and data sink neutral mechanism for distributed data capture, it lacks support for robustness, reliability and security that is expected from an automated data collection application. In particular CIMA applications suffer from the following problems that would stop it from being considered *production ready.*

- No guarantee of reliable data transfer
  - parcel dropouts, arriving out of sequence etc.,
- network issues
  - remote application can be down, but clients will never know.
- No security
  - any one can request subscription
- No access to legacy/historical data
  - drop-out/lost data cannot be re-queried

## 1.3 Dimsim Deliverables for Archer Project

Design and development work carried as part of Dimsim work package for Archer project include :

- support/extension modules to address CIMA quality issues raised in section 1.2.2
- CIMA plug-ins to demonstrate use of Dimsim extensions in the protein crystallography area.
- web based portal for display of data captured using Dimsim extensions.

### 1.3.1 Archer project extensions to CIMA

DIMSIM enables CIMA applications to be production ready by providing support modules that extend/enhance CIMA functions. Dimsim extensions to CIMA include :

- Buffer module :
  - configurable data cache at producer end
    - memory, file or database based
    - Ring buffer, can be replaced with list, queue or other data structures
    - remote clients can query using buffer plug-in

  - ① client buffers to hold data, usage examples include
    - data sink not ready or network link down
    - while waiting to resolve out-of-sync parcels
- Security module :
  - ① access restriction
    - granularity : per plug-in, end-point URL
  - ① user authentication
- Restart & Recovery module :
  - ① State Persistence
    - restore previous subscription on restart
  - ① Local cache Persistence
    - persist local cache for recovery on re-start

### 1.3.2 DIMSIM : CIMA plug-ins to demonstrate extensions

CIMA end user plug-ins provided with Dimsim are tailored to protein crystallography users. Plug-ins and functions provided include:

- SRB plug-in:
  - ① Writes to SRB store
  - ① recognize changes to user. project and sample
  - ① error-processing
    - restore local cache on restart
    - flag error-records for manual process
  - ① Protein crystallography image conversion plug-ins
    - Convert Rigaku OSC and IMG image files to Jpeg
  - ① Protein crystallography image capture
    - Capture x-ray images by monitoring Rigaku CrystalClear configuration files
  - ① Automate Multi-user data collection
    - by monitoring User Administration database for CrystalClear V1.4.

### 1.3.3 DIMSIM : Portal

A simple web portal that acts as client to a Dimsim producer instance and displays captured images along with sensor and crystal feeds is provided for demonstration The portal uses Django/python for back-end processing and extJS/javascript for client display.

## 1.4 Organization of the Manual

This user manual is divided into three parts.

The first part provides an overview of the Dimsim project.

Second part provides basic setup information for users. For a detailed setup and configuration, refer to System Administration and Developer manual for Dimsim

The last section provides a set of trouble shooting tips to resolve most common issues arising from deployment of Dimsim modules.

## 1.5 Acronyms and Abbreviations

| | |
|---|---|
| Archer | Australian ResearCH Enabling enviRonment project |
| CIMA | Common Instrument Middleware Architecture |

| | |
|---|---|
| DIMSIM | *Distributed Instrument and Multi-Sensor Integrated Middleware* |
| Indiana | Indiana University |
| JCU | James Cook University |
| Monash | Monash University |
| SDSC | San Diego SuperComputer Center |
| SRB | SDSC Storage Resource Broker |
| Usyd | University of Sydney |

# 2.0 SYSTEM SET-UP

# 2.0 SYSTEM SETUP

Source code for Dimsim modules can be found on Google code at http://dimsim.googlecode.com. Non-members can check-out code from dimsim repository in read only mode. To upload your changes to dimsim google code repository please send an email to the project owners listed at http://code.google.com/p/dimsim/

## 2.1 Portal Installation

Portal installation of Dismim portal and the Labjack server for sensor readings. The portal and labjack server was developed using Django web framework. For the portal to work properly, django modules must be installed correctly. For details on django installation refer to django installation guide at http://docs.djangoproject.com/en/dev/intro/install/

### 2.1.1 Labjack server for Sensor readings

1. Pre-requisites :
    1. Sensors are connected to a USB labjack attached to the PC. For successful probing, labjack libraries for the corresponding OS and version must be installed in */usr/local/lib* folder. A sample library l*iblabjack.so* that works for linux kernel version 2.6.21-1.3194.fc7 can be found in the 'support-libs ' folder of google code dimsim repository. Copy this file to */usr/local/lib/* folder.
    2. Python libraries must be installed. The code was tested successfully with Python 2.5
    3. Django web framework must be installed.
       Refer to installation guide http://docs.djangoproject.com/en/dev/intro/install/
2. Download Labjack code from dimsim google code svn repository using the following command:
   *svn checkout http://dimsim.googlecode.com/svn/trunk/labjack*
3. Modify name of labjack library if required by editing file labjack_interface.py
4. Deploy Labjack server to Apache as per instructions given in README file.
5. To test the laback server before deploying it on Apache server, run the following command
    1. change directory to the root of the labjack code, (i.e., dir containing manage.py file)
    2. type *python manage.py runserver 0.0.0.0:18081 --settings=settings* and type enter
    3. open your browser and test the labjack server
        1. http://hostURL:18081/temperature for Temperature
        2. http://hostURL:18081/humidity for humidity

### 2.1.2 Dimsim Portal

- Pre-requisites : The following applications must be installed for successful display of portal
  Labjack portal : see section 2.1.1
  Dismim Producer plug-ins : see section 2.2
  Python libraries must be installed. The code was tested successfully with Python 2.5
  Python libraries for Zolera Soap ZSI : http://pywebsvcs.sourceforge.net/zsi.html
  Django web framework must be installed.
  - Refer to installation guide http://docs.djangoproject.com/en/dev/intro/install/
  Video capture program motion must be installed and configured. Refer to section 2.1.3
- Download portal code from svn repository at using the following command.
  *svn checkout http://dimsim.googlecode.com/svn/trunk/portal*

- Edit *portal/settings.py* file and modify if required the following
  MEDIA_URL : check that the host name points to your installation.

INSTALLED_APPS : default value 'portal.portal' must be modified if the portal is checked out to a folder different than 'portal'. The default svn checkout command above will check portal out to 'portal' folder, hence no change is required.
CRYSTALCAM_URL : check that the URL points to URL streaming crystal camera's output
DIMSIM_HOST_URL : ensure that Dimsim source plug-ins are up and running at this URL
PORTAL_HOST_URL : leave the default host name localhost, however modify the port number if required.
SOURCE_ID : refers to the dimsim source plug-in id. Modify this to reflect your source plug-in id.
SOURCEBuffer_ID : leave the default value unless the plugin name is changed in the dimsim source-plugin configuration file.

- To test the portal before deploying it on Apache server, run the following command
  change directory to the root of the portal code, (i.e., dir containing manage.py file)
  type *python manage.py runserver 0.0.0.0:18080 --settings=settings* and type enter
  open your browser and test the portal by pointing to the http://hostURL:18080

- Latest Crystallography image is retrieved from folder media/images/latestCrystallography.jpg. This file must be removed and sym linked to the actual location containing the latest image. For default setup, this will be the location /opt/tomcat/temp/latestCrystallography.jpg. This assumes that the portal and dimsim source plug-ins are running on the same machine and that the source plug-ins are deployed on tomcat running from folder /opt/tomcat.

- Create a apache config file in folder /etc/httpd/conf.d/dimsimPortal.conf with the following lines:

```
<Location "/portal">
    SetHandler python-program
    PythonHandler django.core.handlers.modpython
    PythonPath "['PORTAL_CODE_PATH',
      'PORTAL_CODE_PATH/portal',
      'PORTAL_CODE_PATH/portal/portal',
      'PORTAL_CODE_PATH/portal/sensors',
      '/usr/lib/python2.5/site-packages/django'] + sys.path"
    SetEnv DJANGO_SETTINGS_MODULE portal.settings
    PythonOption django.root /portal
    PythonDebug On
    PythonInterpreter portal
</Location>
```

where PORTAL_CODE_PATHrefers to the root path of your portal code

### 2.1.3 Portal : Misc Configuration

1. The portal expects a streaming video at URL specified by field CRYSTALCAM_URL. On Linux machines it is recommended to use 'motion' software. More details on motion and configuring cameras refer to motion website at http://www.lavrsen.dk/twiki/bin/view/Motion/WebHome.

2. A sample configuration file for CrystalCamera is provided in the support-config folder of the google code repository for dimsim. Please note, this values specified in this file override default values provided in the global motion.conf file for this particular camera thread. To use this file,
   1. Download and copy the crystalcam.conf to /usr/local/etc folder on machine running motion
   2. inform motion to use crystalcam.conf. Append to the end of global motion.conf file :

1. thread /usr/local/etc/crystalcam.conf
3. Restart motion in daemon mode. Open a browser and point URL to http://hostURL:8003. Verify that the crystal camera image is streaming.

## 2.2 Dimsim Crystallography Plug-ins

Dimsim plug-ins for protein crystallography users is configured in the Crystallography folder of the Dimsim repository on google code. To use them without any modification, checkout the crystallography folder and follow the instructions in README file.

### 2.2.1 Pre-requisite :

1. Dimsim uses Maven (http://maven.apache.org/) for Project and Library management. To ensure that dismim war files are produced without errors, ensure that Maven is installed properly.

2. When you run the compilation for the first time, Dimsim support libraries for the core, schema , jaxws, buffer and plugin artifacts will not be available and maven will error out with instructions for installing these artifacts in your local repository. Download the files from the support-libs folder of the dimsim repository on google code and install them using instructions provided by maven.

3. The war files for producer and consumer plugins must be deployed on a suitable servlet container. Tomcat is the preferred and recommended servlet container. The code was tested with Tomcat version 6.0.16

4. For default setup, copy AdminDatabase_140.xml file to /mnt/ctrlpc/AdminDatabase_140.xml. This file defines a single user called Monash and sets his CrystalClear script configuration directory to /mnt/ctrlpcscripts. Make sure that samba mapping for the Script dir containing CrystalClear's SessionScript.scp file is mapped to this folder.

5. Image File location is dependent upon the value provided by the SessionScript.scp file. For Image Capture to work properly, Drive mappings and folder mappings from Windows to Linux must be properly configured in the sourcePlugins.xml file.
   For example, if the SessionScript.scp file sets the ImageDir as D:\abc\Images and this folder is mapped to Linux share /mnt/Images, then the configuration information for bean Rigaku SCPMultiUser in sourcePlugins.xml file must be set as follows
   1. Property dirMap must be set to <entry key="D:" value="/mnt"/>
   2. Property replaceMap must be set as

```
     <map>
          <entry key="\\abc\\Images" value="Images"/>
          <entry key="\" value="/"/>
    </map>
```

### 2.2.2 Bootstrapping Tomcat

To enable Source and Consumer plug-ins to restart at boot time, it is recommended that tomcat installation be started at boot time. On linux machines this adding a config file to /etc/init.d folder and then running it at different system levels *copy the code below to file /etc/init.d/tomcat*

```
RETVAL=$?
CATALINA_HOME="/opt/tomcat"
export JAVA_HOME=/opt/jdk

case "$1" in
 start)
        if [ -f $CATALINA_HOME/bin/startup.sh ];
          then
            echo $"Starting Tomcat"
            /bin/su tomcat $CATALINA_HOME/bin/startup.sh
        fi
        ;;
 stop)
        if [ -f $CATALINA_HOME/bin/shutdown.sh ];
          then
            echo $"Stopping Tomcat"
            /bin/su tomcat $CATALINA_HOME/bin/shutdown.sh
        fi
        ;;
 *)
        echo $"Usage: $0 {start|stop}"
        exit 1
        ;;
esac

exit $RETVAL
```

symlink the above file to various service levels. A sample output from a machine running dimsim is given below :

```
[dimsim@jainis rc.d]$ ls -al */* | grep tomcat
-rwxr-xr-x 1 root root   648 2008-11-30 12:41 init.d/tomcat
lrwxrwxrwx 1 root root    18 2008-11-30 12:33 rc1.d/K99tomcat -> /etc/init.d/tomcat
lrwxrwxrwx 1 root root    18 2008-11-30 12:33 rc2.d/S99tomcat -> /etc/init.d/tomcat
lrwxrwxrwx 1 root root    18 2008-11-30 12:38 rc3.d/S99tomcat -> /etc/init.d/tomcat
lrwxrwxrwx 1 root root    18 2008-11-30 12:38 rc4.d/S99tomcat -> /etc/init.d/tomcat
lrwxrwxrwx 1 root root    18 2008-11-30 12:38 rc5.d/S99tomcat -> /etc/init.d/tomcat
```

### 2.2.3 Cron jobs for monitoring Plug-in subscription

The cron jobs for monitoring client subscriptions to producers are built-into the configuration for client plug-ins. Class au.edu.archer.dimsim.plugins.AutoSubscribe is configured to monitor subscriptions from SRBConsumer to Rigaku_Monash and Buffer Plugins. Once every five minutes AutoSubscribe object queries and validates the session ids with Producer instance. If the validation fails, it resends a new subscription request linking SRBConsumer with producer and buffer plug-ins.

# 3.0 Troubleshooting

# 3.0 TROUBLESHOOTING

## 3.1 Portal Display

### 3.1.1 No camera image

1. Verify that motion software is running, try `ps -ef | grep motion` to see motion is running
2. Verify that the crystal camera is on and connected to the PC
3. Point browser to CrystalCAM_URL and see if the video feed is on

### 3.1.2 No sensor data

1. Verify that labjack server is running. Probe labjack URL http://hostURL/labjack/temperature
2. Verify that labjack library is loaded properly. Run labjack server in standalone mode to see if there are any errors reported while loading labjack library
3. Make sure Labjack is connected to the PC and the sensor is connected to the labjack.

### 3.1.3 No x-ray image/ project information displayed

1. Make sure that an experiment is running
2. Default configuration
3. Make sure that Dimsim producer instance is running. Probe tomcat URL http://hostURL:8080/Cima_Webapp

### 3.1.4 x-ray image not refreshing

1. Make sure an experiment is running
2. Disable browser cache.
3. Latest Crystallography image is retrieved from file media/images/latestCrystallography.jpg. This file must be removed and sym linked to the actual location containing the latest image. For default setup, this will be the location /opt/tomcat/temp/latestCrystallography.jpg. This assumes that the portal and dimsim source plug-ins are running on the same machine and that the source plug-ins are deployed on tomcat running from folder /opt/tomcat.

## 3.2 Dimsim Plug-ins

While the portal provides a visual display, the errors related to Dismim plug-ins can be identified only by reviewing the log files and SRB data store. Dimsim uses Log4j module and by adjusting the log levels in log4j.properties files, the user can vary the amount of log and debug information printed in the log files. For Tomcat container, the ideal place to look for log messages is the catalina.out file in the logs folder.

### 3.2.1 Source Plug-ins

#### 3.2.1.1 Rigaku Plugin not started

1. Ensure that the AdminDatabase_140.xml file is properly configured for each user plug-in and that it is found in the location configured for sourcePlugin bean adminDBMonitor.
2. Review log file to see that the sourcePlugin.xml is parsed correctly and that there are no errors.

#### 3.2.1.2 Jpeg Image not created

- Ensure that the user running the container has write permissions to the location specified by the defaultFileName property of bean RigakuJPGParcelCreator in sourcePlugins.xml

### 3.2.1 Consumer Plug-ins

#### 3.2.2.1 SRB plug-in not started

- Review log file to see that SRB connection is successful. If the authentication information is valid, then restart container.