System Administration & Developers MANUAL

DIMSIM : CIMA Extensions

<u>Australian ResearCH Enabling enviRonment</u>
- ARCHER Project

December 2008

Revision Sheet

Release No.	Date	Revision Description
Rev. 0	5/12/08	Initial Draft
Rev. 1	22/12/08	Release 1.0

System Administrator &
Developer MANUAL

TABLE OF CONTENTS

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.

1.1.1 Intended Audience

The intended audience for this manual are System Administrators who are likely to be asked to install and configure Dimsim in a new environment. In addition Developers intending to develop new CIMA plug-ins can also review this document to gain understanding of plug-in configuration and usage.

1.1.3 Related documents

1. Readers are invited to review Dimsim User Manual to gain an understanding of Dimsim and how it relates to CIMA

2. API docs for CIMA released with Dimsim.

3. CIMA redesign documents at https://www.hpc.jcu.edu.au/projects/DIMSIM/wiki/ReDesign

1.2 Overview

1.2.1 Common Instrument Middleware Architecture (CIMA)

Figure 1 depicts the flow of information from producer to the consumer. Dimsim and CIMA treat each object depicted in the diagram as Spring Bean.

Studying the configuration parameters for each bean defined in a Dimsim application is the preferred and most efficient way to maintain and modify Dimsim.

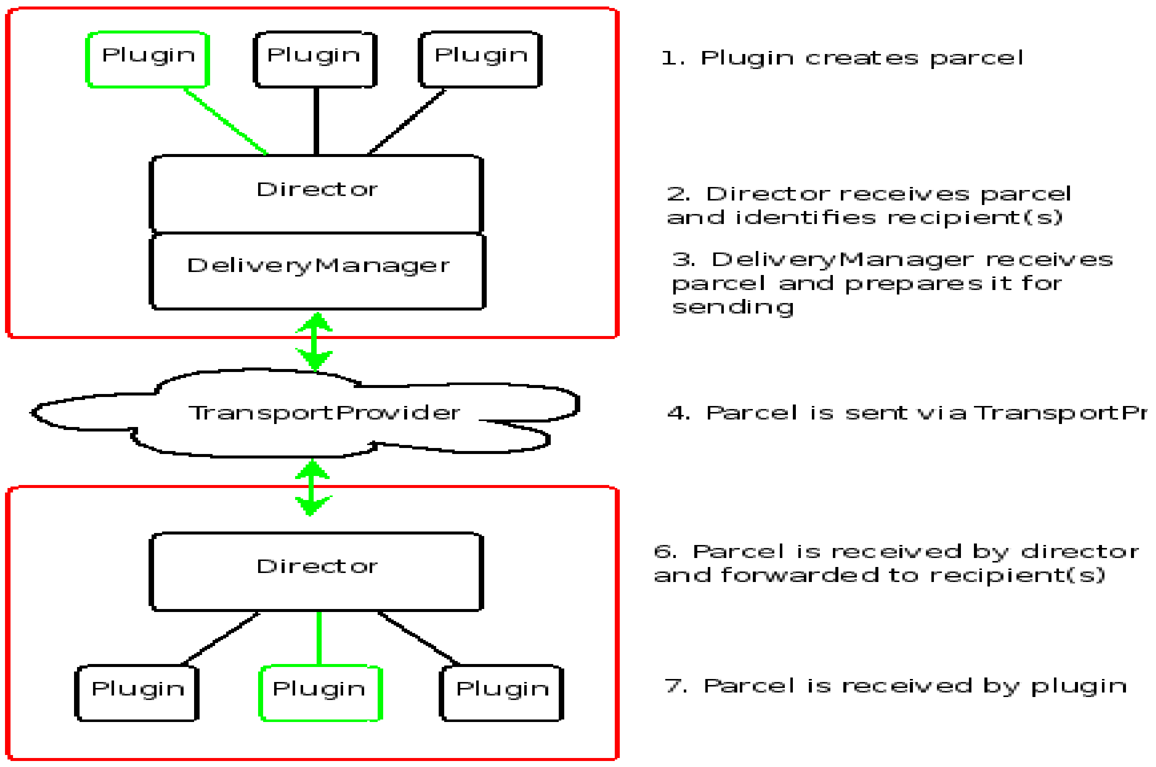
Plugin
Plugin
Plugin
Director
DeliveryManager
TransportProvider
Director
Plugin
Plugin
Plugin
1. Plugin creates parcel
2. Director receives parcel and identifies recipient(s)
3. DeliveryManager receives parcel and prepares it for sending
4. Parcel is sent via TransportPr
6. Parcel is received by director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

Figure 1: CIMA redesign : Modular approach to parcel management

1.4 Organization of the Manual

This manual is organized into three parts.

- Part 1 provides a brief summary and general information
- Part 2 discusses configuration parameters for beans representing Dimsim modules.
- Part 3 provides starting tips for developers keen on designing their own plug-ins.

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 DIMSIM Default Configuration Parameters

2.0 PLUG-IN PARAMETERS

Dimsim Beans and associated parameters for Monash crystallography are defined in the xml configuration files sourcePlugins.xml and consumerPlugins.xml. These two files are found under the crystallography folder of the google code repository for dimsim. The files are symbolic links, hence follow the links if you are browsing the repository using a browser.

2.1 Source/Producer Beans

2.1.1 Security Beans

The main bean for restricting access to Dimsim producers is the endpointSecurity bean. A sample definition is given below :

```
<bean id="endPointSecurity"
        class="au.edu.archer.dimsim.security.impl.EndpointSubscriptionSecurityModel"
        scope="singleton">
        <constructor-arg type="int" value="87"/>
         <constructor-arg>
          <set>
                <set>
                     <value>Cima_Portal</value>
                     <value>*</value>
                     <value>Rigaku_Monash</value>
                </set>
                 .........
           <!-- The following allows any cima client to subscribe and get data from LabJack Plug-in -->
                <set> <value>*</value><value>*</value><value>Labjack_Plugin</value> </set>
          </set>
          </constructor-arg>
     </bean>
```

EndpointSubscriptionSecurityModel class takes in a constructor argument a set of tuples representing the tuple <consumer_id, consumer_url, source_id>. Each field of this tuple can be replaced with a generic value '*'.
CIMA requires a session be established between producers and consumers before any data is sent. The session is established by consumers sending in a subscription request. Dimsim traps this subscription request, validates the tuple <consumer_id consumer_url, requested soure_id> against the pre-configured values provided to the endpointSecurity bean. If the tuple does not match a pre-configured value, subscription request is denied. Security is thus enforced.
Note:- Tuples can be added or removed from the EndpointSubscriptionSecurityModel object dynamically at run time.. However for the default setup such a situation does not arise.

2.1.2 Buffer related Beans

```
<bean
      id="bufferManager"
      class="au.edu.archer.dimsim.buffer.pool.manager.impl.MemoryBufferManager"
      scope="singleton">
 </bean>       <!-- default buffer in MemoryBufferManager is set to RingBuffer -->
 <bean
      id="deliveryBuffer"
```

```
            class="au.edu.archer.dimsim.buffer.impl.ListDeliveryBuffer"
            scope="singleton">
            <constructor-arg ref="bufferManager"/>
            <constructor-arg type="int" value="87"/>
      </bean>
      <bean
            id="ParcelBufferedEventListener"
            class="au.edu.archer.dimsim.buffer.event.handler.ParcelBufferedEventHandler"
            scope="singleton">
      </bean><!-- Make Sure the delivery Strategy does not re-Buffer -->
      <bean
            id="bufferPlugin"
            class="au.edu.archer.dimsim.buffer.plugin.impl.ListBufferPlugin"
            scope="singleton"
            parent="basicProducer">
            <constructor-arg type="java.lang.String" value="Buffer_Plugin"/>
            <constructor-arg ref="deliveryBuffer"/>
            <property name="deliveryStrategy" ref="TooBadDeliveryStrategyClass"/>
            <property name="bufferSecurity" ref="endPointSecurity"/>
       </bean>
```

The buffer related beans defined in the sourcePlugins.xml file are bufferManager, deliveryBuffer and ParcelBufferedEventListener. Buffer module, once set, requires no maintenance or modification.

Of interest to Dimsim developer is the bufferPlugin bean. Buffer plugin provides users with a set of methods for querying historical data.

2.1.3 Admin Database Monitor

This bean monitors changes to AdminDatabase_140.xml file and fires events appropriately. This bean enables multi-user plug-in processing. Since monash does not use a multi-user model, a dummy xml file with a single user named Monash is setup. The bean takes as input a set of listener beans who would like to be notified whenever user details change. It is the responsibility of listener beans to act upon change made to user details.

```
<bean  id="adminDBMonitor"
            class="au.edu.archer.dimsim.CrystalClear.AdminDBMonitor"
            scope="singleton">
            <constructor-arg type="java.lang.String" value="/mnt/ctrlpc/AdminDatabase_140.xml"/>
            <property name="listenerset">
                  <set>
                        <ref bean="RigakuSCPMultiUser"/>
                        <ref bean="RigakuJPGMultiUser"/>
                  </set>
            </property>
 </bean>
```

In the default example, RigakuSCPMultiUSer and RigakuJPGMultiUser beans sign up to received user update details.

2.1.4 Rigaku MultiUser Beans.

Multi-User beans manage creation and destruction of individual data capture beans. For each user returned by AdminDBMonitor bean, MultiUser beans create a new plugin.
Multi-User beans also manage user logins. User credentials are validated against data provided by the AdminDBMonitor bean. The property addValidUsersToEndPointSecurity enables dynamic addition of security tuple to the endpointSecurity bean
Of interest to system administrator is the 'dirMap' and 'replaceMap' property. These two fields play a key role in translating windows path to a linux path.
For example :

- 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
  - Property dirMap must be set to <entry key="D:" value="/mnt"/>
  - Property replaceMap must be set as

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

2.1.5 Ping Handler Bean.

This bean processes Ping requests from clients with session_ids. It validates the session_id and returns success or failure. Remote clients can PING producer server to verify if the session is current. There is no special configuration required for this bean.

2.1.6 Director Bean.

By default CIMA includes the basic director which is accessible from context. The DimsimDirector bean defined in the sourcePlugin.xml overrides the default CIMA director to include buffer, end point security and Ping requests to be processed.

2.2 Consumer Beans

2.2.1 Serializer bean

DimsimPersistenceMonitor manages file based persistent datastore to help client application restart after abnormal shutdown. The folder location for persistence store is defined by property *serializePath.* For this to work correctly, the user running the container should have write access to *serializePath.*

2.2.2 SRB Beans

SRBConfig bean defines the SRB data sink authentication details. To move to a different SRB instance, modify the values for this bean and regenerate the war file.

2.2.3 Cron Job Beans

Dimsim uses Spring Framework's TimerFactoryBean and ScheduledTimerTask to run tasks on a periodic interval. The purpose of running a cron job is to ensure that the subscription to Rigaku source plugin is current and valid.

Dimsim class AutoSubscribe takes as input a set of consumer and source plug-ins to monitor continuously. Update constructor argument values to change plug-ins or add additional plug-ins to monitor.

3.0 Developer Support

3.0 DEVELOPER SUPPORT

3.1 Add new CIMA plug-ins

3.1.1 Producer Plug-ins

By default, the recommended method for creating a new Produce plug-in is to subclass CIMA class org.instrumentmiddleware.cima.plugin.producer.impl.AbstractProducer This class requires the developer to pass in a parcel creator which packages data from the instrument for delivery to clients.

The frequency with which the parcel is created can be varied by subclasses. One such subclass is the org.instrumentmiddleware.cima.plugin.producer.poller.OnDataSinglePollerParcelProducer. This producer aims at producing the requested parcels when the data is available. In other words the unique poller thread has no data interval, hence it will poll repeatedly this producer. It is the responsibility of this producer's developer to implement a wait mechanism in the implementation of the produceParcels method. Dimsim plugins for Rigaku extend OnDataSinglePollerParcelProducer.

Other issues to bear in mind when developing new producer is

1. If you do not want the parcel produced by the producer to be buffered (in case you use the buffered director), then ensure that the parcel producer implements interface au.edu.archer.dimsim.buffer.plugin.IBufferNot

2. If you want to deploy multiple instances of the same plug-in for different users, then review the Multi-User plugin code and re-use the logic.

3.2.2 Consumer Plug-ins

By default consumer Plug-ins are sub classed from AbstractPlugin class. Dimsim provides the following subclasses for those interested in using them.

DimsimConsumerPlugin	The purpose of this class is to provide consumer plug-ins access to Remote Buffer
DimsimBufferedConsumerPlugin	The purpose of this class is to provide consumer plug-ins the ability to locally buffer incoming parcels, while they are waiting to be processed
DimsimBufferedPersistentConsumerPlugin	The purpose of this class is to persist parameters required for smooth restart of consumer plug-ins using serialization.

For usage, refer to source code and plugin SRBConsumer.