Manual for the uPart EQUIP System

isomura@teco.edu
Contents

- Overview of EQIUP
- Install Bundle
- Setup Environment for Compiling
- How to use EQUIP
- How to use uPartDBConnector
- Write a new Component
- References
Overview of EQIUP

- Container: ExporterGUI
- Local data-space: EquipDataSpace
- Client: GraphEditor, ComponentBrowser
Overview of EQIUP (Contd.)

- Container has class libraries and instances of components.
- Local data-space stores location of components. It relays request/response between a container and a client. It also relays setting/getting of property value between components or between a client and a component.
Overview of EQIUP (Contd.)

- Client can request a container to create/delete components. It can also set/get properties to/from the components.
- Component is a unit of a program running on a container.
- Component has multiple properties. Each property has a sort of data.
- Component can pass data to other components by linking their properties.
Install Bundle

Required software:
- Java2 SDK 1.4.X or higher
- EQUIP for Java (EQUIP4J) (Equator4uPart.zip)

Extract Equator4uPart.zip in an arbitrary folder referred to as <Installed folder>

Run by double-click
- >go to <Installed folder>/Equator4uPart/equip4j/infrastructure/install
- >runEquipDataSpace.bat
- >runExporterGUI.bat
- >runGraphEditor.bat
- >runConfigurationManager.bat
Setup Environment for Compiling from Scratch

Required software:
- Java2 SDK 1.4.X or higher
- Ant
- EQUIP for Java (EQUIP4J)

Environmental variables
- set JAVA_HOME=C:\Java\j2sdk1.4.X_0X
- set ANT_HOME=C:\Java\apache-ant-1.X.X
- set PATH=%JAVA_HOME%\bin;%ANT_HOME%\bin;%PATH%

TecO, University of Karlsruhe, Germany
Setup Environment (Contd.)

Compile
- `cd <Installed folder>/Equator4uPart/equip4j/infrastructure`
- `ant`

Run by double-click
- `go to <Installed folder>/Equator4uPart/equip4j/infrastructure/install`
- `runEquipDataSpace.bat`
- `runExporterGUI.bat`
- `runGraphEditor.bat`
- `runConfigurationManager.bat`
Equip Live Usage

USB Bridge

Network & Filter Representation of uPart

Parsing
How to use EQUIP

Create components.

1. On, CapabilityBrowser, select a component by right click.
2. The popup menu appears by left click.
3. Choose “Create request” to create the component.
How to use EQUIP (Contd.)

Check properties.

1. On, GraphComponentEditor, select a component in Component field by right click.
2. Drag&Drop it to Editor field.
How to use EQUIP (Contd.)

Set property value.

1. On, GraphComponentEditor, select a property of a component in Editor field by right click.
2. Press left click to show the popup menu.
3. Select “Set value”, then SetProperty window is appeared.
4. Input value on the window, then press “OK”.

Input value
How to use EQUIP (Contd.)

Make a link between the properties.

1. Create another component and drag&drop it to Editor field in GraphComponentEditor.
2. Select a property of a component and drag&drop it to a property of the other component.
How to use EQUIP (Contd.)

Example: Show sensing values of a uPart at online.

1. Create the following components.
   - ParticleFactory: It retrieves packets from uParts.
   - ParticleFilter: It filters uParts by ID.
   - ParticleDemultiplexer: It de-multiplexes the payload of the data packets into multiple data sets.
   - UPartAmpParser: It interprets a sort of data set human-readable sensing values.

2. Make the following link.
   - From “receivedclpcket” of ParticleFactory to “clpcket” of ParticleFilter.
How to use EQUIP (Contd.)

3. Set the following property value.
   - In ParticleFilter, set uPart ID that you have interested in to “senderid” property.
   - In ParticleFactory, set “true” to “startFlag” property. Then, it starts to receive data packets and creates ParticleInstance component for each uPart.

4. Make the following link.
   - From “aclpayload” of ParticleInstance to “aclpayload” of ParticleDemultiplexer.
   - From “acltuple1” of ParticleDemultiplexer to “acltuple” of UPartAmpParser.
Store Your Model

Store components
- Configuration manager (runconfigurationmanager.bat)
- Save as New... / Save as

Store the relationship
- In Graph Component Editor
- File -> Save as ...
Load Your Model

 Restore the components
  - Configuration Manager
  - Select your configuration
  - Press load button

 Restore the relationships
  - File -> Load ...
  - Set file type to „All files“ (your file has no suffix)
  - Select your file
  - Press Open
How to use uPartDBConnector

uPartDBConnector component can fetch the records from the database that stores incoming packets from uParts.

- Start "Apache" and "MySql" from XAMPP.
- Start EQUIP and create “uPartDBConnector”. 
How to use uPartDBConnector (Contd.)

After creating uPartDBConnector, DataExporterGUI will come up automatically.

You can specify “uPartID”, “start date”, “end date” as the query keys to the database.

Press “Fetch” button to fetch the records from the database.
How to use uPartDBConnector (Contd.)

After fetching the records, you can export them to a local file.

You can specify multiple data items to export.

You can also specify the delimiter and file name.

Press “Export” button to write down the records to the file.
Replay DB Data

- Fetch data in UPartDBConnector
- Set timescale property
  - +num slower replay
  - -num faster replay
- Set replay to true
Write a new Component

Create new directory
- `<Installed folder>/Equator4uPart/equip4j/infrastructure/src/equip/ect/components/<Name of component>`

Create `<Name of component>`.class and `<Name of component>BeanInfo.class` in the directory.

Modify build.xml on `<Installed folder>/Equator4uPart/equip4j/infrastructure`
- Modify target rule for “all”.
  `<target name="all" depends="ectCore,..., <Name of component>, install">`
Write a new Component (Contd.)

- Add target rule for the component.
  
  <!-- <Name of component> component target -->
  
  <target name="<Name of component> " depends="init">
    <mkdir dir="${build}/<Name of component> ">
    <!-- compile <Name of component> component -->
    <javac srcdir="${src}" destdir="${build}/<Name of component>" debug="on" deprecation="on" includeAntRuntime="no">
      <include name="equip/ect/components/<Name of component>/**/*.java"/>
    </javac>

    <!-- create jar -->
    <jar jarfile="${dist}/${components}/<Name of component>-component.jar" basedir="${build}/<Name of component>">
      <include name="equip/ect/components/<Name of component>/**/*.class"/>
    </jar>
  </target>
  
  <!-- <Name of component> component target -->

  <!-- compile <Name of component> component -->
  
  <javac srcdir="${src}" destdir="${build}/<Name of component>" debug="on" deprecation="on" includeAntRuntime="no">
    <include name="equip/ect/components/<Name of component>/**/*.java"/>
  </javac>

  <!-- create jar -->
  
  <jar jarfile="${dist}/${components}/<Name of component>-component.jar" basedir="${build}/<Name of component>">
    <include name="equip/ect/components/<Name of component>/**/*.class"/>
  </jar>
  
  <!-- <Name of component> component target -->
Write a new Component (Contd.)

Component consists of two kind of classes at least.

- <Name of component>.class: Component itself.
- <Name of component>BeanInfo.class: It defines properties that are visible on GUI.
- Template of these classes are provided.
  - <Installed folder>/Equator4uPart/equip4j/infrastructure/src/equip/ect/components/template/

EQUIP works as event driven fashion, thus you have to put your own processing code in constructor or getter, setter methods.
Write a new Component (Contd.)

<Name of component>.class
- It should implement “Serializable” interface.
  - public class Template implements Serializable
- It should have no argument constructor.
  - public Template()
- Stop method is recommended for finalize processes.
  - public synchronized void stop()
- Each property can be accessed by getter/setter method, thus the valuable for the property is recommended to be “protected” or “private”.
  - protected String input;
Write a new Component (Contd.)

Type of the property can be any kind of class, however String, int and boolean are enough for simple component.

To make a link between properties, the type of them should be same or compatible.

Getter and setter are recommended to be “synchronized”

- public synchronized void setInput(String newVal)
- public String void getInput()
Write a new Component (Contd.)

- `firePropertyChange()` is required in setter to reflect new value on GUI. The first argument is a property name. It should be matched with BeanInfo class.
  - `propertyChangeListeners.firePropertyChange("input", oldVal, newVal);`
- Setter should be protected, if the property is read only.
  - `protected synchronized void setOutput(String newVal)`
- The followings are magic words for EQUIP.
  - `propertyChangeListeners`
  - `removePropertyChangeListener()`
  - `addPropertyChangeListener()`
Write a new Component (Contd.)

<Name of component>BeanInfo.class

The following statement binds the BeanInfo and the component.

- protected Class beanClass = Template.class;

getPropertyDescriptors() defines properties that are visible on GUI.

Each PropertyDescriptor defines the property name and the corresponding getter and setter.

- PropertyDescriptor("input", beanClass, "getInput", "setInput");
Write a new Component (Contd.)

- It becomes read-only property on GUI, when you assign “null” value to setter.
  - `PropertyDescriptor("output", beanClass, "getOutput", null);`

- Please check the property name and getter and setter in PropertyDescriptor carefully. Compiler does not make any error/warning even if they are mismatched with the component class!
References


2. Chris's Equator Pages,
http://www.crg.cs.nott.ac.uk/~cmg/Equator/