

# Adaptive Home Automation Source Documentation

Ueli Rutishauser and Alain Schaefer

July 11'th 2002

# Contents

<b>1</b>	<b>Package <code>aha.framework</code></b>	<b>3</b>
1.1	Classes . . . . .	4
1.1.1	CLASS <b>Customizer</b> . . . . .	4
1.1.2	CLASS <b>GeneralUtils</b> . . . . .	5
1.1.3	CLASS <b>Pipeline</b> . . . . .	6
1.1.4	CLASS <b>PipelineData</b> . . . . .	7
1.1.5	CLASS <b>PipelineHandler</b> . . . . .	7
1.1.6	CLASS <b>RingBuffer</b> . . . . .	8
1.1.7	CLASS <b>RingBufferEnumerator</b> . . . . .	11
1.1.8	CLASS <b>RingBufferIterator</b> . . . . .	12
<b>2</b>	<b>Package <code>aha.bus</code></b>	<b>14</b>
2.1	Interfaces . . . . .	15
2.1.1	INTERFACE <b>BusController</b> . . . . .	15
2.2	Classes . . . . .	16
2.2.1	CLASS <b>PropertyDescription</b> . . . . .	16
<b>3</b>	<b>Package <code>aha.history</code></b>	<b>17</b>
3.1	Interfaces . . . . .	18
3.1.1	INTERFACE <b>HistoryRemote</b> . . . . .	18
3.2	Classes . . . . .	18
3.2.1	CLASS <b>HistoryAgent</b> . . . . .	18
3.2.2	CLASS <b>HistoryServer</b> . . . . .	20
3.2.3	CLASS <b>LogArgs</b> . . . . .	21
<b>4</b>	<b>Package <code>aha.control</code></b>	<b>24</b>
4.1	Interfaces . . . . .	26
4.1.1	INTERFACE <b>ControlRemote</b> . . . . .	26
4.2	Classes . . . . .	26
4.2.1	CLASS <b>ExecutionPipelineData</b> . . . . .	26
4.2.2	CLASS <b>PipelineHandlerAble</b> . . . . .	27
4.2.3	CLASS <b>PipelineHandlerHistory</b> . . . . .	27
4.2.4	CLASS <b>PipelineHandlerOutput</b> . . . . .	28
4.2.5	CLASS <b>PipelineHandlerProcessing</b> . . . . .	28
4.2.6	CLASS <b>Presence</b> . . . . .	29
4.2.7	CLASS <b>Room</b> . . . . .	29
4.2.8	CLASS <b>RoomControlAgent</b> . . . . .	33
4.2.9	CLASS <b>TimeSeriesPlotter</b> . . . . .	34
4.2.10	CLASS <b>TimeSeriesPlotterExtended</b> . . . . .	35

<b>5</b>	<b>Package aha.middleware</b>	<b>38</b>
5.1	Classes . . . . .	39
5.1.1	CLASS <b>Decision</b> . . . . .	39
5.1.2	CLASS <b>Message</b> . . . . .	41
<b>6</b>	<b>Package aha</b>	<b>45</b>
6.1	Interfaces . . . . .	46
6.1.1	INTERFACE <b>BusRemote</b> . . . . .	46
6.2	Classes . . . . .	48
6.2.1	CLASS <b>AhaServer</b> . . . . .	48
6.2.2	CLASS <b>BossAgent</b> . . . . .	48
6.2.3	CLASS <b>BusAgent</b> . . . . .	49
6.2.4	CLASS <b>BusServer</b> . . . . .	51
6.2.5	CLASS <b>Config</b> . . . . .	52
6.2.6	CLASS <b>ControlServer</b> . . . . .	52
6.2.7	CLASS <b>DisplayServer</b> . . . . .	53
6.2.8	CLASS <b>RoomDisplay</b> . . . . .	54
6.2.9	CLASS <b>RoomDisplayAgent</b> . . . . .	58
<b>7</b>	<b>Package aha.bus.lonworks</b>	<b>62</b>
7.1	Classes . . . . .	63
7.1.1	CLASS <b>LNSController</b> . . . . .	63

# Chapter 1

## Package aha.framework

<i>Package Contents</i>	<i>Page</i>
<hr/>	
<b>Classes</b>	
<b>Customizer</b> .....	4
<i>Generic Customizer for AdaAgent's.</i>	
<b>GeneralUtils</b> .....	5
<i>general utilities, converting of numbers.</i>	
<b>Pipeline</b> .....	6
<i>Pipeline.</i>	
<b>PipelineData</b> .....	7
<i>generic parent-class for data that can be sent through a processing pipeline.</i>	
<b>PipelineHandler</b> .....	7
<i>abstract base class for pipeline system (chain of responsibility).</i>	
<b>RingBuffer</b> .....	8
<i>A RingBuffer can be used to store a limited number of entries of any type within a buffer.</i>	
<b>RingBufferEnumerator</b> .....	11
<i>Enumerator for the RingBuffer.</i>	
<b>RingBufferIterator</b> .....	12
<i>Iterator for the RingBuffer.</i>	

---

## 1.1 Classes

### 1.1.1 CLASS Customizer

---

Generic Customizer for AdaAgent's.

This class allows it to generate generic Customizers for every AdaAgent. Variables that should be modifiable/viewable inside of the Customizer are discovered automatically.

The AdaAgent has to provide a getMethod for every variable that it wants to be displayed inside a customizer. The name of the get Method must be according to this pattern:  
get[A-Z][a-zA-Z0-9]\*\_Customizer

The AdaAgent must furthermore provide a setMethod for every variable that should be modifiable. Not every variable that has a getMethod must have a setMethod. Variables that only have a getMethod are read only. The name of the set Method must be according to this pattern: set[A-Z][a-zA-Z0-9]\*\_Customizer

The following types of variables are currently supported: String, Boolean, Double, Float, int, float, double, boolean Other types can easily be added (see method setNewValue in the class CustomizerVariable)

This class uses java regular expression support which means that it requires a jdk1.4 or better

see ada/tests/AdaAgent1.java on how to use this class.

#### DECLARATION

---

```
public class Customizer
extends java.lang.Object
```

#### CONSTRUCTORS

---

- *Customizer*  
public **Customizer**( java.lang.Object agent )
  - **Usage**
    - \* creates a new customizer; content of the customizer is discovered automatically through reflection, see comment for this class
  - **Parameters**
    - \* agent -

#### METHODS

---

- *close*  
public void **close**( )
  - **Usage**

- 
- *getAgent*  
public Object **getAgent**( )  
– **Returns** -
  - *refresh*  
public void **refresh**( )  
– **Usage**  
\* get new values for all variables (tell frame to do so)

### 1.1.2 CLASS GeneralUtils

---

general utilities, converting of numbers.

#### DECLARATION

---

```
public class GeneralUtils
extends java.lang.Object
```

#### CONSTRUCTORS

---

- *GeneralUtils*  
public **GeneralUtils**( )  
– **Usage**  
\* Creates a new instance of Utils

#### METHODS

---

- *convertFahrenheitToCelsius*  
public static double **convertFahrenheitToCelsius**( double **fahrenheit** )  
– **Usage**  
\* converts fahrenheit degrees to celsius degrees  
– **Parameters**  
\* **fahrenheit** -  
– **Returns** - celsius degrees (double)

### 1.1.3 CLASS Pipeline

---

Pipeline.

Implements an adapt version of the chain of responsibility pattern (processing doesn't stop if one handler processes data, it is nevertheless carried on to the next one in the chain). This enables serial pipeline-based-processing (stream).

be carefull while using `addHandler(...)` , it is important in which order you add your handlers. they will be processed in the same order then `addHandler(...)` is called during initialization

see "design patterns" (chain of responsibility) by gamma et all and doc of this project for more explanations

#### DECLARATION

---

```
public class Pipeline
extends java.lang.Object
```

#### CONSTRUCTORS

---

- *Pipeline*  

```
public Pipeline( )
```

  - **Usage**  
 \* Creates a new instance of Pipeline

#### METHODS

---

- *addHandler*  

```
public void addHandler( int pos, aha.framework.PipelineHandler handler )
```
- *addHandler*  

```
public void addHandler( aha.framework.PipelineHandler handler )
```
- *empty*  

```
public void empty( )
```
- *feed*  

```
public void feed( aha.framework.PipelineData data )
```
- *removeHandler*  

```
public void removeHandler( int pos )
```
- *size*  

```
public int size( )
```

### 1.1.4 CLASS PipelineData

---

generic parent-class for data that can be sent through a processing pipeline. See subclasses for more explanations.

#### DECLARATION

---

```
public class PipelineData
extends java.lang.Object
```

#### FIELDS

---

- public static int UNDEFINED

–

#### CONSTRUCTORS

---

- *PipelineData*  
public **PipelineData**( int pipelineDataType )

– **Usage**

\* Creates a new instance of PipelineData

### 1.1.5 CLASS PipelineHandler

---

abstract base class for pipeline system (chain of responsibility).

#### DECLARATION

---

```
public abstract class PipelineHandler
extends java.lang.Object
```

#### CONSTRUCTORS

---

- *PipelineHandler*  
public **PipelineHandler**( )

#### METHODS

---

- *feed*  
public abstract PipelineData feed( aha.framework.PipelineData data )

### 1.1.6 CLASS RingBuffer

---

A RingBuffer can be used to store a limited number of entries of any type within a buffer.

As soon as the maximum number of entries is reached, the next entry is added to the end of the list and the first entry is removed from it. In this case, all elements are stored in an Object[]. There are variants for primitive types as well.

#### DECLARATION

---

```
public class RingBuffer
extends java.lang.Object
implements java.lang.Cloneable, java.util.Collection, java.io.Serializable
```

#### SERIALIZABLE FIELDS

---

- private Object theBuffer  
–
- private int size  
–
- private int position  
–

#### CONSTRUCTORS

---

- *RingBuffer*  
public RingBuffer( )
- *RingBuffer*  
public RingBuffer( int aSize )

#### METHODS

---

- *add*  
public boolean add( java.lang.Object anObject )  
– **Usage**  
\* Adds an element to the ring buffer, potentially removing the first element to make more room.

– **Parameters**

\* anObject - java.lang.Object

---

• *addAll*

public boolean **addAll**( java.util.Collection coll )

• *clear*

public void **clear**( )

• *contains*

public boolean **contains**( java.lang.Object obj )

• *containsAll*

public boolean **containsAll**( java.util.Collection coll )

• *elementAt*

public Object **elementAt**( int aPosition )

– **Usage**

\* Returns the element at the specified position. Positioning starts at 0.

– **Parameters**

\* aPosition - int

– **Returns** - java.lang.Object

---

• *elements*

public Enumeration **elements**( )

– **Usage**

\* Returns an enumeration of all elements within this ring buffer.

This is not thread safe, like most enumerations.

– **Returns** - java.util.Enumeration

---

• *getMaximumSize*

public int **getMaximumSize**( )

– **Usage**

\* Returns the maximum number of elements in the ring buffer.

– **Returns** - int

---

• *init*

public void **init**( )

– **Usage**

\* Initializes some fields.

- *isEmpty*

public boolean isEmpty( )

- *iterator*

public Iterator iterator( )

- *remove*

public boolean remove( java.lang.Object obj )

- *removeAll*

public boolean removeAll( java.util.Collection c )

– Usage

- \* Removes all elements from the buffer my overwriting them with null and sets the position field to 0.

- 
- *removeElement*

public void removeElement( int aPosition )

– Usage

- \* Removes the element at aPosition from the RingBuffer.

– Parameters

- \* aPosition - int

- 
- *retainAll*

public boolean retainAll( java.util.Collection c )

- *setMaximumSize*

public void setMaximumSize( int newSize )

– Usage

- \* Sets the maximum size of elements in the RingBuffer. All Elements currently in the buffer will be copied to the new buffer, if there is enough space.

Otherwise, only the newest elements will be copied.

This method contains a **synchronized** block, which will be used if theBuffer has already been instantiated.

– Parameters

- \* newSize - int

- 
- *size*

public int size( )

\* Returns the current number of elements in the ring buffer.

– **Returns** - int

• *toArray*

**public Object toArray( )**

• *toArray*

**public Object toArray( java.lang.Object [] a )**

• *toString*

**public String toString( )**

– **Usage**

\* Returns a string representation of the RingBuffer and it's contents.

– **Returns** - java.lang.String

### 1.1.7 CLASS RingBufferEnumerator

Enumerator for the RingBuffer.

DECLARATION

```
public class RingBufferEnumerator
extends java.lang.Object
implements java.util.Enumeration
```

CONSTRUCTORS

• *RingBufferEnumerator*

**public RingBufferEnumerator( )**

– **Usage**

\* RingBufferEnumerator constructor comment.

• *RingBufferEnumerator*

**public RingBufferEnumerator( aha.framework.RingBuffer aRingBuffer )**

– **Usage**

\* RingBufferEnumerator constructor comment.

## METHODS

- *hasMoreElements*  
 public boolean **hasMoreElements**( )  
 – **Usage**  
 \* hasMoreElements method comment.

---

- *nextElement*  
 public Object **nextElement**( )  
 – **Usage**  
 \* nextElement method comment.

## 1.1.8 CLASS RingBufferIterator

Iterator for the RingBuffer.

## DECLARATION

```
public class RingBufferIterator
extends java.lang.Object
implements java.util.Iterator
```

## CONSTRUCTORS

- *RingBufferIterator*  
 public **RingBufferIterator**( )  
 – **Usage**  
 \* RingBufferEnumerator constructor comment.

---

- *RingBufferIterator*  
 public **RingBufferIterator**( aha.framework.RingBuffer aRingBuffer )  
 – **Usage**  
 \* RingBufferEnumerator constructor comment.

## METHODS

- *hasNext*  
 public boolean **hasNext**( )  
 – **Usage**  
 \* hasMoreElements method comment.

- *next*  
public Object **next**( )  
    – **Usage**  
    \* nextElement method comment.  

---
- *remove*  
public void **remove**( )

## Chapter 2

# Package aha.bus

*Package Contents* *Page*

---

### Interfaces

**BusController** ..... 15  
*Interface to abstract the access to different implementations of building buses.*

### Classes

**PropertyDescription** ..... 16  
*\$Id: PropertyDescription.java,v 1.2 2002/04/16 10:19:09 urut Exp \$ \$Name:*  
*\$*

*\$Log: PropertyDescription.java,v \$*  
*Revision 1.2 2002/04/16 10:19:09 urut*  
*javadoc added*

---

## 2.1 Interfaces

### 2.1.1 INTERFACE BusController

---

Interface to abstract the access to different implementations of building buses.

It makes the assumption that a bus allows the access to a bunch of variables which can be queried and set

#### DECLARATION

---

```
public interface BusController
```

---

#### METHODS

---

- *getDescription*  
`public PropertyDescription getDescription( java.lang.String name )`
  - **Usage**
    - \* what the does this method do ???

---
- *getProperties*  
`public Collection getProperties( java.lang.String regex )`
  - **Usage**
    - \* returns the names of all bus variables matching a certain regular expression
  - **Parameters**
    - \* `regex` - Regular expression for the variable names
  - **Returns** - Collection of variable names

---
- *getProperty*  
`public String getProperty( java.lang.String name )`
  - **Usage**
    - \* gets the value of a bus variable
  - **Parameters**
    - \* `name` - the bus variables name

---
- *registerPropertyChangeListener*  
`public void registerPropertyChangeListener( java.lang.String name, java.beans.PropertyChangeListener listener )`
  - **Usage**
    - \* registers a listener for a bus variable
  - **Parameters**
    - \* `name` - the bus variable name
    - \* `listener` - for variable changes

- *removePropertyChangeListener*

```
public void removePropertyChangeListener( java.lang.String name,
java.beans.PropertyChangeListener listener )
```

– **Usage**

\* removes a listener from the list of listeners for a bus variable

– **Parameters**

\* **name** - the bus variable name

\* **listener** - the listener which should be removed

- *setProperty*

```
public void setProperty( java.lang.String name, java.lang.String value )
```

– **Usage**

\* sets the value for a bus variable

– **Parameters**

\* **name** - the bus variables name

\* **value** - the value to be set

## 2.2 Classes

### 2.2.1 CLASS PropertyDescription

\$Id: PropertyDescription.java,v 1.2 2002/04/16 10:19:09 urut Exp \$ \$Name: \$

\$Log: PropertyDescription.java,v \$  
Revision 1.2 2002/04/16 10:19:09 urut  
javadoc added

#### DECLARATION

```
public class PropertyDescription
extends java.lang.Object
```

#### CONSTRUCTORS

- *PropertyDescription*

```
public PropertyDescription( )
```

# Chapter 3

## Package aha.history

<i>Package Contents</i>	<i>Page</i>
<hr/>	
<b>Interfaces</b>	
<b>HistoryRemote</b> ..... 18	
<i>add the ability to directly send AbleEvents to the HistoryAgent without using the Able event system.</i>	
<b>Classes</b>	
<b>HistoryAgent</b> ..... 18	
<i>Responsible for the logging of "Events" in the BusSystem and decisions taken by controll agents to a database.</i>	
<b>HistoryServer</b> ..... 20	
<i>starts and initializes the HistoryAgent</i>	
<b>LogArgs</b> ..... 21	
<i>a value object for log records.</i>	

---

## 3.1 Interfaces

### 3.1.1 INTERFACE HistoryRemote

---

add the ability to directly send AbleEvents to the HistoryAgent without using the Able event system.

#### DECLARATION

---

```
public interface HistoryRemote
implements java.rmi.Remote
```

#### METHODS

---

- *logAbleEvent*

```
public void logAbleEvent( AbleEvent  theEvent )
```

  - **Usage**
    - \* inserts the event into the database
  - **Parameters**
    - \* **theEvent** - to be inserted AbleEvent
  - **See Also**
    - \* `com.ibm.able.AbleEvent`

## 3.2 Classes

### 3.2.1 CLASS HistoryAgent

---

Responsible for the logging of "Events" in the BusSystem and decisions taken by controll agents to a database. It reads the dbDriver and the jdbc-Connection String for the db from the config file.

It uses Ables TimerEvent Systems to periodically log the pending messages to the database. The size of the period is read from the config file.

Variable updates in the Bussystem are only logged if at least one other agent has registered as a listener for that variable. An exception are the standard variables. These are defined in the Config file and logged every time the agent wakes up for logging.

Other agents can log their decisions by sending a Decision ( in 5.1.1, page 39) .

#### DECLARATION

---

```
public class HistoryAgent
extends AbleDefaultAgent
implements HistoryRemote
```

## CONSTRUCTORS

- *HistoryAgent*  
public **HistoryAgent**( )

## METHODS

- *getId*  
public Integer **getId**( java.lang.String **varName** )
  - **Usage**
    - \* Returns the Database Id for a given variableName. Tries to optimize Executiontime for this, by caching the Id, if the variable is not found in the database a new entry is created.
  - **Parameters**
    - \* **varName** - The variable name

---

- *handleAbleEvent*  
public void **handleAbleEvent**( AbleEvent **theAbleEvent** )
  - **Usage**
    - \* processes able events sent to this agent.

this agent is able to handle decision and log events. Called from able framework
  - **Parameters**
    - \* **theAbleEvent** - to be processed AbleEvent
  - **See Also**
    - \* **aha.middleware.Message** ( in 5.1.2, page 41)
    - \* **aha.middleware.Decision** ( in 5.1.1, page 39)
    - \* **aha.history.LogArgs** ( in 3.2.3, page 21)

---

- *init*  
public void **init**( java.lang.String **busAgentServer** )
  - **Usage**
    - \* Initializes the Agent, connects to BusAgent and JDBC-Database, binds itself into the RMIRegistry
  - **Parameters**
    - \* **busAgentServer** - Servername where the BusAgent is running

---

- *insert*  
public void **insert**( aha.history.LogArgs **args** )
  - **Usage**
    - \* stores a LogArgs object directly into the database
  - **Parameters**
    - \* **args** - The LogArgs to be stored

---

\* aha.history.LogArgs ( in 3.2.3, page 21)

---

- *insert*

```
public void insert( aha.history.LogArgs args, java.util.Date norm )
```

- **Usage**

- \* stores a LogArgs object directly into the database.

- **Parameters**

- \* **args** - The LogArgs to be stored

- \* **norm** - The Date which should be used for the normalization

---

- *logAbleEvent*

```
public void logAbleEvent( AbleEvent theAbleEvent )
```

- **Usage**

- \* is the same as if sending an AbleEvent over the Able event system

---

- *processTimerEvent*

```
public void processTimerEvent( )
```

- **Usage**

- \* called from able whenever the timer event occurs. Inserts all pending log records into the database

---

- *reset*

```
public void reset( )
```

- **Usage**

- \* Reset this Agent to its initialized state

### 3.2.2 CLASS HistoryServer

---

starts and initializes the HistoryAgent

#### DECLARATION

---

```
public class HistoryServer
extends java.lang.Object
```

#### CONSTRUCTORS

---

- *HistoryServer*

```
public HistoryServer( )
```

METHODS

---

- *init*  
public void **init**( java.lang.String busAgent )  
– **Usage**  
\* starts and initializes the HistoryAgent

---

- *main*  
public static void **main**( java.lang.String [] args )  
– **Usage**  
\* parses cmd arguments and calls init

### 3.2.3 CLASS LogArgs

---

a value object for log records. Mainly used to pass around log records to the HistoryAgent

DECLARATION

---

```
public class LogArgs
extends java.lang.Object
implements java.io.Serializable
```

SERIALIZABLE FIELDS

---

- private Date timeStamp  
–
- private String varName  
–
- private String value  
–
- private boolean normalize  
–

CONSTRUCTORS

---

- *LogArgs*  
public **LogArgs**( java.util.Date timeStamp, java.lang.String varName,  
java.lang.String value, boolean normalize )

METHODS

---

- *getNormalize*  
public boolean **getNormalize**( )
  - **Usage**
    - \* Gets the normalize.
  - **Returns** - Returns a boolean

---
- *getTimeStamp*  
public Date **getTimeStamp**( )
  - **Usage**
    - \* Gets the timeStamp.
  - **Returns** - Returns a Date

---
- *getValue*  
public String **getValue**( )
  - **Usage**
    - \* Gets the value.
  - **Returns** - Returns a String

---
- *getVarName*  
public String **getVarName**( )
  - **Usage**
    - \* Gets the varName.
  - **Returns** - Returns a String

---
- *setNormalize*  
public void **setNormalize**( boolean **normalize** )
  - **Usage**
    - \* Sets the normalize value. True if this log record should be normalized
  - **Parameters**
    - \* **normalize** - The normalize to set

---
- *setTimeStamp*  
public void **setTimeStamp**( java.util.Date **timeStamp** )
  - **Usage**
    - \* Sets the timeStamp.
  - **Parameters**
    - \* **timeStamp** - The timeStamp to set

---
- *setValue*  
public void **setValue**( java.lang.String **value** )
  - **Usage**

– **Parameters**

\* value - The value to set

---

• *setVarName*

```
public void setVarName( java.lang.String varName )
```

– **Usage**

\* Sets the varName.

– **Parameters**

\* varName - The varName to set

---

• *toString*

```
public String toString( )
```

– **Usage**

\* converts the logrecord into a string

# Chapter 4

## Package aha.control

<i>Package Contents</i>	<i>Page</i>
<hr/>	
<b>Interfaces</b>	
<b>ControlRemote</b> .....	26
<i>remote interface for rmi calls to ControlAgent.</i>	
<b>Classes</b>	
<b>ExecutionPipelineData</b> .....	26
<i>data to be transfered through pipeline to initiate processing of decisions.</i>	
<b>PipelineHandlerAble</b> .....	27
<i>this pipeline handler acts as a gateway between ada and the internal processing structure data that is available in the pipeline is converted and put into the able message delivery system for processing by other agents.</i>	
<b>PipelineHandlerHistory</b> .....	27
<i>Pipeline handler that acts as supplier for the history agent</i>	
<b>PipelineHandlerOutput</b> .....	28
<i>Pipeline handler for debug output</i>	
<b>PipelineHandlerProcessing</b> .....	28
<i>Pipeline handler that executes decisions taken by the inference engine (feeder of the pipeline).</i>	
<b>Presence</b> .....	29
<i>presence detector handling normalizes HVAC (movement) and daylight values</i>	
 <i>filters HVAC pulses so that presence yes/no decisions are only taken with a low frequency</i>	
 <i>runs as an independent thread that is being notified of changes by the controlAgent that is responsilbe for this room</i>	
 <i>one instance of this class handles all the presence detectors for one room</i>	
<b>Room</b> .....	29
<i>general interface to a room.</i>	
<b>RoomControlAgent</b> .....	33
<i>Agent that is responsible to control one specific room.</i>	
<b>TimeSeriesPlotter</b> .....	34
<i>plottina time series (with one data set)</i>	

*plotting time series (with multiple data sets).*

---

## 4.1 Interfaces

### 4.1.1 INTERFACE ControlRemote

---

remote interface for rmi calls to ControlAgent.

This interface doesn't contain any methods so far but it is required to identify classes who have certain capabilities.

DECLARATION

---

```
public interface ControlRemote
implements java.rmi.Remote
```

## 4.2 Classes

### 4.2.1 CLASS ExecutionPipelineData

---

data to be transfered through pipeline to initiate processing of decisions.

DECLARATION

---

```
public class ExecutionPipelineData
extends aha.framework.PipelineData
```

CONSTRUCTORS

---

- *ExecutionPipelineData*  

```
public ExecutionPipelineData( aha.middleware.Decision decision,
AbleDefaultAgent sender )
```

  - **Usage**
    - \* Creates a new instance of ExecutionPipelineData

METHODS

---

- *getDecision*  

```
public Decision getDecision( )
```

    - **Usage**
      - \* return content of this data holder
    - **Returns** - Decision
- 

- *getSender*

- **Usage**
  - \* to find out which agent put this decision into the pipeline.
- **Returns** - agent

#### 4.2.2 CLASS PipelineHandlerAble

---

this pipeline handler acts as a gateway between ada and the internal processing structure data that is available in the pipeline is converted and put into the able message delivery system for processing by other agents. the event delivery queues of the all the subscribers to this messages act as continuation of the local pipeline on a remote location.

##### DECLARATION

---

```
public class PipelineHandlerAble
extends aha.framework.PipelineHandler
```

##### CONSTRUCTORS

---

- *PipelineHandlerAble*  
public **PipelineHandlerAble**( )

##### METHODS

---

- *feed*  
public PipelineData **feed**( aha.framework.PipelineData data )

#### 4.2.3 CLASS PipelineHandlerHistory

---

Pipeline handler that acts as supplier for the history agent

##### DECLARATION

---

```
public class PipelineHandlerHistory
extends aha.framework.PipelineHandler
```

##### CONSTRUCTORS

---

- *PipelineHandlerHistory*  
public **PipelineHandlerHistory**( aha.history.HistoryRemote historyRemote )

## METHODS

- *feed*  
`public PipelineData feed( aha.framework.PipelineData data )`

#### 4.2.4 CLASS PipelineHandlerOutput

Pipeline handler for debug output

## DECLARATION

```
public class PipelineHandlerOutput
extends aha.framework.PipelineHandler
```

## CONSTRUCTORS

- *PipelineHandlerOutput*  
`public PipelineHandlerOutput( )`

## METHODS

- *feed*  
`public PipelineData feed( aha.framework.PipelineData data )`

#### 4.2.5 CLASS PipelineHandlerProcessing

Pipeline handler that executes decisions taken by the inference engine (feeder of the pipeline). takes care of oscillation avoidance, timeouts, local history, limits

## DECLARATION

```
public class PipelineHandlerProcessing
extends aha.framework.PipelineHandler
```

## CONSTRUCTORS

- *PipelineHandlerProcessing*  
`public PipelineHandlerProcessing( aha.control.Room theRoom )`

## METHODS

- *feed*

### 4.2.6 CLASS Presence

---

presence detector handling normalizes HVAC (movement) and daylight values

filters HVAC pulses so that presence yes/no decisions are only taken with a low frequency

runs as an independent thread that is being notified of changes by the controlAgent that is responsible for this room

one instance of this class handles all the presence detectors for one room

#### DECLARATION

---

```
public class Presence
extends java.lang.Object
implements java.lang.Runnable
```

#### CONSTRUCTORS

---

- *Presence*  

```
public Presence( aha.control.Room theRoom, aha.control.RoomControlAgent
controlAgent )
```

  - **Parameters**
    - \* `theRoom` -
    - \* `controlAgent` - callback for notifying

#### METHODS

---

- *isPresent*  

```
public synchronized boolean isPresent( )
```

  - **Usage**
    - \* true = someone is present false = no one is present
  - **Returns** -

---
- *run*  

```
public void run( )
```

---
- *setGraphActive*  

```
public void setGraphActive( boolean graphActive )
```

### 4.2.7 CLASS Room

---

general interface to a room. encapsulates all room-specific processing. Each room is related to an arbitrary number of network variables that represent blind controllers, lights, switches and presence detectors

DECLARATION

---

```
public class Room
extends AbleObject
```

FIELDS

---

- public static int LIGHT1  
–
- public static int LIGHT2  
–
- public static int UNDEFINED  
–
- public static int ON  
–
- public static int OFF  
–
- public static int BLIND\_UP  
–
- public static int BLIND\_DOWN  
–
- public static int BLIND\_STOP  
–
- public static int BLIND\_CHANGE  
–
- public static int LIGHT\_ON  
–
- public static int LIGHT\_OFF  
–

CONSTRUCTORS

---

- *Room*  
public **Room**( aha.BusRemote busRemote )

## METHODS

- *action*  
 public void **action**( int **what**, java.lang.String **name** )  
 – **Usage**  
 \* for compatibility reasons, calls action method without parameters  


---
- *action*  
 public void **action**( int **what**, java.lang.String **name**, java.lang.String **parameter1**, java.lang.String **parameter2** )  
 – **Usage**  
 \* methods that changes the value of a variable only possible if this variable is not r/o  
 what : LIGHT1, LIGHT2, etc (static definitions)  


---
- *allBlindsDown*  
 public void **allBlindsDown**( )  


---
- *allBlindsUp*  
 public void **allBlindsUp**( )  


---
- *allLightOff*  
 public void **allLightOff**( )  


---
- *allLightOn*  
 public void **allLightOn**( )  


---
- *changeRoom*  
 public void **changeRoom**( java.lang.String **newRoom** )  


---
- *getBlinds*  
 public Set **getBlinds**( )  
 – **Returns** -  


---
- *getHumidity*  
 public String **getHumidity**( )  
 – **Returns** -  


---
- *getIlluminance1*  
 public String **getIlluminance1**( )  
 – **Returns** -  


---
- *getIlluminance2*  
 public String **getIlluminance2**( )  
 – **Returns** -  


---
- *getLights*

– **Returns** -

---

- *getPresenceDetectorsLight*

public Set **getPresenceDetectorsLight**( )

– **Usage**

\* get all variable names of presence detector (light) network variables

– **Returns** -

---

- *getPresenceDetectorsMovement*

public Set **getPresenceDetectorsMovement**( )

– **Usage**

\* get all variable names of presence detector (movement) network variables

– **Returns** -

---

- *getPresenceLightValues*

public Collection **getPresenceLightValues**( )

– **Usage**

\* get all values of all available presence detector light network variables

– **Returns** -

---

- *getPresenceMovementValues*

public Collection **getPresenceMovementValues**( )

– **Usage**

\* get all values of all available presence detector movement network variables

– **Returns** -

---

- *getSunEast*

public String **getSunEast**( )

– **Returns** -

---

- *getSunSouth*

public String **getSunSouth**( )

– **Returns** -

---

- *getSunWest*

public String **getSunWest**( )

– **Returns** -

---

- *getTemp*

public String **getTemp**( )

– **Returns** -

---

- *getWeatherVariables*

public List **getWeatherVariables**( )

- *handleAbleEvent*  
public void handleAbleEvent( AbleEvent theAbleEvent )
- *init*  
public void init( )
- *reset*  
public void reset( )
- *updateVariableValues*  
public void updateVariableValues( )

– Usage

\* get all data in case not everything is available (no updates received so far)

#### 4.2.8 CLASS RoomControlAgent

---

Agent that is responsible to control one specific room. One instance of this agent just controls one room. This agent processes all inputs to one room and takes decisions.

##### DECLARATION

---

```
public class RoomControlAgent
extends AbleDefaultAgent
implements ControlRemote
```

##### CONSTRUCTORS

---

- *RoomControlAgent*  
public **RoomControlAgent**( java.lang.String roomName )

##### METHODS

---

- *getBlindAction\_Customizer*  
public String getBlindAction\_Customizer( )
- *getHumidity\_Customizer*  
public String getHumidity\_Customizer( )
- *getIlluminance1\_Customizer*  
public String getIlluminance1\_Customizer( )
- *getLightAction\_Customizer*  
public String getLightAction\_Customizer( )
- *getPresenceLightState\_Customizer*  
public String getPresenceLightState\_Customizer( )

- *getPresenceMovementState\_Customizer*  
public boolean **getPresenceMovementState\_Customizer**( )
- *getRadiationState\_Customizer*  
public String **getRadiationState\_Customizer**( )
- *getSunEast\_Customizer*  
public String **getSunEast\_Customizer**( )
- *getSunSouth\_Customizer*  
public String **getSunSouth\_Customizer**( )
- *getSunWest\_Customizer*  
public String **getSunWest\_Customizer**( )
- *getTemperature\_Customizer*  
public String **getTemperature\_Customizer**( )
- *handleAbleEvent*  
public void **handleAbleEvent**( AbleEvent theAbleEvent )

– Usage

\* events received from able are handed over to this method by able

- 
- *init*  
public void **init**( java.lang.String busAgentServer, java.lang.String historyAgent )
  - *processTimerEvent*  
public void **processTimerEvent**( )
  - *reset*  
public void **reset**( )
  - *setActive*  
public void **setActive**( boolean a )
  - *start*  
public void **start**( )

#### 4.2.9 CLASS TimeSeriesPlotter

---

plotting time series (with one data set). The TimePeriod is 1 Millisecond (maximum 1 data value per millisecond).

#### DECLARATION

---

```
public class TimeSeriesPlotter
extends ApplicationFrame
implements java.awt.event.ActionListener
```

## CONSTRUCTORS

- *TimeSeriesPlotter*

```
public TimeSeriesPlotter( java.lang.String title, java.lang.String
valueDesc, java.lang.String xDesc, java.lang.String yDesc, double
yRangeLower, double yRangeUpper, double fixedRange )
```

- Parameters

- \* title - Title of the Plot/Chart
- \* valueDesc - Description of the data
- \* xDesc - x axis description
- \* yDesc - y axis description
- \* yRangeLower - lower limit of the y axis
- \* yRangeUpper - upper limit of the y axis
- \* fixedRange - in milliseconds, range of the x axis

## METHODS

- *actionPerformed*

```
public void actionPerformed( java.awt.event.ActionEvent e )
```

- Parameters

- \* e -

- *addNewDataRecord*

```
public void addNewDataRecord( double value )
```

- Usage

- \* adds a new data record. this record is automatically timestamped with the actual time that this method is called

- Parameters

- \* value - double

- *close*

```
public void close( )
```

#### 4.2.10 CLASS TimeSeriesPlotterExtended

plotting time series (with multiple data sets).

## DECLARATION

```
public class TimeSeriesPlotterExtended
extends ApplicationFrame
implements java.awt.event.ActionListener
```

## CONSTRUCTORS

- *TimeSeriesPlotterExtended*  
public **TimeSeriesPlotterExtended**( )

## METHODS

- *actionPerformed*  
public void **actionPerformed**( java.awt.event.ActionEvent e )  
  
– **Parameters**  
\* e -

---

- *addDataPlot*  
public void **addDataPlot**( java.lang.String descr )

---

- *addNewDataRecord*  
public void **addNewDataRecord**( java.lang.String descr, double value )  
  
– **Usage**  
\* adds a new data record. this record is automatically timestamped with the actual time that this method is called  
  
The parameter descr specifies to which data series this value is added.  
  
– **Parameters**  
\* descr - Name of this time series  
\* value - value to add

---

- *close*  
public void **close**( )

---

- *init*  
public void **init**( java.lang.String title, java.lang.String xDesc,  
java.lang.String yDesc, double yRangeLower, double yRangeUpper,  
double fixedRange )  
  
– **Parameters**  
\* title - Title of the Plot/Chart  
\* xDesc - x axis description  
\* yDesc - y axis description  
\* yRangeLower - lower limit of the y axis  
\* yRangeUpper - upper limit of the y axis  
\* fixedRange - in milliseconds, range of the x axis

---

- *removeOldDataRecords*  
public void **removeOldDataRecords**( java.lang.String descr, int maxCount )  
  
– **Usage**  
\* removes old data records from a time series.

- \* **descr** - Name of this time series
- \* **maxCount** - max number of entries in this timeseries. counted from the latest entry towards the oldest entry.

# Chapter 5

## Package `aha.middleware`

<i>Package Contents</i>	<i>Page</i>
<hr/>	
<b>Classes</b>	
<b>Decision</b> .....	39
<i>Instances of this class are sent out by the control agent as content of a instance of the class Message.</i>	
<b>Message</b> .....	41
<i>generic able message (serializable), serves as generic container for values that should be sent arround between different agents.</i>	

---

## 5.1 Classes

### 5.1.1 CLASS Decision

---

Instances of this class are sent out by the control agent as content of a instance of the class Message. it contains all values associates with a decision. this class furthermore defines all possible decisions of the system as static variables.

#### DECLARATION

---

```
public class Decision
extends java.lang.Object
implements java.io.Serializable
```

#### SERIALIZABLE FIELDS

---

- private Map parameters  
–
- private int decision  
–
- private int type  
–
- private String roomName  
–

#### FIELDS

---

- public static int INVALID  
–
- public static int DECISION\_BLIND  
– possible type for a message  
  
decisions of this type contain only decisions concerning Blind's
- public static int DECISION\_BLIND\_UP  
– DECISION\_BLIND\_UP = decision that blinds should go up
- public static int DECISION\_BLIND\_DOWN  
– DECISION\_BLIND\_DOWN = decision that blinds should go down

- DECISION\_BLIND\_CHANGE = change position of blinds (neither 100% up nor 100% down, something between). A parameter supplied with this decision specifies the details of the decision.
- public static int DECISION\_LIGHT
  - possible type for a message
  - decisions of this type contain only decisions concerning Light's
- public static int DECISION\_LIGHT\_ON
  - DECISION\_LIGHT\_ON = light should be switched on
- public static int DECISION\_LIGHT\_OFF
  - DECISION\_LIGHT\_OFF = light should be switched off
- public static int DECISION\_RADIATION
  - possible type for a message
  - decisions of this type contain only decisions concerning Radiation's
- public static int DECISION\_OUTSIDE\_RADIATION
  - DECISION\_OUTSIDE\_RADIATION: new decision regarding the intensity of the outside radiation (taxonomically)

## CONSTRUCTORS

---

- *Decision*

```
public Decision( java.lang.String roomName, int decision, int type )
```

  - **Usage**
    - \* Creates a new instance of Decision.
  - **Parameters**
    - \* **roomName** - to which room does this decision belong to
    - \* **type** - type of decision (defined as static int's in this class, other values are invalid.
    - \* **decision** - decision taken. All possible decisions are defined as static fields of this class.

## METHODS

---

- *addParameter*

```
public void addParameter( java.lang.String name, java.lang.String value )
```

  - **Usage**
    - \* adds a parameter to this decision.
  - **Parameters**
    - \* **name** - name of the parameter (arbitrary)
    - \* **value** - value of this parameter (arbitrary)

- *getDecision*  
**public int getDecision( )**
  - **Usage**  
\* returns the decision identification code.
  - **Returns** - id of this decision

---
- *getParameters*  
**public Map getParameters( )**
  - **Usage**  
\* returns a map of all parameters carried by this decision.
  - **Returns** - map of all parameters, map is empty if there are none.

---
- *getRoomName*  
**public String getRoomName( )**
  - **Usage**  
\* Gets the roomName.
  - **Returns** - Returns a String

---
- *getType*  
**public int getType( )**
  - **Usage**  
\* returns the type of this decision
  - **Returns** - int (defined as static constants in this class)

### 5.1.2 CLASS Message

---

generic able message (serializable), serves as generic container for values that should be sent around between different agents.

#### DECLARATION

---

```
public class Message
extends java.lang.Object
implements java.io.Serializable
```

#### SERIALIZABLE FIELDS

---

- private Object arg  
–
- private String name  
–
- private int contentQualifier

## FIELDS

---

- public static int MESSAGEID\_PROPERTY\_CHANGE
  - messageid, used as identification of the AbleEvent containing this message
- public static int MESSAGEID\_VARIABLE\_UPDATE
  - messageid, used as identification of the AbleEvent containing this message
- public static int MESSAGEID\_DECISION
  - messageid, used as identification of the AbleEvent containing this message
- public static int MESSAGEID\_LOG
  - messageid, used as identification of the AbleEvent containing this message
- public static int ILLUMINANCE1
  - Content identifier
- public static int ILLUMINANCE2
  - Content identifier
- public static int HUMIDITY
  - Content identifier
- public static int SUNEAST
  - Content identifier
- public static int SUNWEST
  - Content identifier
- public static int SUNSOUTH
  - Content identifier
- public static int TEMPERATURE
  - Content identifier
- public static int PRESENCE\_MOVEMENT
  - Content identifier
- public static int LIGHT
  - Content identifier
- public static int BLIND
  - Content identifier
- public static int PRESENCE\_LIGHT
  - Content identifier
- public static int DECISION\_OBJECT

## CONSTRUCTORS

- *Message*

```
public Message( java.lang.String name, java.lang.Object arg )
```

- **Usage**

- \* construct a message without a type identifier

- **Parameters**

- \* **name** - name of this messages
  - \* **arg** - Content of the message

- *Message*

```
public Message( java.lang.String name, java.lang.Object arg, int
contentQualifier )
```

- **Usage**

- \* Construct a message with content Qualifier

- **Parameters**

- \* **name** - name of this message
  - \* **arg** - Content of the message
  - \* **contentQualifier** - id identifying the content (arguments) of this message.

## METHODS

- *getArgument*

```
public Object getArgument( )
```

- **Usage**

- \* return the content of this message.

- **Returns** - content

- *getContentQualifier*

```
public int getContentQualifier( )
```

- **Usage**

- \* return the content identifier of this message instance.

The content identifier is a int that is defined as a static field of this class.

- **Returns** - id identifying the content (arguments) of this message.

- *getName*

```
public String getName( )
```

- **Usage**

- \* return the name of this message.

- **Returns** - name

- *setContentQualifier*

– **Usage**

- \* set content identifier (id).

The content identifier is a int that is defined as a static field of this class.

– **Parameters**

- \* `contentQualifier` - id identifying the content (arguments) of this message.

## Chapter 6

# Package aha

<i>Package Contents</i>	<i>Page</i>
<hr/>	
<b>Interfaces</b>	
<b>BusRemote</b> .....	46
<i>Remote interface for rmi calls to BusAgent.</i>	
<b>Classes</b>	
<b>AhaServer</b> .....	48
<i>general server that runs the roomDisplayAgent (s)</i>	
<b>BossAgent</b> .....	48
<i>agent that deals with network events like new device attached etc.</i>	
<b>BusAgent</b> .....	49
<i>Responsible for the access to the building bus.</i>	
<b>BusServer</b> .....	51
<i>\$Id: BusServer.java,v 1.3 2002/06/12 17:53:42 urut Exp \$ \$Name: \$</i>	
<i>server for running the bus agent.</i>	
<b>Config</b> .....	52
<i>The Singleton for the config file access, name of the config file is hard coded to .ahaConfig</i>	
<b>ControlServer</b> .....	52
<i>Server that provides the runtime environment for a single RoomControlAgent.</i>	
<b>DisplayServer</b> .....	53
<i>Server that acts a the runtime environment for running the roomDisplay Agent(s).</i>	
<b>RoomDisplay</b> .....	54
<i>GUI of the roomdisplay agent.</i>	
<b>RoomDisplayAgent</b> .....	58
<i>Agent that shows all available data about one specific room.</i>	

---

## 6.1 Interfaces

### 6.1.1 INTERFACE BusRemote

---

Remote interface for rmi calls to BusAgent.

#### DECLARATION

---

```
public interface BusRemote
implements java.rmi.Remote
```

#### METHODS

---

- *getDescription*

```
public PropertyDescription getDescription( java.lang.String name )
```

  - **Usage**
    - \* Return a description of this Agent.
  - **Parameters**
    - \* name -
  - **Returns** -
  - **Exceptions**
    - \* java.rmi.RemoteException -

---
- *getLogQueue*

```
public Collection getLogQueue( )
```

  - **Usage**
    - \* return LogQueue of the Bus Agent (pending logs)
  - **Returns** -
  - **Exceptions**
    - \* java.rmi.RemoteException -

---
- *getProperties*

```
public Collection getProperties( java.lang.String regex )
```

  - **Usage**
    - \* Return properties which names fullfill the provided regular expression (JDK1.4 syntax)
  - **Parameters**
    - \* regex - jdk1.4 regex syntax
  - **Returns** - Collection of Strings
  - **Exceptions**
    - \* java.rmi.RemoteException -

---
- *getProperty*

- **Usage**
    - \* get the value of a property.
  - **Parameters**
    - \* name -
  - **Returns** -
  - **Exceptions**
    - \* java.rmi.RemoteException -
- 

- *propertyChange*

```
public void propertyChange( java.beans.PropertyChangeEvent evt )
```

- **Parameters**
    - \* evt -
  - **Exceptions**
    - \* java.rmi.RemoteException -
- 

- *registerListener*

```
public void registerListener( AbleEventListener listener, java.lang.String name )
```

- **Usage**
    - \* add Agent to the listener List of one network variable.
  - **Parameters**
    - \* listener -
    - \* name -
  - **Exceptions**
    - \* java.rmi.RemoteException -
- 

- *removeListener*

```
public void removeListener( AbleEventListener listener, java.lang.String name )
```

- **Usage**
    - \* Remove Agent from the listener list
  - **Parameters**
    - \* listener -
    - \* name -
  - **Exceptions**
    - \* java.rmi.RemoteException -
- 

- *setProperty*

```
public void setProperty( java.lang.String name, java.lang.String value )
```

- **Usage**
  - \* set the value of a property (network variable).
- **Parameters**
  - \* name - name of the network variable
  - \* value - value that should be set
- **Exceptions**

## 6.2 Classes

### 6.2.1 CLASS AhaServer

---

general server that runs the roomDisplayAgent (s)

#### DECLARATION

---

```
public class AhaServer
extends java.lang.Object
```

#### CONSTRUCTORS

---

- *AhaServer*  
 public **AhaServer**( )
  - **Usage**
    - \* Creates a new instance of AhaServer

#### METHODS

---

- *init*  
 public void **init**( java.lang.String **busAgentServer**, java.lang.String **historyAgentServer** )
- *main*  
 public static void **main**( java.lang.String [] **args** )

### 6.2.2 CLASS BossAgent

---

agent that deals with network events like new device attached etc.

#### DECLARATION

---

```
public class BossAgent
extends AbleDefaultAgent
implements java.rmi.Remote
```

#### CONSTRUCTORS

---

- *BossAgent*  
 public **BossAgent**( )

## METHODS

- *init*  

```
public void init( java.lang.String  busAgentServer, java.lang.String
  historyServer )
```

---
- *reset*  

```
public void reset( )
```

### 6.2.3 CLASS BusAgent

---

Responsible for the access to the building bus. Provides methods to get variable values for the home automation devices and subscribing for variable changes. Tries to mimic the JavaBeans naming convention.

The implementation for the businterface is read from the config file

## DECLARATION

```
public class BusAgent
extends AbleDefaultAgent
implements java.beans.PropertyChangeListener, BusRemote
```

## CONSTRUCTORS

- *BusAgent*  

```
public BusAgent( )
```

## METHODS

- *getDescription*  

```
public PropertyDescription getDescription( java.lang.String  name )
```

  - **Usage**  
 \* what the hell does this function !!!

---
- *getLogQueue*  

```
public Collection getLogQueue( )
```

  - **Usage**  
 \* returns and empties the Queue of events which have not been logged so far
  - **Returns** - Collection of LogArgs Objects which should be logged
  - **See Also**  
 \* `aha.history.LogArgs` ( in 3.2.3, page 21)

---

- *getProperties*  

```
public Collection getProperties( java.lang.String regex )
```

  - **Usage**
    - \* returns the names of all bus variables matching a certain regular expression
  - **Parameters**
    - \* **regex** - Regular expression for the variable names
  - **Returns** - Collection of variable names

---
- *getProperty*  

```
public String getProperty( java.lang.String name )
```

  - **Usage**
    - \* gets the value for a bus variable
  - **Parameters**
    - \* **name** - variable whose value should be get

---
- *init*  

```
public void init( )
```

  - **Usage**
    - \* Initializes the Agent.  
Connects to BusController, reads from Config file which BusController implementation should be used. Binds itself into the RMIRegistry

---
- *propertyChange*  

```
public void propertyChange( java.beans.PropertyChangeEvent evt )
```

  - **Usage**
    - \* callback for propertyChanges.  
called whenever a variable value which it is subscribed for changes.
  - **See Also**
    - \* `aha.bus.BusController.registerPropertyChangeListener(String, PropertyChangeListener)`

---
- *registerListener*  

```
public void registerListener( AbleEventListener listener, java.lang.String name )
```

  - **Usage**
    - \* registers a listener for a bus variable
  - **Parameters**
    - \* **listener** - listener for variable changes
    - \* **name** - variable name

---
- *removeListener*  

```
public void removeListener( AbleEventListener listener, java.lang.String name )
```

  - **Usage**

\* removes a listener from the list of listeners for a bus variable.

removes itself as a listener from busController if there are no more listeners for this variable

– **Parameters**

\* **listener** - listener to be removed  
 \* **name** - variable where it should be removed as a listener

---

• *setProperty*

`public void setProperty( java.lang.String name, java.lang.String value )`

– **Usage**

\* sets the value for a bus variable

– **Parameters**

\* **name** - variable which should be set  
 \* **value** - new value for the variable

## 6.2.4 CLASS BusServer

---

\$Id: BusServer.java,v 1.3 2002/06/12 17:53:42 urut Exp \$ \$Name: \$

server for running the bus agent.

### DECLARATION

---

```
public class BusServer
extends java.lang.Object
```

### CONSTRUCTORS

---

• *BusServer*

`public BusServer( )`

### METHODS

---

• *init*

`public void init( )`

– **Usage**

\* Initialize Bus agent

---

• *main*

`public static void main( java.lang.String [] args )`

– **Parameters**

\* **args** - takes no parameters

### 6.2.5 CLASS Config

---

The Singleton for the config file access, name of the config file is hard coded to .ahaConfig

#### DECLARATION

---

```
public class Config
extends java.lang.Object
```

#### CONSTRUCTORS

---

- *Config*  
public **Config**( )

#### METHODS

---

- *getInstance*  
public static **Config getInstance**( )  
– **Usage**  
\* returns the singleton instance for Config

---

- *getProperty*  
public static **String getProperty**( java.lang.String name )  
– **Usage**  
\* returns the value for a config property

---

- *setProperty*  
public static void **setProperty**( java.lang.String name, java.lang.String value )  
– **Usage**  
\* sets the value for a property in the config file.

write back to the config file is not implemented yet, so it only changes the value for the running application

### 6.2.6 CLASS ControlServer

---

Server that provides the runtime environment for a single RoomControlAgent.

#### DECLARATION

---

```
public class ControlServer
extends java.lang.Object
```

## CONSTRUCTORS

- *ControlServer*  
public **ControlServer**( )

## METHODS

- *init*  
public void **init**( java.lang.String **historyServer**, java.lang.String **busAgentServer**, java.lang.String **roomName** )  
    - **Usage**  
\* Initialize Control Agent, start event processing.
    - **Parameters**  
\* **historyServer** - hostname of the History Server  
\* **busAgentServer** - Hostname of the BusAgentServer  
\* **roomName** - Identification of the Room which this control agent should control
- 
- *main*  
public static void **main**( java.lang.String [] **args** )  
    - **Parameters**  
\* **args** -

## 6.2.7 CLASS DisplayServer

Server that acts a the runtime environment for running the roomDisplay Agent(s).

```
$Log: DisplayServer.java,v $
Revision 1.4  2002/07/05 14:49:17  urut
*** empty log message ***
```

```
Revision 1.3  2002/06/12 17:53:42  urut
lots of docs added
```

```
Revision 1.2  2002/05/29 15:07:40  urut
moved to independent display server
```

```
Revision 1.1  2002/05/29 14:44:54  urut
initial release of an independent server for the display
```

## DECLARATION

```
public class DisplayServer
extends java.lang.Object
```

## CONSTRUCTORS

- *DisplayServer*  
**public DisplayServer( )**
  - **Usage**
    - \* Creates a new instance of this Server.

## METHODS

- *init*  
**public void init( java.lang.String busAgentServer, java.lang.String controlAgentServer )**
    - **Usage**
      - \* Initialize the Server, instantiate Agents, start event queue processing
    - **Parameters**
      - \* busAgentServer -
      - \* controlAgentServer -
- 
- *main*  
**public static void main( java.lang.String [] args )**
    - **Parameters**
      - \* args - Parameters are: busAgentServerName controlAgentServerName

## 6.2.8 CLASS RoomDisplay

GUI of the roomdisplay agent. GUI is generated by NetBeans, logic glue code is in here as well that ties this to the displayagent.

## DECLARATION

```
public class RoomDisplay
extends javax.swing.JFrame
```

## SERIALIZABLE FIELDS

- private RoomDisplayAgent agent
  -
- private JLabel temp
  -

- 
- private JPanel jPanel10
- 
- private JButton jButton4
- 
- private JLabel currentRoom
- 
- private JButton jButton3
- 
- private JButton jButton2
- 
- private JButton jButton1
- 
- private JLabel jLabel17
- 
- private JLabel jLabel15
- 
- private JLabel valueBlind
- 
- private JLabel jLabel13
- 
- private JTextField angle
- 
- private JLabel jLabel11
- 
- private JLabel sunWest
- 
- private JButton buttonLight2
- 
- private JButton buttonLight1
- 
- private JLabel sunSouth
-

- 
- private JPanel light1Color
- 
- private JLabel humidity
- 
- private JComboBox blinds
- 
- private JButton stopButton
- 
- private JPanel light2Color
- 
- private JPanel jPanel9
- 
- private JPanel jPanel8
- 
- private JPanel jPanel7
- 
- private JPanel jPanel6
- 
- private JPanel jPanel5
- 
- private JPanel jPanel4
- 
- private JPanel jPanel3
- 
- private JPanel jPanel2
- 
- private JPanel jPanel1
- 
- private JLabel sunEast
- 
- private JLabel jLabel9
-

- 
- private JLabel jLabel6
- 
- private JLabel jLabel5
- 
- private JLabel jLabel4
- 
- private JLabel jLabel3
- 
- private JLabel jLabel2
- 
- private JLabel jLabel1
- 
- private JLabel illuminance2
- 
- private JLabel illuminance1
- 
- private JLabel valueLight2
- 
- private JLabel valueLight1
- 
- private JTextField roomName
- 
- private JLabel chosenBlind
- 

## CONSTRUCTORS

---

- *RoomDisplay*  
public **RoomDisplay**( )
  - **Usage**
    - \* Creates new form RoomDisplay

## METHODS

- *addBlind*  
public void addBlind( java.lang.String name )
- *main*  
 public static void main( java.lang.String [] args )  
     – **Parameters**  
     \* args - the command line arguments
- *removeAllBlinds*  
public void removeAllBlinds( )
- *setBlind*  
public void setBlind( java.lang.String chosenBlindText, java.lang.String valueBlindText )
- *setColorLight1*  
public void setColorLight1( boolean state )
- *setColorLight2*  
public void setColorLight2( boolean state )
- *setRoomDisplayAgent*  
public void setRoomDisplayAgent( aha.RoomDisplayAgent agent )
- *setValueLight1*  
public void setValueLight1( java.lang.String valueLight1Value )
- *setValueLight2*  
public void setValueLight2( java.lang.String valueLight2Value )
- *setWeatherData*  
 public void setWeatherData( java.lang.String illuminance1Text,  
     java.lang.String illuminance2Text, java.lang.String humidityText,  
     java.lang.String sunEastText, java.lang.String sunSouthText,  
     java.lang.String sunWestText, java.lang.String tempText )

## 6.2.9 CLASS RoomDisplayAgent

Agent that shows all available data about one specific room. it furthermore has the capability to manually interact with all devices of a room and displays statistical data showing the historical values of various devices and decisions taken.

## DECLARATION

```
public class RoomDisplayAgent
extends AbleDefaultAgent
```

## FIELDS

- 
- public static int LIGHT1  
–
  - public static int LIGHT2  
–
  - public static int UNDEFINED  
–
  - public static int ON  
–
  - public static int OFF  
–

## CONSTRUCTORS

- 
- *RoomDisplayAgent*  
public **RoomDisplayAgent**( )

## METHODS

- 
- *action*  
public void **action**( int **what**, java.lang.String **name** )  
– **Usage**  
\* take action (as substitute for the GUI).  
  
what: 1=light1, 2=light2 (defined in Room, static int constants)  
– **Parameters**  
\* **what** - action to take  
\* **name** - value

---

  - *action*  
public void **action**( int **what**, java.lang.String **name**, java.lang.String **parameter1**, java.lang.String **parameter2** )  
– **Usage**  
\* take action (as substitute for the GUI).  
  
what: defined in Room (static constants)  
– **Parameters**  
\* **what** - decision to take  
\* **name** -

---

\* parameter2 -

---

- *changeRoom*

```
public void changeRoom( java.lang.String newRoom )
```

- **Usage**

- \* change the room for which this agent is displaying data. This method first desubscribes all network variables of the old room (if there was one) and then tries to subscribe to the network variables belonging to the new room requested.

This method is called by the gui.

- **Parameters**

- \* **newRoom** - room Identification

- **Exceptions**

- \* `java.rmi.RemoteException` -

---

- *chooseBlind*

```
public void chooseBlind( java.lang.String name )
```

- **Usage**

- \* Choose active blind (blind that is currently controlled by this Agent).

gets called from gui.

- **Parameters**

- \* **name** -

---

- *handleAbleEvent*

```
public void handleAbleEvent( AbleEvent theAbleEvent )
```

- **Usage**

- \* events received from able are handed over to this method by able.

- **Parameters**

- \* **theAbleEvent** - event

---

- *init*

```
public void init( java.lang.String busAgentServer, java.lang.String controlAgentServer )
```

- **Usage**

- \* Initializes the Agent.

Able standard method, see able docu for explanations.

- **Parameters**

- \* **busAgentServer** -

- \* **controlAgentServer** -

- **Exceptions**

- \* `java.rmi.RemoteException` -

---

- *processTimerEvent*

– **Usage**

\* Called whenever able sends a timer event to this Agent.

– **Exceptions**

\* `java.rmi.RemoteException` -

---

• *reset*

`public void reset( )`

– **Usage**

\* Reset the Agent, reset all state values to the initial values.

Able standard method, see able docu for explanations

– **Exceptions**

\* `java.rmi.RemoteException` -

---

• *start*

`public void start( )`

– **Usage**

\* Start the agent.

---

• *updateLightDisplay*

`public void updateLightDisplay( )`

– **Usage**

\* Called when new values are available (from GUI).

react to actions from GUI.

# Chapter 7

## Package `aha.bus.lonworks`

<i>Package Contents</i>	<i>Page</i>
<hr/>	
<b>Classes</b>	
<b>LNSController</b> .....	63
<i>BusController implementation for LonNetwork</i>	

---

## 7.1 Classes

### 7.1.1 CLASS LNSController

---

BusController implementation for LonNetwork

#### DECLARATION

---

```
public class LNSController
extends java.lang.Object
implements aha.bus.BusController
```

#### CONSTRUCTORS

---

- *LNSController*  
public **LNSController**( )

#### METHODS

---

- *connect*  
public void **connect**( )
  - **Usage**
    - \* connect to the lonbus gateway and loads all network variables.
    - the ip-address and port are read from the config file
  - **See Also**
    - \* `aha.Config` ( in 6.2.5, page 52)
    - \* `aha.bus.lonworks.LNSController.loadNetworkVariables()`
- *getDescription*  
public PropertyDescription **getDescription**( java.lang.String name )
- *getProperties*  
public Collection **getProperties**( java.lang.String regex )
- *getProperty*  
public String **getProperty**( java.lang.String name )
- *main*  
public static void **main**( java.lang.String [] arguments )
- *onNetworkVariableUpdate*  
public void **onNetworkVariableUpdate**( LNSNVUpdateEvent e )
- *onRawUpdate*  
public void **onRawUpdate**( LNSRawUpdateEvent e )

- *onUpdateError*  
public void **onUpdateError**( LNSUpdateErrorEvent e )
- *registerPropertyChangeListener*  
public void **registerPropertyChangeListener**( java.lang.String name,  
java.beans.PropertyChangeListener listener )
- *registerPropertyChangeListenerTest*  
public void **registerPropertyChangeListenerTest**( java.lang.String name )
- *removePropertyChangeListener*  
public void **removePropertyChangeListener**( java.lang.String name,  
java.beans.PropertyChangeListener listener )
- *setProperty*  
public void **setProperty**( java.lang.String name, java.lang.String value )