



TWaver[®] Java

Developer Guide

Version 4.0

Jun 2011

Serva Software

info@servasoftware.com

<http://www.servasoftware.com>

PO Box 8143, Wichita Falls, Texas, USA 76307



For more information about Serva Software and TWaver please visit the web site at:

<http://www.servasoftware.com>

Or send e-mail to:

info@servasoftware.com

Jun, 2011

Notice:


This document contains proprietary information of Serva Software. Possession and use of this document shall be strictly in accordance with a license agreement between the user and Serva Software, and receipt or possession of this document does not convey any rights to reproduce or disclose its contents, or to manufacture, use, or sell anything it may describe. It may not be reproduced, disclosed, or used by others without specific written authorization of Serva Software.

TWaver, servasoft, Serva Software and the logo are registered trademarks of Serva Software. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S.A. and other countries. Other company, brand, or product names are trademarks or registered trademarks of their respective holders. The information contained in this document is subject to change without notice at the discretion of Serva Software.

Copyright © 2011 Serva Software LLC

All Rights Reserved

Table of Contents

- TWaver Java Developer Guide 
 - Overview
 - Introduction
 - TWaver Overview
 - TWaver Structure
 - Graphic Components
 - Network Component
 - Tree Component
 - Table Component
 - TreeTable Component
 - PropertySheet Component
 - List Component
 - Chart Component
 - DataBox Overview
 - Managed Objects
 - Predefined Managed objects
 - Structure of Managed objects
 - Properties of Managed Objects
 - Predefined Managed Objects Overview
 - Alarm Support
 - Alarm Representation in TWaver
 - Using Alarm Severities
 - Development Overview
 - Install Java and TWaver
 - TWaver Configuration
 - Creating Data by XML or API
 - Creating Data by API
 - Creating Data by XML
 - Comparing XML and API
 - Basic Programming
 - What All Developers Should Know
 - Executing the Example Application
 - Creating a Basic Network Component
 - Creating a Basic Tree Component
 - Showing Equipment Chassis
 - Creating Chassis by API
 - Creating Chassis by XML
 - Element Selection
 - Using Selection State
 - Using Selection Model
 - Using Selection Listener
 - Using Selection Interaction
 - Customizing Interactions
 - Adding Popup Menu Generator

- Adding Mouse & Keyboard Action
 - Handling Selection
- Adding Alarms
- Adding Decorated Icons
- Adding Special Effect
- Predefined Managed Object
 - Common Features
 - Using HTML Label
 - Using Alpha Transparent
 - Using Node Element
 - Creating Node by API
 - Creating Node by XML
 - Using ShapeNode Element
 - Using Link Element
 - Creating Link by API
 - Creating Link by XML
 - Working with Link Bundle
 - Using Link Bundle Agent
 - Using Link Type
 - Using Group Element
 - Using Group
 - Creating Group by XML
 - Group Properties
 - Using SubNetwork Element
 - What is SubNetwork
 - Difference between SubNetwork and Group
 - Inheritance of SubNetwork
 - Using TSubNetwork Background
 - Using SubNetwork Data File
 - SubNetwork Client Properties
 - Creating SubNetwork via Shape
 - Using BTS & BTSAntenna Element
 - Equipment Elements
 - Grid Element
 - Chassis Element
 - Rack Element
 - Shelf Element
 - Slot Element
 - Card Element
 - Port Element
 - Extending Equipment Element
 - Using Follower Element
 - Using Element Properties
 - Using User Properties
 - Using Business Object

- Element Self-Copy
 - Element Self-Copy Overview
 - Using Interceptor
- Using DataBox
 - Changing Data of DataBox
 - Visiting Data of DataBox
 - Using ElementCallbackHandler Simplify Traversal DataBox
 - Using DataBox Listener
 - Using Element Property Change Listener
 - Using SelectionChangeInterceptor
 - Using DataBox LayerModel
 - LayerModel and Layer
 - Managing Layer via LayerModel
 - Using LayerModelListener
 - Using Layer
 - Predefined properties
 - Layer Client Properties
 - Monitoring Layer Property Change Event
 - Element and Layer
 - Serializing Layer to XML
 - Layer on Network
 - Using LayerManagerPane
 - On-Demand-Load
 - DataBox and Multi-Thread
 - Customizing Alarm Propagator
 - Using QuickFinder
 - Undo and Redo Support
 - Using UndoRedoManager
 - Using UndoRedoListener
 - Using UndoRedoInterceptor
- Using Components
 - TWaver Resources Redirection
 - TWaver Task Scheduler
 - Scheduler and Task
 - Swing and Threads
 - Components Comment Features
 - AutoScroll
 - Exporting Images
 - Popup Component
 - Using Network Component
 - Customizing Toolbar
 - Network Component Hierarchy
 - Network Component MVC Model
 - Network Coordinate System
 - Getting Elements By Location

- Network Viewport & Other Methods
- Network Printing
- Network Representation
 - Attaching Swing Components in Network
 - Customizing Network Background
 - Network Canvas Interaction
 - Customizing Default Mouse Action
 - Customizing Popup Menu
 - Interaction Mode
 - Listening Keyboard & Mouse Event
 - Using Copy & Paste Shortcut Key
 - Using Alpha Transparent.
 - Using Attachment
 - Using ElementUI
 - Using Filters
 - ElementBoundsInvalidatableFilter
 - ElementLabelEditableFilter
 - ElementPropertyChangeRepaintFilter
 - PaintSelectionModeFilter
 - SelectableFilter
 - SendToTopFilter
 - Using DoubleClickableFilter
 - Using MovableFilter
 - Using ResizableFilter
 - Using VisibleFilter
 - Using Generator
 - Using Link Layouter
 - Using Network Layout
- Operations for Network
 - Full Screen Network
 - Network Overview
 - Using Dragging Speed
 - Using Magnifier
 - Using Network Zoomer
- Using Blinking Rule
- More Network Component Features
- Using Tree Component
 - Check-Box Tree Node
 - Creating Tree Component
 - Customizing Tree Node Handle Icon
 - Customizing Tree Node Renderer
 - Introducing the Tree Component
 - Iterating Tree Node
 - Lazy Loading

- Size of Tree Node
- Sorting the Tree Node
- Tooltip of Tree Node
- Tree Node Double Clicked Listening
- Tree Node Drag & Drop
- Tree Node Index Control
- Tree Popup Menu
- Tree Root Shift
- TWaver Tree Node
- Using Tree Generators
- Using Table Component
 - Creating Table Component
 - Handling Data with TableModel
 - Hiding Table Columns
 - Locking Table
 - More Tips for Table
 - Resizing Table Column & Row
 - Table Cell Editor
 - Table Cell Renderer
 - Table Column Sorting
 - Table Paging
 - Table Popup Menu
 - Using Internal Columns
 - Check Column
 - OID Column
 - Using Table Columns
 - Using Table Filter
 - Using Table Listener
 - Using Table Navigator
- Using Element Table Component
 - Creating Element Table Component
 - Customizing Element Table Component
 - Getting Elements from Table
 - Loading Data
 - Using SendToTop Filter and SendToBottom Filter
 - Using Visible Filter
- Using TreeTable Component
 - Creating Tree Table Component
 - Using Tree Table Component
- Using Alarm Table Component
 - Creating Alarm Table Component
 - Customizing Alarm Table Component
 - Operations for TAlarmTable
 - Using AlarmVisibleFilter
















- Working with DataBox and AlarmModel
- Using AlarmOverview Component
 - Alarm Statistic Table
 - Creating AlarmOverview Component
- Using List Component
 - Creating List Component
 - Using Checkable Filter
 - Using List Visible Filter
 - Using Selection Mode
 - Using Sort Comparator
- Using PropertySheet Component
 - BeanInfo and BeanInfoLoader
 - BeanInfo Format
 - Customizing PropertySheet Style
 - Define BeanInfo by XML
 - Extending BeanInfo
 - Properties Grouping
 - Grouping by API
 - Grouping by XML
 - PropertySheet Events
 - TPropertySheet Cell Editor
 - Customizing Editor
 - Managing Editor
 - Default Editor
 - TPropertySheet Cell Renderer
 - Customizing PropertySheet Cell Renderer
 - Managing PropertySheet Cell Renderers
 - Predefined Renderers
 - TWaver and JavaBeans
 - Using Property Sheet Pane
- Using Chart Component
 - Using BarChart
 - Using Bubble Chart
 - Using DialChart
 - Using LineChart
 - Using PercentChart
 - Using PieChart
 - Using Radar Chart
- Using Alarm
 - Alarm Object Definition
 - Extending Alarm Properties
 - Displaying Alarm with AlarmUI
 - Using AlarmType
 - Using AlarmTrendIndication
 - Using AlarmState

- Using AlarmSeverity
 - Customizing AlarmSeverity
 - Customizing AlarmSeverity Comparator
 - Customizing Renderer Color for Different Severity Levels
- Using Alarm Model
 - Managing Alarms
 - Listening Alarm Model
 - Listening Alarm Property Changing
 - Using Alarm Mapping
 - Using AlarmModelQuickFinder
- Using AlarmStateStatistics
- Creating Alarm by XML
- Creating Alarm by API
- Customizing Alarm Renderer Color
- Using TWaverUtil
 - Using Resource Agent
- Working with XML
 - XML Long-Term Persistence for JavaBeans
 - Creating Data by XML.
 - Updating Data by XML
 - Deleting Data by XML
 - Using XML Persistent Filter
 - Transient Element Property
 - Using TWaver.XML
 - Alarm Severity Configuration
 - Icon Attachment Definition
 - Property Sheet Category Definition
 - Network Toolbar Configuration
 - UI Default Configuration
- Internationalization
 - Changing Internal i18n Strings
 - Adding i18n Strings
 - Adding I18n Element Property
 - Customizing i18n
- TWaver Java FAQ
 - Chart FAQ
 - Can I make a popup menu for a Chart?
 - What's the differences between Chart Value and List Values?
 - Common FAQ
 - Why can not I change the default locale of TWaver?
 - DataBox Element FAQ
 - Are `element.putClientProperty("name", "ABC")` and `element.setName("ABC")` same?
 - How to combine a packaged Element class with a non-packaged ElementUI class?
 - How to use DataBoxQuickFinder to find element or alarm in DataBox?

- In TWaver, what's the difference between Border Outline & StateOutline?
 - What happened after removing a element from databox
 - Why can not I add element property change listener to a databox
- License FAQ
 - How to use license
- Network FAQ
 - Can alarm or message attachment be displayed in front of any elements?
 - Can I change the flashing speed for the blinking elements?
 - Can I use animated GIF for node image?
 - I add KeyListener or MouseListener to Network, why it does not work?
- Others FAQ
 - How to connect X11 window server?
 - How to intercept TWaver exception?
 - How to make interaction between Applet and JavaScript?
 - How to set image file path?
 - My web application runs fine on my Windows development machine, but when I deploy it to the Unix or Linux production server, it doesn't work. What is the problem?
- Swing FAQ
 - How to change the line style of TWaver Tree nodes?
 - Why I change table column width, it does not work?
- Table Tree Sheet FAQ
 - How to add tree expand listener
 - How to set tree column's tooltip for tree table
 - I'm using tristate tree. Can I make the parent node unselected when there is no child node selected?
 - In TTableModel, what's the difference between methods getPublishedColumn() and getPublishedData()?
- Appendix
 - TWaver Element Property List
 - AbstractElement Property List
 - BaseElement Property List
 - Node Property List
 - Group Property List
 - Link Property List
 - Other Nodes
 - Other Links
 - Class ElementAttribute
 - ElementAttribute Overview
 - Register ElementAttribute
 - Element Attribute Property Key
 - More Settings of Element Attribute
 - An Example for Using Element Attribute
 - Major Interfaces in TWaver
 - Equipment Interface Method List
 - Resizable Interface Method List

- TSubNetwork Interface Method List
- Network Related Interfaces
- TWaver Property Key List

TWaver Java Developer Guide

TWaver Java Developer Guide	Recently Updated
<div data-bbox="165 448 560 528"> <h2>Basic Programming</h2> </div> <p>New to TWaver Java? You can take this tutorial to have a quick start. This tutorial will take you step by step through the process of building a Swing application with TWaver components.</p> <div data-bbox="165 701 560 781"> <h2>TWaver Java FAQ</h2> </div> <p>Can not find answer? Take a look at here you can find most frequently asked questions about TWaver Java. This collection of FAQs provides brief answers to many common questions about TWaver Java, Swing programming, and Java2D.</p>	<p>by haley.deng (2 minutes ago)</p> <p> Adding I18n Element Property (TWaver Java Developer Guide (English))</p> <p>by haley.deng (11 minutes ago)</p> <p> Network Component (TWaver Java Developer Guide (English))</p> <p>by haley.deng (11 minutes ago)</p> <p> Predefined Managed Objects Overview (TWaver Java Developer Guide (English))</p> <p>by haley.deng (13 minutes ago)</p> <p> Using BarChart (TWaver Java Developer Guide (English))</p> <p>by haley.deng (18 minutes ago)</p> <p> Other Nodes (TWaver Java Developer Guide (English))</p> <p>by haley.deng (26 minutes ago)</p> <p> Register ElementAttribute (TWaver Java Developer Guide (English))</p> <p>by haley.deng (28 Jun)</p> <p> Working with Link Bundle (TWaver Java Developer Guide (English))</p> <p>by haley.deng (28 Jun)</p> <p> Creating Node by API (TWaver Java Developer Guide (English))</p> <p>by haley.deng (27 Jun)</p> <p> Using Selection Listener (TWaver Java Developer Guide (English))</p> <p>by haley.deng (27 Jun)</p> <p> Using Selection Model (TWaver Java Developer Guide (English))</p> <p>by haley.deng (27 Jun)</p> <p> Using Selection State (TWaver Java Developer Guide (English))</p> <p>by haley.deng (27 Jun)</p> <p> Creating Chassis by XML (TWaver Java Developer Guide (English))</p> <p>by haley.deng (27 Jun)</p> <p> Basic Programming (TWaver Java Developer Guide (English))</p> <p>by haley.deng (27 Jun)</p> <p> Comparing XML and API (TWaver Java Developer Guide (English))</p> <p>by haley.deng (27 Jun)</p> <p> Creating Data by XML (TWaver Java Developer Guide (English))</p>
TWaver News	



This document contains proprietary information of Serva Software. Possession and use of this document shall be strictly in accordance with a license agreement between the user and Serva Software, and receipt or possession of this document does not convey any rights to reproduce or disclose its contents, or to manufacture, use, or sell anything it may describe. It may not be reproduced, disclosed, or used by others without specific written authorization of Serva Software.

TWaver, servasoft, Serva Software and the logo are registered trademarks of Serva Software. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S.A. and other countries. Other company, brand, or product names are trademarks or registered trademarks of their respective holders. The information contained in this document is subject to change without notice at the discretion of Serva Software.
Copyright © 2009 Serva Software LLC, All Rights Reserved

Overview

TWaver is a set of Java Swing components built on the Java 2 platform, including a MVC framework. TWaver provides several predefined Swing components to present network topology data for software developers. TWaver is super lightweight, fast and easy-to-use. TWaver can be used in any applications which need to present network topology or equipment monitoring, like network management system (NMS), telecom operation support system, or resource management system. TWaver also can be used in any Swing based programs with the enhanced Swing components like Tree, Table, List, PropertySheet, Chart etc.

Naming Conventions

TWaver prefixes all major component classes with 'T' to avoid conflicts with other applications or packages. All method names begin with a lowercase letter.

In this document, we use conventional name to instead of the class names. The following table lists some conventions and the classes/description to which they refer to.

Conventional Name	Description
DataBox	twaver.TDataBox
Network	twaver.network.TNetwork
Tree	twaver.tree.TTree
Table	twaver.table.TTable
AlarmTable	twaver.table.TAlarmTable
ElementTable	twaver.table.TElementTable
PropertySheet	twaver.table.TPropertySheet
AlarmOverview	twaver.table.TAlarmOverview
List	twaver.list.TList
OSS	Operation Support System
NMS	Network Management System
GUI	Graphical User Interface
MVC	Model-View-Controller

Contacting SERVA Software

Customer input is essential for successful product development. Contact us with comments and questions, and visit our website regularly for additional information and new product announcements.

Website

<http://www.servasoftware.com>

Mailing address

P.O. Box 8143Wichita Falls,TX. 76307

U.S.A.

Telephone and fax

Tel: 918-698-1644

Fax: 208-247-4258

E-Mails

info@servasoftware.com

Introduction

TWaver Java is a set of Java Swing components for rapidly developing GUI for Swing based programs, especially OSS-like software.

TWaver provides a set of ready-to-use Java Swing components and a MVC framework to build telecom OSS GUI. TWaver reduces the complexity of the high-performance and high-quality OSS GUI development. It helps the telecom equipment vendors or independent telecom software companies to enhance the GUI quality, performance and aesthetic feeling of telecom software used in telecom field. It speeds up the product or prototype development and deployment, helps you to occupy the market quickly.

The Advantage for TWaver

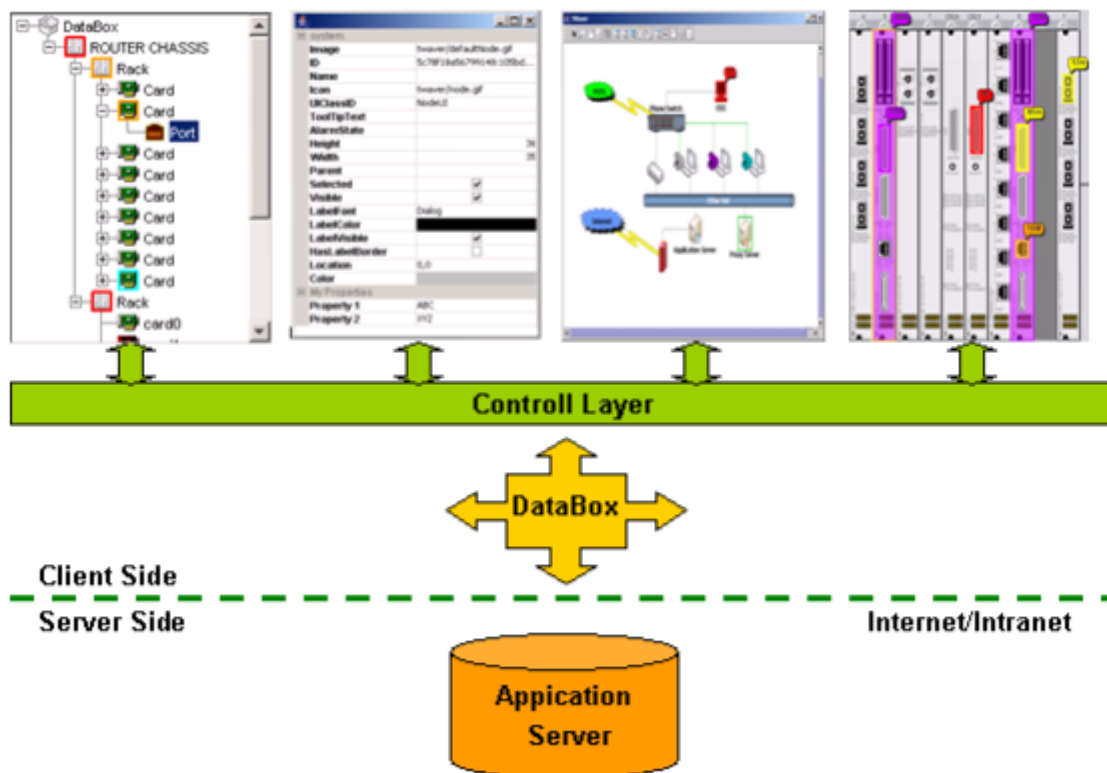
- Pure Java components: TWaver is 100% based on Java and Swing. It supports all platforms that support Java.
- MVC framework: TWaver provides well-designed MVC framework to let developers easy to use, extend and customize in applications.
- High performance lightweight graphical engine: TWaver is designed for large scale network representation. It can deal with thousands of nodes easily.
- A set of predefined managed objects: Including network element, link and equipment chassis elements like rack, card, port and other managed objects often used in Telecom applications.
- Supports XML data source, easy to integrate your application to other back end systems.
- Provides network/equipment data editor application, it is easy to compose network topology and equipment template.
 - [TWaver Overview](#)
 - [TWaver Structure](#)
 - [Graphic Components](#)
 - [DataBox Overview](#)
 - [Managed Objects](#)
 - [Alarm Support](#)

TWaver Overview

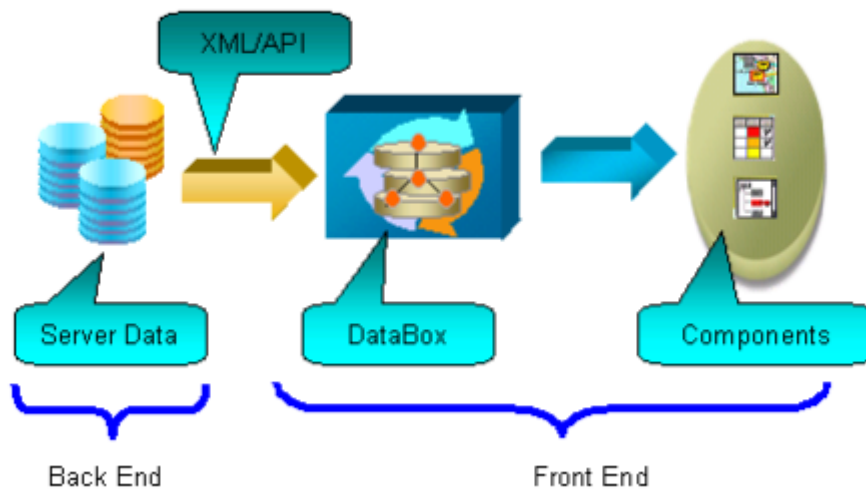
TWaver is based on the Java platform and Swing technology. TWaver takes data from a variety of back-end sources, typically OSS applications and displays the data on all components. TWaver makes it easy to represent network data into consistent, high-quality graphics that display across various graphic components.

TWaver can be integrated with any back-end applications, like XML stream, serialized Java objects.

TWaver provides many predefined graphic components: network, tree, property sheet, table, list chart and others. The Swing components connect to the back end through a DataBox that transforms data into business objects. Graphic components render these objects graphically by retrieving associated graphic properties like color, font, image, state and others.



TWaver Structure



TWaver is a set of Swing components based on Java. It takes data from a variety of back end systems, such as OSS applications or database systems, and transforms the data into consistent, high-quality graphics that display across various graphic components. The customer can use and extend the predefined managed objects to represent more complex telecom objects. DataBox drive all connected graphical components like network, chassis, tree, or table automatically. All components are customizable to meet your special requirements.

TWaver supplies predefined graphical components like network, tree and table. More components will be provided in the further releases. All components are designed following MVC (Model-View-Controller) structure. All graphical components act as a view of the data model(DataBox). You can make multiple components share the same network data to display a consistent representation.

TWaver provides a set of predefined business classes that you can use directly in your applications. These classes are specifically designed for telecom OSS systems to ease the development and leverage the overall graphic quality and the ergonomics of user interfaces in telecommunication applications. For node, link, group, subnetwork, equipment rack, card, port etc., all you need to do is create and add these business class instances to TWaver components. All these objects will be translated into high quality graphic representations with a common look-and-feel in all the TWaver graphic components. Predefined managed objects include the default graphic renderer that provides the common graphic representations for you automatically, thus significantly minimizing your coding efforts. Developers can create these elements or load it from other back-end systems. They can be loaded into data container by API or XML data stream.

Graphic Components

TWaver provides a set of ready-to-use Swing components for OSS application developers. In current version of TWaver, the major Swing components are:

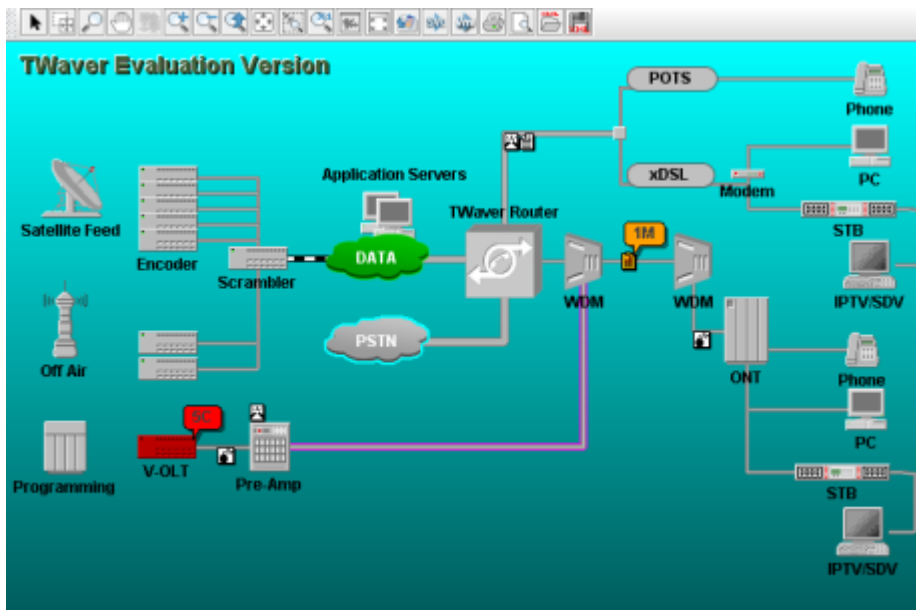
Swing Component	Description
DataBox	Provides data source for all graphical components
Network	Represent network topology or equipment chassis
Tree	Represent data hierarchy with a tree
Table	Represent data in two-dimension table
Element Table	Display network element data
Alarm Table	Display network alarm data
Alarm Overview	Display network alarm state information
Property Sheet	Display and edit properties for network data
List	A improved JList working with TWaver DataBox
Chart	Present data in form of information graphic

- [Network Component](#)
- [Tree Component](#)
- [Table Component](#)
- [TreeTable Component](#)
- [PropertySheet Component](#)
- [List Component](#)
- [Chart Component](#)

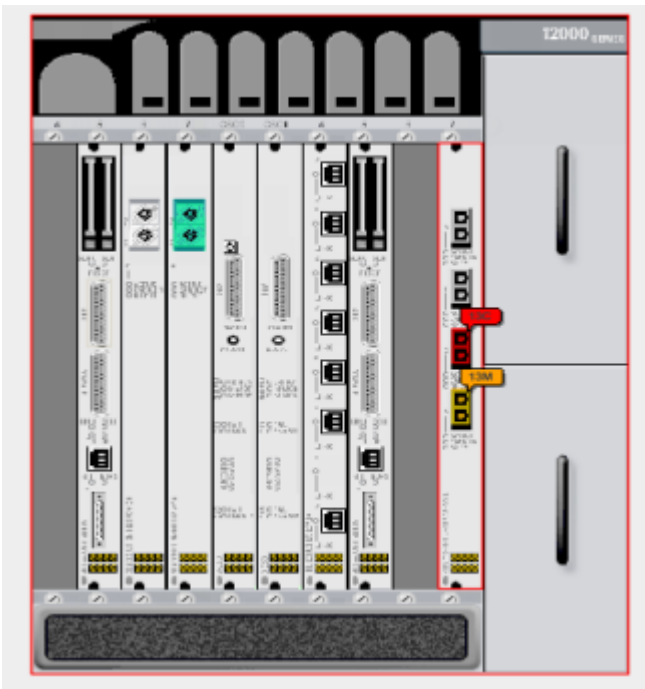
Network Component

Network component is the most important component of TWaver. The network component displays telecommunication networks. It can display the topology of a network or the equipment chassis panel. Network component support many features like selecting, navigation, editing and background maps. Network is derived from Swing JComponent class, is a pure Java light-weight component. The network component can display predefined managed objects of the type's network element, link, group, subnetwork, equipment elements. See the following chapters for more information of network component. Here are some pictures of network component.

Network Topology



Equipment Chassis



Network component use a DataBox instance as its data source. Developers can change the properties of network component like background map, toolbar etc. Network component also provide the facilities to customize the default behavior and look-and-feel.

Buttons of Network Toolbar

There are 3 type toolbar, as follows




```
/** define simple toolbar */
public static final String SIMPLE_TOOLBAR = "simple";

/** define default toolbar */
public static final String DEFAULT_TOOLBAR = "default";

/** define editor toolbar */
public static final String EDITOR_TOOLBAR = "editor";
```

The following is the DEFAULT_TOOLBAR type:



-  Selection Mode (default mode)
Click this button to reset default mode (selection mode)
-  Magnifier Mode
Click this button to view the network with a magnifier
-  Pan Mode
Click this button to pan the network view



Lazy Move Mode

Click this button to move elements with lazy rearrange



To Upper Sub Network

Click this button to navigate to upper subnetwork view



Zoom In

Click this button to zoom in network view



Zoom Out

Click this button to zoom out network view



Zoom Back

Click this button back to the last zoom of network view



Zoom To Overview

Click this button to zoom the network view to fit the network data contents



Zoom to Rectangle

Click this button to enlarge selected rectangle area to whole network view



Zoom Reset

Click this button to reset the zoom



Overview Window

Click this button to show/hide network overview window



Full Screen

Click this button (or press F11 button) to show/reset the network on full screen



Print

Click this button to print network



Print Preview

Click this button to open the print preview dialog



Save data to file

Click this button to save data to file



Open data file

Click this button to open data file

Tree Component

Tree component is derived from Java Swing JTree component. It provides a hierarchical view of the data contained in a DataBox. The tree component supports drag-and-drop, popup menu, alarm propagation and on-demand-load features. Another important feature is that tree component can display network topology data and equipment chassis data in the same view. See the following chapters for more information on tree component.

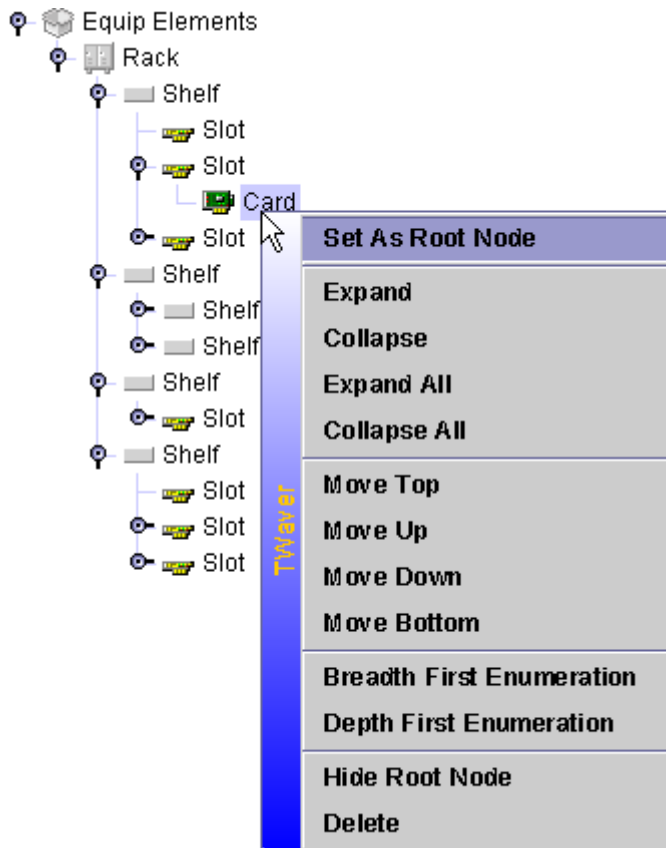















Table Component

Table component extends from Java Swing JTable component. It provides a lot of new features JTable missing. For example, show/hide columns, multi-column-sorting, data paging, predefined cell editors/renderers, row filter, etc. Several special extended table components are also be provided in TWaver for OSS development, such as AlarmTable, ElementTable, TreeTable. With these table components, you can create professional OSS GUIs easily. Here is a quick look at table component.

Page 1 of 1, Items 1 to 90 of 90					Size	All	Page	1	◀	▶	⏪	⏩
AlarmID	AlarmSeverity	Acked	ElementID	ProbableCause								
ALM_10	Critical		Router-1	Message out of sequence								
ALM_11	Minor	✓	Router-2	Remote alarm interface								
ALM_12	Critical		Router-1	Frequency hopping failure								
ALM_13	Critical	✓	Router-2	Synchronisation source mi								
ALM_14	Major		Router-1	Power supply failure								
ALM_15	Indeterminate	✓	Router-2	Clock synchronisation prob								
ALM_16	Critical		Router-1	Link failure								
ALM_17	Warning	✓	Router-2	Transcoder or rate adapter								
ALM_18	Indeterminate		Router-1	Invalid pointer								
ALM_19	Warning	✓	Router-2	Trunk card problem								
ALM_20	Major		Router-1	A-bis to TRX interface failur								
ALM_21	Critical	✓	Router-2	Call establishment error								

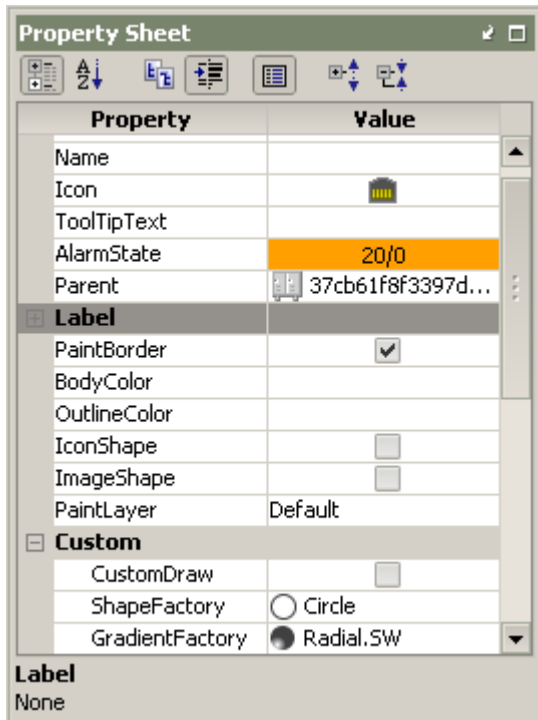
TreeTable Component

TreeTable component is a combination of a Tree and Table. This component is capable to expand and contract rows, as well as show multiple columns of data. The Swing package does not contain a TreeTable component, so TWaver TreeTable is a complement for Swing.

	Name	Icon	Visible
 Network Editor			
 EllipseGroup	EllipseGroup		<input checked="" type="checkbox"/>
 Node	Node		<input checked="" type="checkbox"/>
 Node	Node		<input checked="" type="checkbox"/>
 ParallelogramG	Parallelogr...		<input checked="" type="checkbox"/>
 Group	Group		<input checked="" type="checkbox"/>
 All Links	All Links		<input checked="" type="checkbox"/>

PropertySheet Component

PropertySheet component extends from the Swing JTable component. PropertySheet component displays and allows editing the managed object properties with two columns table. Developers can customize the cell renderer and cell editor. PropertySheet component looks like:



List Component

TWaver List component is an extension from Java Swing JList. The most important difference between TWaver List and Swing JList is that TWaver List provides the ability to work with TWaver DataBox. It also provides more features that JList component does not have, for example, row filter, visible filter, check box row support, checkable filter. Please see more information about TWaver List in following section. Here is a quick look at TWaver List component.

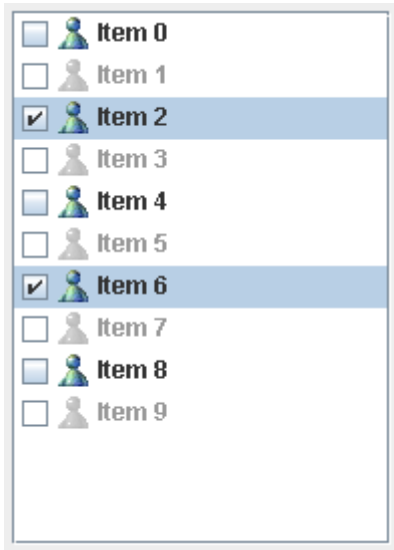
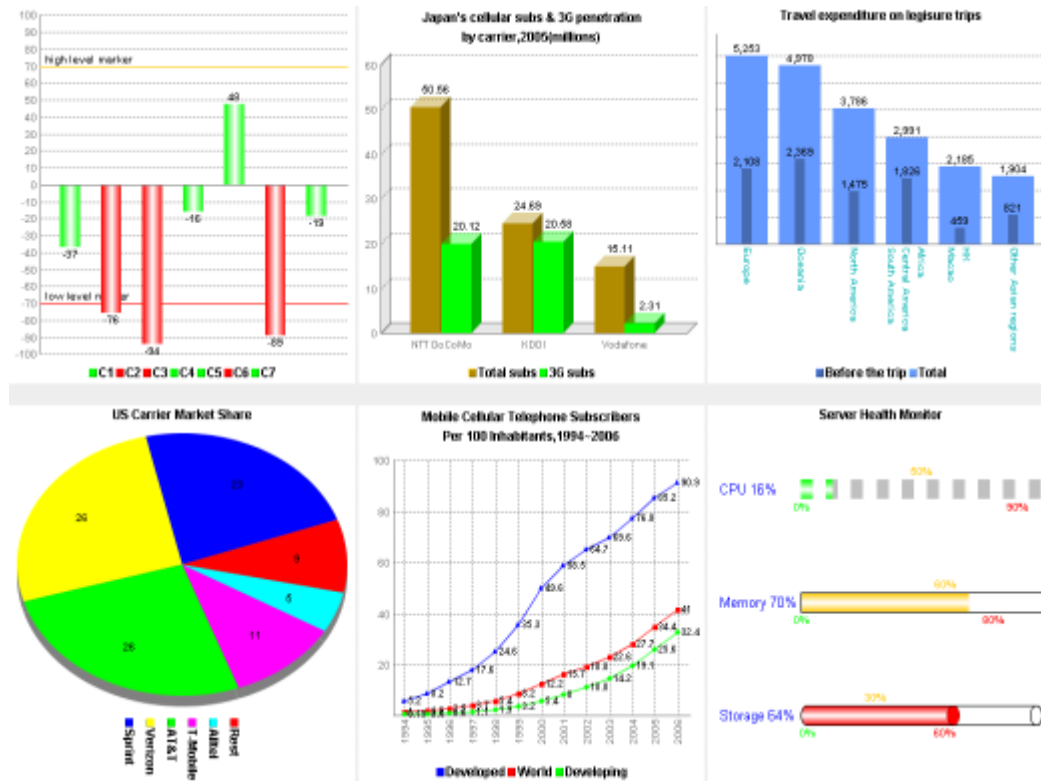


Chart Component

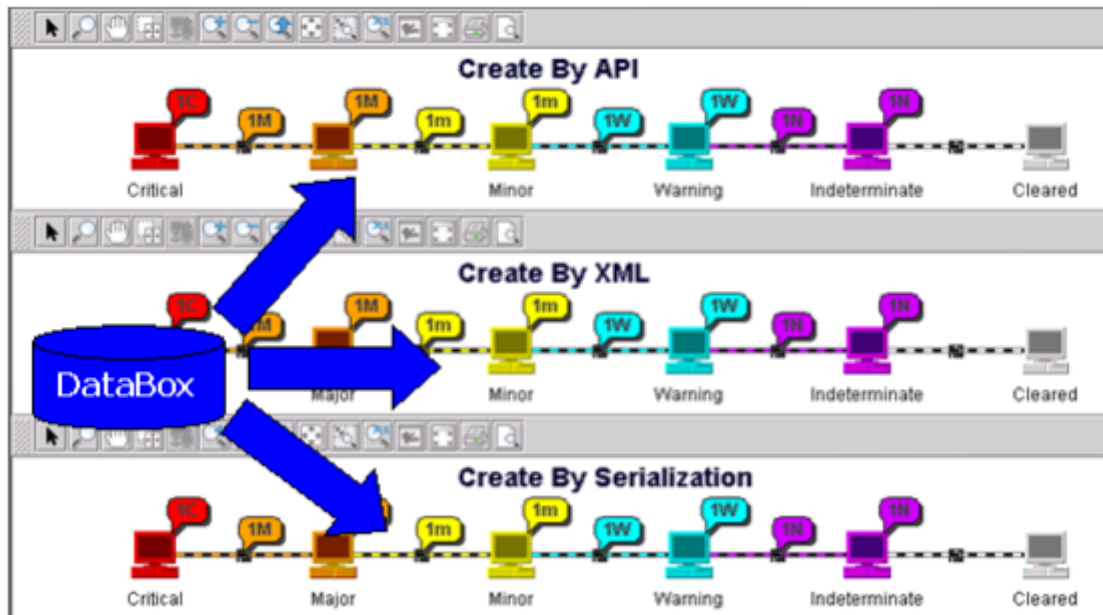
TWaver Chart Component works with DataBox and presents the data in form of information graphic, clearly and intuitionistically, TWaver Chart makes it easy for developers to display professional quality charts in their OSS applications without using extra third party chart product. TWaver support many output types, including image files and vector graphics file formats (including SVG and VML). Please see more information about TWaver Chart in following section. Here is a quick look at TWaver Chart component.



DataBox Overview

DataBox is the data container of all TWaver graphical components. It contains, manages, and monitors all business elements. As the model of MVC structure, it transforms data from back end into business elements that will then be rendered as graphic objects at the level of the graphic components.

DataBox support XML files or streams data. On the other side, the same data can be connected to multiple graphic components, allowing you to have different views of the same original data with a consistent appearance. One example is, when three network components share the same DataBox data, once the node moved from one network component, the other views will moved at the same time:



Managed Objects

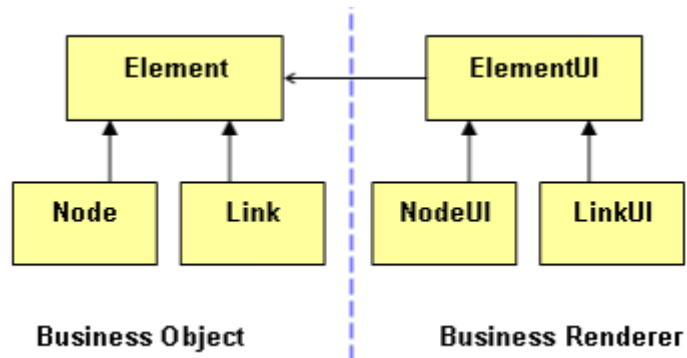
Managed Objects are the classes defined in TWaver and represent the business data in OSS, such as Node, Link etc.

- [Predefined Managed objects](#)
- [Structure of Managed objects](#)
- [Properties of Managed Objects](#)
- [Predefined Managed Objects Overview](#)

Predefined Managed objects

Managed Objects are defined to describe the application business model. TWaver provides many predefined managed objects like network element, link, subnetwork, rack, card etc. All managed objects are defined as the subclasses of the unique interface `Element`. `Element` instances are dynamic, which means that you can modify them and add/remove them at run time. Elements can also be defined directly in XML. The element definition is following the JavaBeans specifications, so you can regard elements as a JavaBeans object. It can fire events when its properties are changed.

TWaver elements are rendered by its UI class. That is, element are business model description, and element UI class is the painter or renderer of this element.

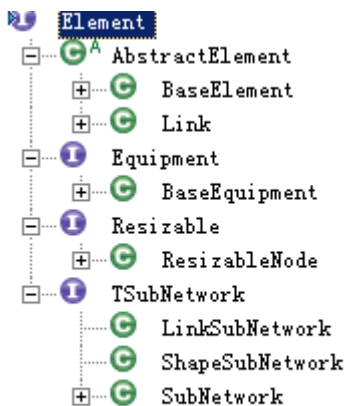


Structure of Managed objects

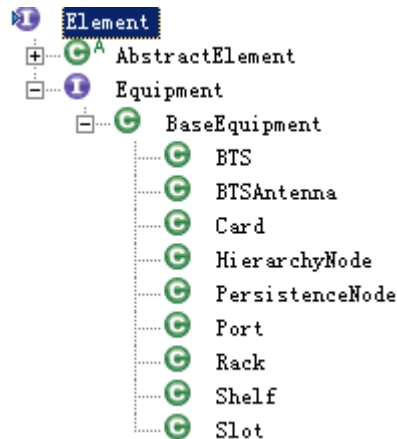
TWaver provides a set of predefined business elements to build OSS applications. They are:

- Node
- Link
- Group
- Dummy
- SubNetwork
- Chassis
- Equipment elements, includes Rack, Shelf, Slot, Card, Port etc.

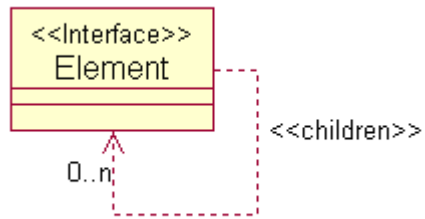
All these elements are derived from Element interface.



All equipment elements extend Node class and implements Equipment interface.



Each element is a container. That is, each element can contain one or more other element instances logically. TWaver doesn't specify the contained relations for these elements. For example, you can create a node to represent one person, and create five nodes to represent equipment. You can specify the parent of these equipment nodes is the person node to describe that they are managed by this person. The six nodes may display in the one layer view but in tree view it will display the hierarchy layer.



Properties of Managed Objects

Element interface defines many properties you can use, they are:

- **ID:** Identifier of the element. ID is a handler which can not be duplicated in one DataBox container. You can get the element reference from container by ID. ID is an immutable property once the element instance is created. You can create new element instances without specified identifier. When that, TWaver will create a default one by the IdentifierFactory automatically. Developers can specify a customized IdentifierFactory instance by invoking the static method of `TWaverUtil.setIdentifierFactory`.
- **Name:** The name of managed element. It will display as label on network component if it is set.
- **DisplayName:** The display name of managed element. It will display as label on network component if it is set and make it easy to remember the element.
- **Tool Tip Text:** The text to display in a tool tip. The text displays when the cursor lingers over the element.
- **UI Class ID:** the UI class name of element. UI class is the renderer of element responsible for drawing the element appearance.
- **SVGUIClassID:** A string that specifies the full qualified name of the UI class that is responsible for converting this element to SVG content.
- **VMLUIClassID:** A string that specifies the full qualified name of the UI class that is responsible for converting this element to VML content.
- **Selected:** Whether this element is now selected.
- **Visible:** Whether the element is visible.
- **Icon:** Element icon. The normal element icon is a 16*16 size small icon used to identify different element types.
- **Image:** The image of the element. It will display in network canvas.
- **Children:** Children elements. All elements contained by this element.
- **Parent:** Parent element.
- **Alarm State:** Contains all alarm state information.
- **Bound/Height/Width:** Size of this element.
- **Location:** location of this element. Link elements will return null.
- **User Object:** The object stored at this element by the user. It can be any object type that constitutes this element's user-specified data.
- **Client Properties:** Store a set of client properties with a specified key. Only properties added with `putClientProperty` will return a non-null value. Client properties are used to store and retrieve drawing information for the UI renderers.
- **User Property:** Store a set of user properties with a specified key. User property has the same function as client property, but because TWaver has already used client property to store some predefined properties of element, so user property is recommended to store other business data information to avoid conflict and confusion.
- **Business Object:** An element object can bind a business object to carry additional information. This looks like the same with user object bound by element, but TWaver treats business object as a special property of element, `twaver.table.TPropertySheet` and `twaver.table.TElementTable` can display information of business object's internal properties.

Predefined Managed Objects Overview

In this section we introduce you to several major managed objects briefly. Elements can be divided into node-like elements and link-like elements. Node-like elements have network topology node elements and equipment chassis elements. They are:

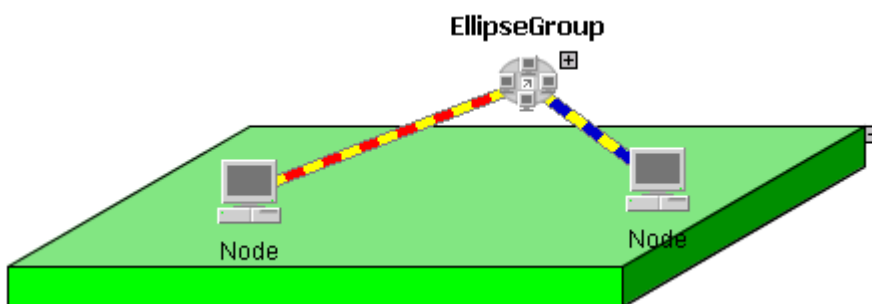
Node

Node element. Node can used to represent one network element node in a telecom application, like one router, computer or building. Node can specified an image to represent different staff:



Group

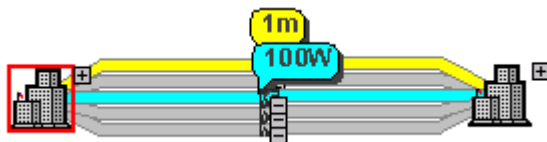
Group is used to represent a set of network resources grouped logically or geographically. Group can expand or closed. When it closed, it will be displayed like a normal node element, and when expand, it will be displayed as a region over all children elements:



Link

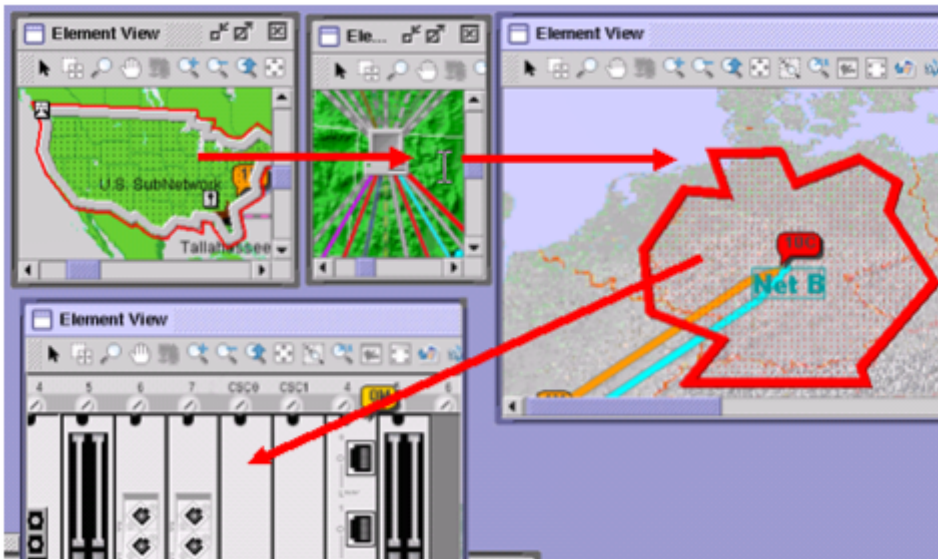
Link element is used to display the transmission elements making up the network lines. Links can also carry decorations, like alarms and states information:

!worddav03edcabdccc2d1b986501f9fbda63f6e.png!Link support multi-link bundled, it can expand or close by double-click mouse:



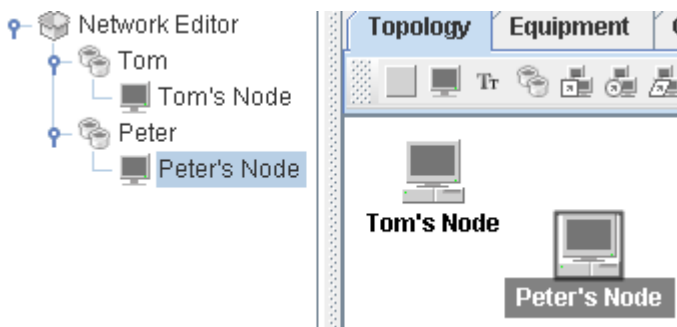
SubNetwork

SubNetwork is a special element which can display all children in a standalone layer. Sub network can used to represent a physical or logical resource container in your applications. In top level view, sub network will be displayed like a normal node. You can double click sub network element to enter the sub network to display all internal elements. When your application have thousands nodes and equipments, you can reorganize your network with multy layer by sub network elements. User can drill-down the network to make network structure easier to understand:



Dummy

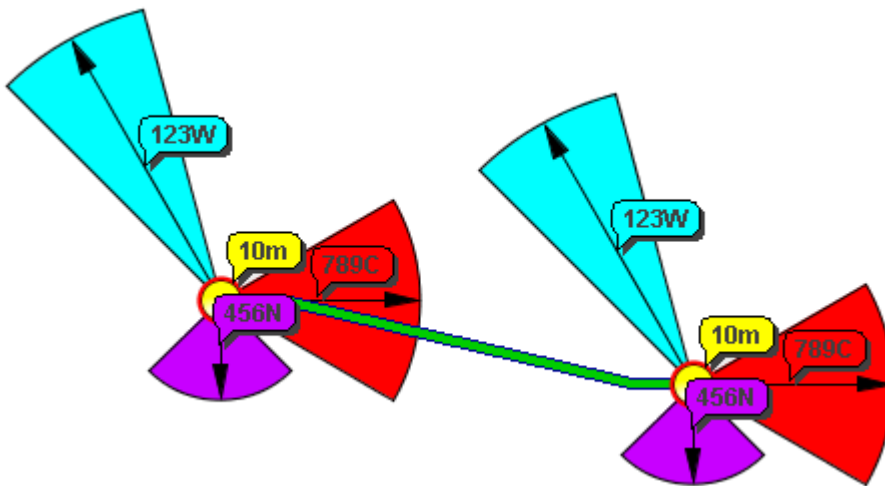
Dummy element is an abstract container used to reorganize your elements relationship and hierarchy. Dummy element is invisible container in network component, but it will be displayed in tree component. Dummy element can be used to represent some logical contained relationship. For example, you can create dummy element to represent the user of your application, then put some nodes into this person:



Dummy element has no any effect to network component painting.

BTS & BTSAntenna

Base Transceiver Stations (BTS) are base stations composed of antennas that relay (receive and transmit) radio messages within cells of a cellular phone system. Each antenna has an orientation and a beam width that are graphically represented. Each antenna can have its own state and graphical characteristics.



Alarm Support





TWaver provides a comprehensive and user-friendly alarm graphical environment for TWaver components. This section provides a detail description of visual aspect of alarms.

- [Alarm Representation in TWaver](#)
- [Using Alarm Severities](#)

Alarm Representation in TWaver

Alarms can be either new or acknowledged. In TWaver, the term native alarm refers to both new and acknowledged alarms occurred on the element object, propagative alarm refers to the highest native alarm for all contained elements, overall alarm refers to the highest alarm of native alarm and propagative alarm. They are represented in the following manner:

- New alarms: The new alarms of a telecom object will show the number and highest severity (color and letter) in the alarm balloon.
- Acknowledge alarms: TWaver compare the highest new alarm severity and highest acknowledged alarm severity. The more severe alarm severity color will be displayed on element body region.
- Alarm propagation: All contained elements' highest (no matter new alarms or acknowledged alarms) alarm severity color will display on the border of this element.

	One new critical alarm
	Ten new critical alarms, plus one or more less severe new alarms
	One new major alarm, plus one or more acknowledged critical alarms
	One new warning alarm, plus one or more acknowledged minor alarms, plus propagate critical alarm

Using Alarm Severities

TWaver provides a standard range of severity levels. The levels and their associated color and initial are:

Severity	Letter	Value	Color
CRITICAL	C	500	Red
MAJOR	M	400	Orange
MINOR	m	300	Yellow
WARNING	W	200	Cyan
INDETERMINATE	N	100	Purple
CLEARED	R	0	Green

Each alarm severity has one int value as its severity value. It dedicates this alarm severity severe level. By default, the value is bigger, it's more severe. Developer also can define new alarm severity comparator. Please see [Customizing AlarmSeverity Comparator](#) for more details.

Development Overview

- [Install Java and TWaver](#)
- [TWaver Configuration](#)
- [Creating Data by XML or API](#)

Install Java and TWaver

TWaver doesn't need any special installation. Please make sure you've install JDK/JRE property. You can find more information at <http://java.sun.com/> about JDK/JRE installation. TWaver needs JDK/JRE 1.4 or above.

From TWaver 1.4.1, TWaver license should be validated before start up your application. Please make sure you have received a license file from TWaver sales representative. The license is a file which contains TWaver license type, additional information and fingerprint data.

Active your TWaver license with following steps:

- Put the license file to a location where can be accessed with a proper Java URL. Following URLs are all legal license location:
 - - "file:/c:/myapp/data/license.dat"
 - "http://www.mycompany.com/application/license.dat"
 - "/com/myapp/oss/data/license.dat"
- Validate the license with following code when start your applications:

```
TWaverUtil.validateLicense ("file:/c:/myapp/data/license.dat");
```



A proper exception will be thrown if the license validation does not pass. The possible cause of the exception are:

1. The license type defined in the license file does not match.
2. If any content of the license file has been changed or damaged, validation will failed.

TWaver Configuration

TWaver is designed to present much different kind of network resources, so there are a large number of configuration parameters. Fortunately, most of them have sensible default values and TWaver is distributed with an example resource/TWaver.xml file that shows the various options. Just put the sample file in your classpath, customize it and enable it by [TWaver Resources Redirection](#).

With TWaver.xml file, you can:

- Change the default rendering values of TWaver UIManager
- Define new Alarm Severity
- Define element properties group for PropertySheet component
- Define buttons and toolbars for network components
- Define element icon and image
- Define table renderers and editors

Each of above will be explained in detail in the following sections.

Creating Data by XML or API

You can create data by using XML or API in TWaver.

- [Creating Data by API](#)
- [Creating Data by XML](#)
- [Comparing XML and API](#)

Creating Data by API

Normally you can create network data by APIs. In this way, you'll control everything in your codes.

As the MVC structure of TWaver, the data and view are designed separately. All network data are contained by DataBox, all components connect to DataBox to provide different views:

```
//Create a new DataBox instance.
TDataBox box = new TDataBox();
//Create a new Network component.
TNetwork network1 = new TNetwork(box);
//or creates then set databox.
TNetwork network2 = new TNetwork();
network2.setDataBox(box);
```

Once the DataBox and components are created and connected, you can create any managed objects you want and add them into the DataBox container. It will be displayed by all component views automatically:

```
//Create an Element and add into databox.
Element node = new Node("Node A");
node.setLocation(100,100);
box.addElement(node);
```

In this example, both network components will display the same Node instance in the location of (100,100). You can change the data properties directly if you want:

```
//Change the element's location.
node.setName("I'm Node B");
node.setLocation(200,200);
```

Now the node will move to location (200,200) automatically in both network components.

Creating Data by XML

You can create network data by XML. In this way, you'll define all managed objects in XML file or XML stream and load it to DataBox.

The XML can be composed by user, or generated dynamically by back end systems. DataBox will parse the XML data stream to transform it into TWaver managed objects:

```
//Parse a XML file  
box.parse("elements.xml");
```

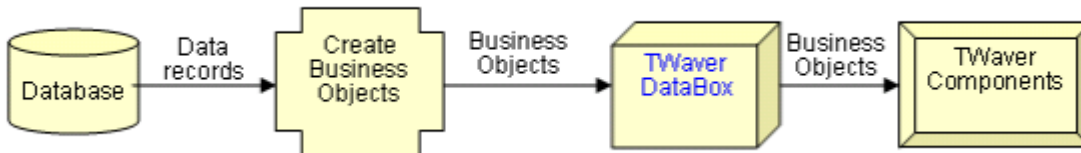
The elements.xml file can be defined as follows:

```
<?xml version="1.0" encoding="UTF-8"?>  
<java version="1.4.1" class="java.beans.XMLDecoder">  
<object id="Node1" class="twaver.Node">  
  <void property="location">  
    <object class="java.awt.Point">  
      <int>100</int>  
      <int>100</int>  
    </object>  
  </void>  
  <void property="name">  
    <string>Node A</string>  
  </void>  
</object>  
</java>
```

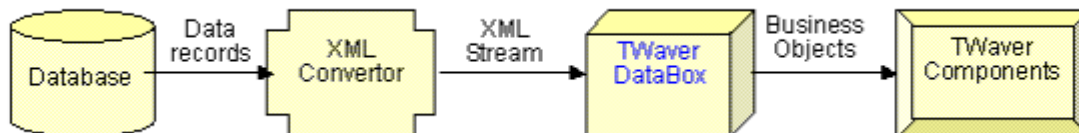
By XML stream mode, your code can more simple and brief. You don't need to create numerous elements in your codes. All elements can be composed in advance and loaded in run time. Moreover, TWaver provides a network data editor application to reduce the endeavor of composing an XML file.

Comparing XML and API

In most cases you can create data by API. By this way, developers can control the data easily and flexibly. You can add, remove or change data freely. The disadvantage is, you have to write and maintain more code in your application. Besides, if your application needs to get data from other non-Java systems, you may have to use another cross-platform interface like CORBA.



Creating data by XML stream can build cross-platform applications, reduce the coding effort. It can exchange data from other non-Java systems by a standard XML format data stream:



You can create data by XML stream when the network data is seldom changed, or generated by other remote non-Java systems.

More knowledge about CORBA, please visit website <http://www.corba.org/>.

Basic Programming

In this chapter we use a tutorial to show you how to create some graphic components, configure a network component, use custom managed objects, update objects, create a dialog to show chassis details, add some interactions, and handle selection. In short, it steps you through the typical tasks that you want to handle with TWaver.

- [What All Developers Should Know](#)
- [Executing the Example Application](#)
- [Creating a Basic Network Component](#)
- [Creating a Basic Tree Component](#)
- [Showing Equipment Chassis](#)
- [Element Selection](#)
- [Customizing Interactions](#)
- [Adding Alarms](#)
- [Adding Decorated Icons](#)
- [Adding Special Effect](#)

What All Developers Should Know

TWaver is aimed at an audience of Java developers. Therefore, in this manual we make the assumption that you can write Java code and that you are familiar with your Java environment so as to manipulate files and directories, use a text editor, and compile and run Java programs. We also assume that you are familiar with the PC or UNIX environment in which you are going to use TWaver, including its specific window system.

Executing the Example Application

First we need to create a frame, and then create and put some graphic components into the frame. Now we begin with the code for create the main frame.

We first create a frame, and specify the frame content pane to contain what is to be displayed. Use the "doSample" function to create and initialize all TWaver components:

```
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import twaver.*;
import twaver.network.*;
import twaver.network.ui.*;
import twaver.table.*;
import twaver.tree.*;

public class Tutorial extends JFrame {
    private TDataBox box = new TDataBox("Simple Data Box");
    private TNetwork network;
    private TTree tree;
    private JPanel networkPane = new JPanel(new BorderLayout());
    private JPanel treePane = new JPanel(new BorderLayout());
    private JSplitPane split =
        new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
            treePane,
            networkPane);

    public Tutorial() {
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        getContentPane().add(split, BorderLayout.CENTER);
        split.setDividerLocation(100);

        doSample();
    }

    public static void main(String[] args) {
        Tutorial frame = new Tutorial();
        frame.setSize(500, 300);
        frame.setTitle("TWaver Tutorial");
        TWaverUtil.centerWindow(frame);
        frame.setVisible(true);
    }

    private void doSample() {
        try {
            step1();
            step2();
            step3();
            step4();
            step5();
            step6();
            step7();
            step8();
            step9();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    private void step1() {}
}
```

```
private void step2() {}

private void step3() {}

private void step4() {}

private void step5() {}

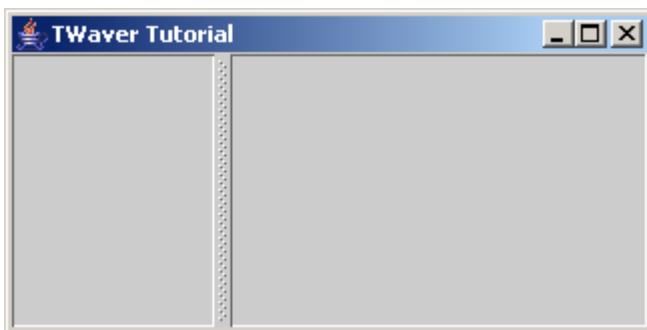
private void step6() {}

private void step7() {}

private void step8() {}

private void step9() {}
}
```

Run this class, it would display an empty frame, which looks like this:



Creating a Basic Network Component

This section shows you how to create a basic network component to model and represent a telecommunication network. This section of the code is referred to as step 1.

To allow the network component to function, you must create a TDataBox object to hold the managed objects which represent the telecommunication elements. Then you can create a network component and connect it to the DataBox, add the network components to the frame. Next you can create some elements like nodes, links and add it to the DataBox. Finally, the network will display in the window.

This section of the code is referred to as Step 1.

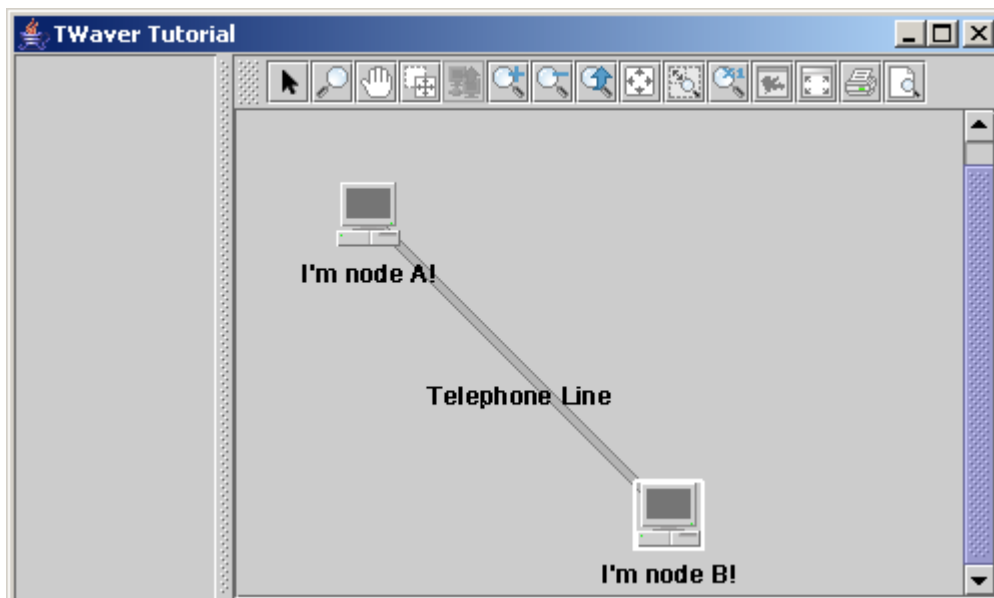
```
private void step1() {
    network = new TNetwork(box);
    networkPane.add(network, BorderLayout.CENTER);

    Node nodeA = newNode("A");
    nodeA.setName("I'm node A!");
    nodeA.setLocation(50, 50);
    box.addElement(nodeA);

    Node nodeB = newNode("B");
    nodeB.setName("I'm node B!");
    nodeB.setLocation(200, 200);
    box.addElement(nodeB);

    Link link = new Link("link", nodeA, nodeB);
    link.setName("Telephone Line");
    box.addElement(link);
}
```

Run this program:



Creating a Basic Tree Component

This section shows you how to create a tree component to show the containment hierarchy.

To allow the tree component to function, you must create a tree component and connect it to a DataBox. This example uses the same DataBox that was created for the network component to show the same data in the tree and the network. Finally, we add the tree component to the frame.

This section of the code is referred to as step 2.

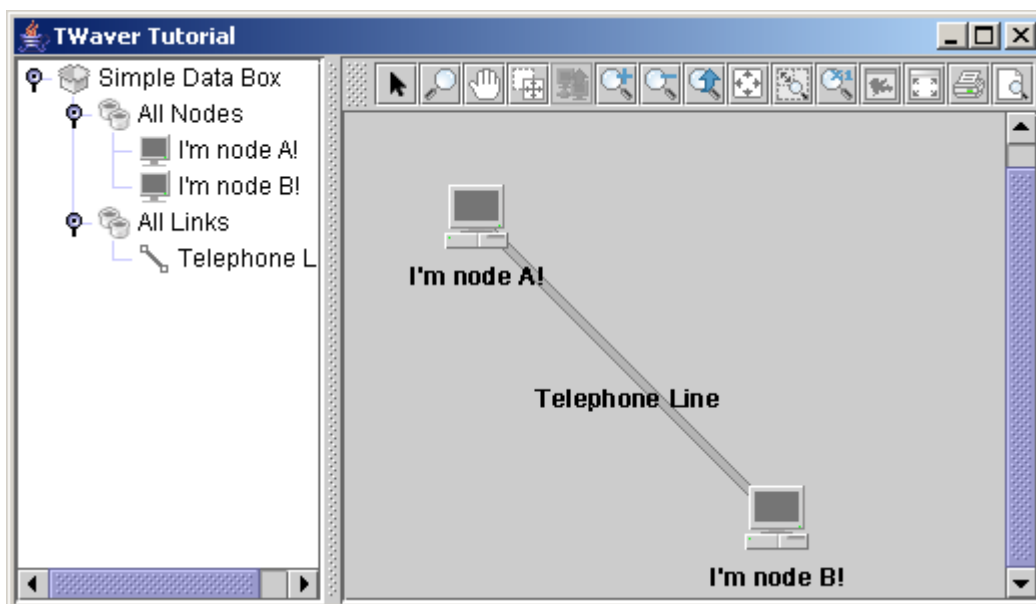
```
private void step2() {
    Dummy nodeDummy = new Dummy("node dummy");
    nodeDummy.setName("All Nodes");
    nodeDummy.addChild(box.getElementByID("A"));
    nodeDummy.addChild(box.getElementByID("B"));
    box.addElement(nodeDummy);

    Dummy linkDummy = new Dummy("link dummy");
    linkDummy.setName("All Links");
    linkDummy.addChild(box.getElementByID("link"));
    box.addElement(linkDummy);

    tree = new TTree(box);

    JScrollPane scroll = new JScrollPane(tree);
    treePane.add(scroll, BorderLayout.CENTER);
}
```

Run the program:



Showing Equipment Chassis

In this section we will show you how to display telecom equipment chassis detail information. We use both the API way and XML way for the demonstration.

- [Creating Chassis by API](#)
- [Creating Chassis by XML](#)

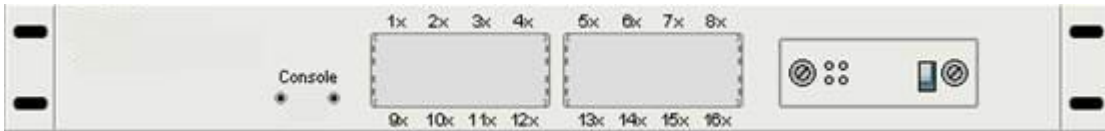
Creating Chassis by API

To display an equipment detail by API, we need:

- Create a new chassis and add into node A
- Create a rack and add into this chassis
- Create some ports and add them into this rack

Before coding, we prepare three images to represent the rack and ports as follows:

The image of rack:



The image of port1:



The image of port2:



This section of the code is referred to as step 3:

```
private void step3() {
    Node node = (Node) box.getElementById("A");
    Chassis chassis = new Chassis("Chassis A");
    node.addChild(chassis);
    box.addElement(chassis);

    //1.add rack to chassis.
    Rack rack = new Rack("Rack A");
    rack.setName("Rack");
    rack.setLocation(50, 50);
    rack.setImage("/demo/resource/tutorial/rack.png");
    chassis.addChild(rack);
    box.addElement(rack);

    //2.add ports to rack.
    String imgPort1 = "/demo/resource/tutorial/port1.png";
    String imgPort2 = "/demo/resource/tutorial/port2.png";
    for (int module = 0; module < 4; module++) {
        Dummy dummy = new Dummy("PortDummy" + module);
        dummy.setName("module" + module);
        rack.addChild(dummy);
        box.addElement(dummy);

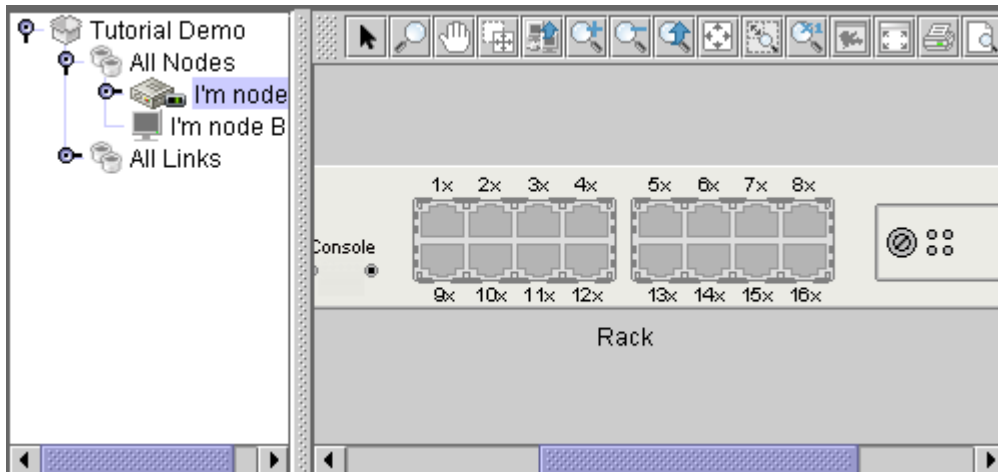
        for (int index = 0; index < 4; index++) {
            Port port = new Port(module + "." + index);
            int x, y;
            if (module % 2 == 0) {
                x = 210 + index * 24;
            } else {
                x = 319 + index * 24;
            }
            if (module < 2) {
                y = 16;
                port.setImage(imgPort1);
            } else {
                y = 37;
                port.setImage(imgPort2);
            }
        }
    }
}
```

```

    }
    x += rack.getLocation().x;
    y += rack.getLocation().y;
    port.setLocation(newPoint(x, y));
    dummy.addChild(port);
    box.addElement(port);
  }
}
}

```

Run the program:



Creating Chassis by XML

Creating an equipment chassis by API is boring sometimes. You have to change and rebuild your code if the equipment structure or appearance was changed. Now we create the chassis by XML file. The XML file contains all chassis elements definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.1" class="java.beans.XMLDecoder">

<!-- rack -->
<object class="twaver.Rack" id="rack">
<void property="Image">
<string>/image/twaverRack.png</string></void>
<void property="location">
<object class="java.awt.Point">
<int>100</int>
<int>100</int>
</object>
</void>
</object>

<!-- ports -->

<!-- console port -->
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverConsolePort.png</string>
</void>
<void property="parent">
<object idref="rack"/>
</void>
<void property="location">
<object class="java.awt.Point">
<int>259</int><int>145</int>
</object>
</void>
</object>

<!-- fiber port -->
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverFiberPort.png</string>
</void>
<void property="parent">
<object idref="rack"/>
</void>
<void property="location">
<object class="java.awt.Point">
<int>582</int><int>132</int>
</object>
</void>
</object>

<!-- module 1 -->
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverPort1.png</string>
</void>
<void property="parent">
<object idref="rack"/>
</void>
<void property="location">
```

```

<object class="java.awt.Point">
<int>310</int> <int>116</int>
</object>
</void>
</object>

<object class="twaver.Port">
<void property="Image">
<string>/image/twaverPort1.png</string>
</void>
<void property="parent">
<object idref="rack"/>
</void>
<void property="location">
<object class="java.awt.Point">
<int>334</int> <int>116</int>
</object>
</void>
</object>
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverPort1.png</string>
</void>
<void property="parent">
<object idref="rack"/>
</void>
<void property="location">
<object class="java.awt.Point">
<int>358</int> <int>116</int>
</object>
</void>
</object>
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverPort1.png</string>
</void>
<void property="parent">
<object idref="rack"/>
</void>
<void property="location">
<object class="java.awt.Point">
<int>382</int> <int>116</int>
</object>
</void>
</object>

<!-- module 2-->
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverPort1.png</string>
</void>
<void property="parent">
<object idref="rack"/>
</void>
<void property="location">
<object class="java.awt.Point">
<int>419</int> <int>116</int>
</object>
</void>
</object>
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverPort1.png</string>
</void>
<void property="parent">
<object idref="rack"/>

```

```

</void>
<void property="location">
<object class="java.awt.Point">
<int>443</int> <int>116</int>
</object>
</void>
</object>
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverPort1.png</string>
</void>
<void property="parent">
<object idref="rack"/>
</void>
<void property="location">
<object class="java.awt.Point">
<int>467</int> <int>116</int>
</object>
</void>
</object>
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverPort1.png</string>
</void>
<void property="parent">
<object idref="rack"/> </void>
<void property="location">
<object class="java.awt.Point">
<int>491</int> <int>116</int>
</object>
</void>
</object>

<!-- module 3-->
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverPort2.png</string>
</void>
<void property="parent">
<object idref="rack"/>
</void>
<void property="location">
<object class="java.awt.Point">
<int>310</int> <int>137</int>
</object>
</void>
</object>
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverPort2.png</string>
</void>
<void property="parent">
<object idref="rack"/>
</void>
<void property="location">
<object class="java.awt.Point">
<int>334</int> <int>137</int>
</object>
</void>
</object>
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverPort2.png</string>
</void>
<void property="parent">
<object idref="rack"/>

```

```

</void>
<void property="location">
<object class="java.awt.Point">
<int>358</int> <int>137</int>
</object>
</void>
</object>
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverPort2.png</string>
</void>
<void property="parent">
<object idref="rack"/>
</void>
<void property="location">
<object class="java.awt.Point">
<int>382</int> <int>137</int>
</object>
</void>
</object>

<!-- module 4-->
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverPort2.png</string>
</void>
<void property="parent">
<object idref="rack"/>
</void>
<void property="location">
<object class="java.awt.Point">
<int>419</int> <int>137</int>
</object>
</void>
</object>
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverPort2.png</string>
</void>
<void property="parent">
<object idref="rack"/>
</void>
<void property="location">
<object class="java.awt.Point">
<int>443</int> <int>137</int>
</object>
</void>
</object>
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverPort2.png</string>
</void>
<void property="parent">
<object idref="rack"/>
</void>
<void property="location">
<object class="java.awt.Point">
<int>467</int> <int>137</int>
</object>
</void>
</object>
<object class="twaver.Port">
<void property="Image">
<string>/image/twaverPort2.png</string>
</void>
<void property="parent">

```

```
<object idref="rack"/>
</void>
<void property="location">
<object class="java.awt.Point">
<int>491</int><int>137</int>
</object>
</void>
</object>
</java>
```

Then we save the current contents into a XML file named `equipment1_template.xml`, and load it with following code:

```
private void step3() {
    //add chassis to node A.
    Node node = (Node) box.getElementByID("A");
    Chassis chassis = new Chassis("Chassis A");
    chassis.setDataSource("equipment1_template.xml");
    node.addChild(chassis);
    box.addElement(chassis);
}
```

Now run this application, you can see the application has the same behavior. The only difference is that equipment chassis are loaded only once dynamically from hard disk when you double click the equipment node the first time.

In this application, the benefits of using XML are:

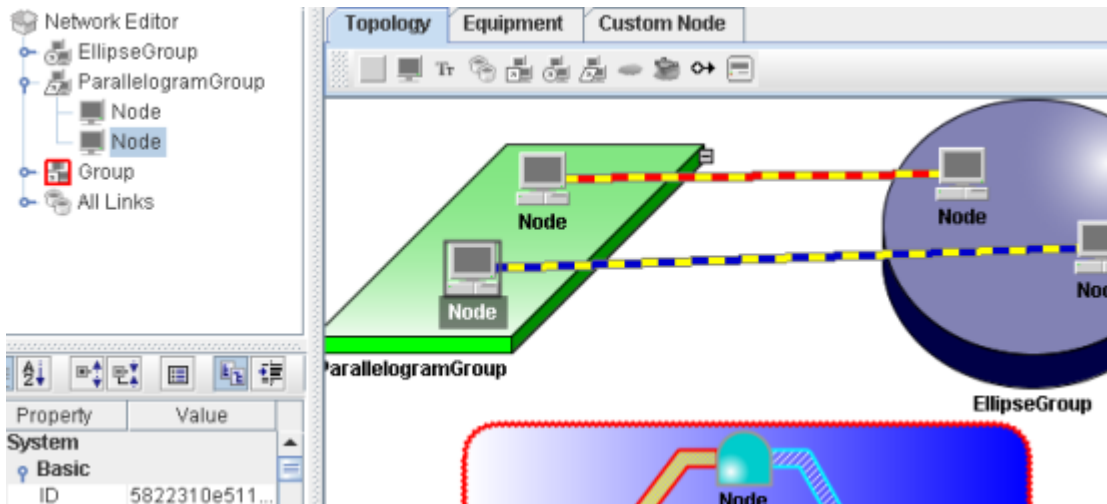
- Speed up application launch: If your application has thousands of different equipment with chassis data, it will be very slow to load all the data by API when application is launching. Using XML data we save this initializing time.
- Simplify your codes: Put all chassis data into XML data instead of in code. It'll more easy to maintain and manage.
- Reducing memory consumption: All equipment data are created on demand. That will reduce the amount of object creation.
- Sharing template data: The XML can be regards as the equipment template and shared by multiple applications.

Element Selection

- [Using Selection State](#)
- [Using Selection Model](#)
- [Using Selection Listener](#)
- [Using Selection Interaction](#)

Using Selection State

TWaver element can be selected or deselected. Each element objects maintenance the current selection state. Developers can visit the state by `isSelected/setSelected` methods. Selection state of element will affect its appearance on different views. For network component, all selected elements have a white selection border. For tree component, selected elements have selection background color.



Using Selection Model

As a property of DataBox, selection model is an element container which contains all selected elements. It also provides APIs to visit, iterate, add or clear the elements.

Using Selection Listener

You can create selection listener to detect selection model change. DataBox selection events occur when the selection is either changing or has just changed. To detect DataBox selection events, you can just register a listener on the selection model object. Here's the code that sets up the selection model and adds a listener to it:

```
DataBoxSelectionListener l =new DataBoxSelectionAdapter(){
    public void selectionChanged(DataBoxSelectionEvent e) {
        System.out.println("selection changed.");
    };
};
TDataBox box=new TDataBox();
box.getSelectionModel().addDataBoxSelectionListener (l);
//add more code here...
```

Using Selection Interaction

TWaver Network component provides the build in interaction for data selection. You can manipulate data selection by API, mouse or keyboard.

Device	Operation	Result
Mouse	Click an element	Reverse selection state of the element
	Drag a rectangle on Network view	Select all elements located in the rectangle
	Drag a rectangle from right-bottom to left-top	Select all elements intersect the rectangle
Keyboard & Mouse	Press "Ctrl" key and click an element	Add or remove the element to/from the selection model
	Press "Ctrl" key and drag rectangle	Add or remove the rectangle-related elements to/from the selection model

Customizing Interactions

The build-in interactions may still not meet your needs on network interaction. Then you can learn how to customize more actions on network.

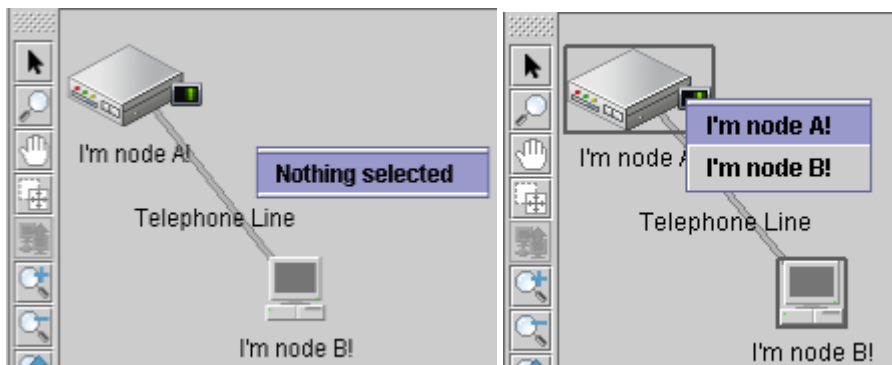
- [Adding Popup Menu Generator](#)
- [Adding Mouse & Keyboard Action](#)
- [Handling Selection](#)

Adding Popup Menu Generator

In this section we show you how to create a popup menu. TWaver create popup menu by a popup menu generator. More detailed information about popup menu generator please sees following sections.

This section of the code is referred to as step 4:

```
private void step4() {
//Create a popup menu generator
PopupMenuGenerator popupMenuGenerator = new PopupMenuGenerator() {
/**
 * Add the identifier of each of the selected objects to the menu.
 * In this example, the items added to the menu do nothing.
 * In a real application, you would probably associate an
 * implementation of the Swing Action interface with each menu item.
 */
public JPopupMenu generate(TView tview, MouseEvent mouseEvent){
//Create an empty pop-up menu.
JPopupMenu popMenu = new JPopupMenu();
JMenuItem item;
//If the selectedObjects collection is empty, no objects are selected.
if (tview.getDataBox().getSelectionModel().isEmpty()) {
popMenu.add("Nothing selected");
} else {
//Access the selected objects from the selection model.
Iterator it = tview.getDataBox().getSelectionModel().selection();
while (it.hasNext()) {
Element element = (Element) it.next();
popMenu.add(element.getName());
}
}
//If menu is empty, return null.
if (popMenu.getComponentCount() == 0) {
return null;
} else {
return popMenu;
}
};
//Set the pop-up menu generator for network components
network.setPopupMenuGenerator(popupMenuGenerator);
}
```



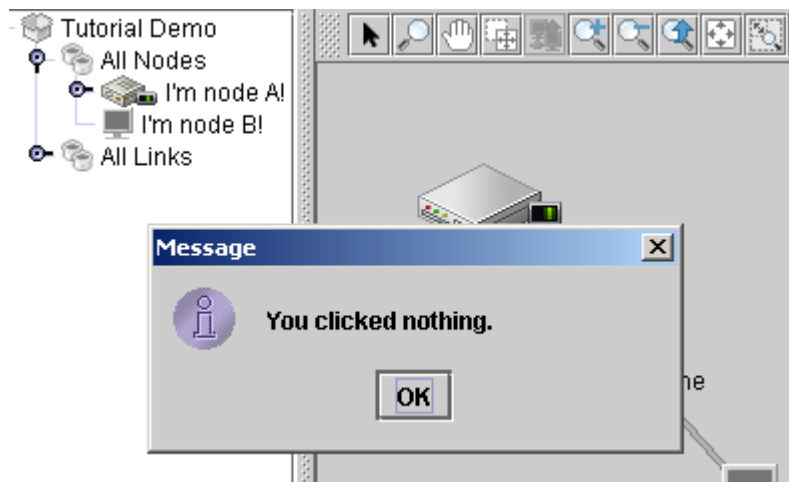
Adding Mouse & Keyboard Action

Network component is based on Java and Swing. Like any other Swing components, network component can provide the ability to listen for user input actions easily, including mouse events and keyboard events. To respond the events, you can get the canvas area of the network component. Then add mouse or keyboard event listener like Swing component. In order to demonstrate this, we show a message dialog when user double-click mouse left button.

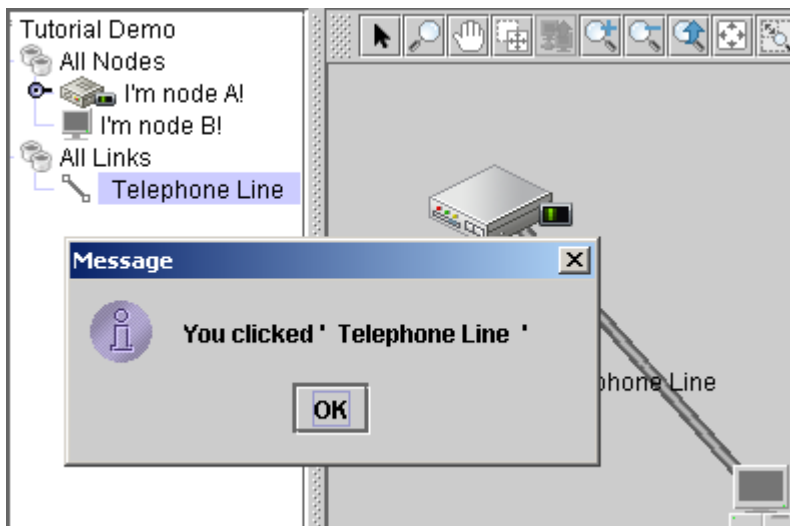
This section of the code is referred to as step 5.

```
private void step5() {
    network.getCanvas().addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e) {
            if (e.getClickCount() == 2) {
                //get the element the mouse clicked.
                Element element = network.getElementPhysicalAt(e.getPoint());
                String message;
                if (element == null) {
                    message = "You clicked nothing.";
                } else {
                    message = "You clicked " + element.getName() + "";
                }
                JOptionPane.showMessageDialog(network, message);
            }
        }
    });
}
```

Run this program



Double click blank area



Double click one element

From TWaver1.3.0, you can use new methods to monitoring mouse double click on network:

- `TNetwork.addElementDoubleClickedActionListener`
- `TNetwork.addBackgroundDoubleClickedActionListener`
- `TNetwork.addElementClickedActionListener`

More methods will be added to in the following versions.

Handling Selection

Handling Selection

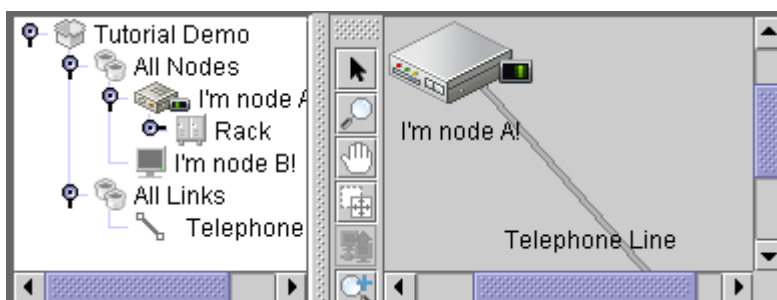
This section shows you how to:

- Control the selection.
- Listen for selection changes.
- Pan the network so that the selected object becomes visible.

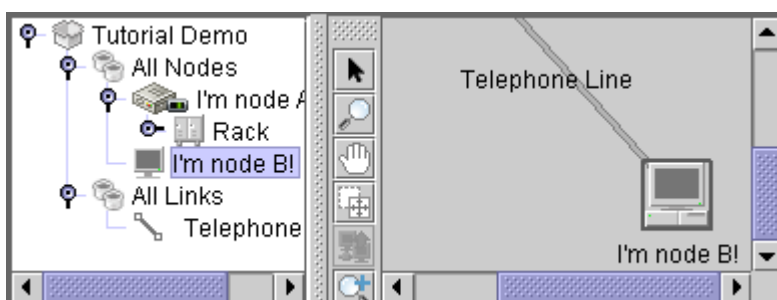
The TWaver components that have the same DataBox share the same selection model. You can get the selection model by DataBox object, which allows you to access the selection as managed objects. If you are interested in the selection changing, you can create a new selection listener to listen the managed object selection events. Selection event will send to the listener when the content of the selection model changes. To demonstrate this, we create a selection listener. When you select one element from tree component, the element will be selected on network component as well. Network component also move its view to ensure the selected element is visible.

This section of the code is referred to as Step 6.

```
private void step6() {
    //create a selection listener.
    DataBoxSelectionListener listener = new DataBoxSelectionListener() {
        public void selectionChanged(DataBoxSelectionEvent e) {
            //get the last selected element and make it visible.
            Element element = e.getDataBox().lastElement();
            if (element != null) {
                network.ensureVisible(element);
            }
        }
    };
    box.getSelectionModel().addDataBoxSelectionListener(listener);
}
```



None element selected



Select one element from tree component

Adding Alarms

This section introduces you how to add alarms on managed objects. TWaver provides a comprehensive, user-friendly interface designed to illustrate changes in network telecom equipment states and alarms. A number of graphical properties have been developed to notify the telecommunication network operator that alarms are present. When a new alarm is detected on an element, visual cues are added to its graphical representation.

To add alarm on the managed objects, you should get the alarm state object, and set the alarm count for each alarm severity.

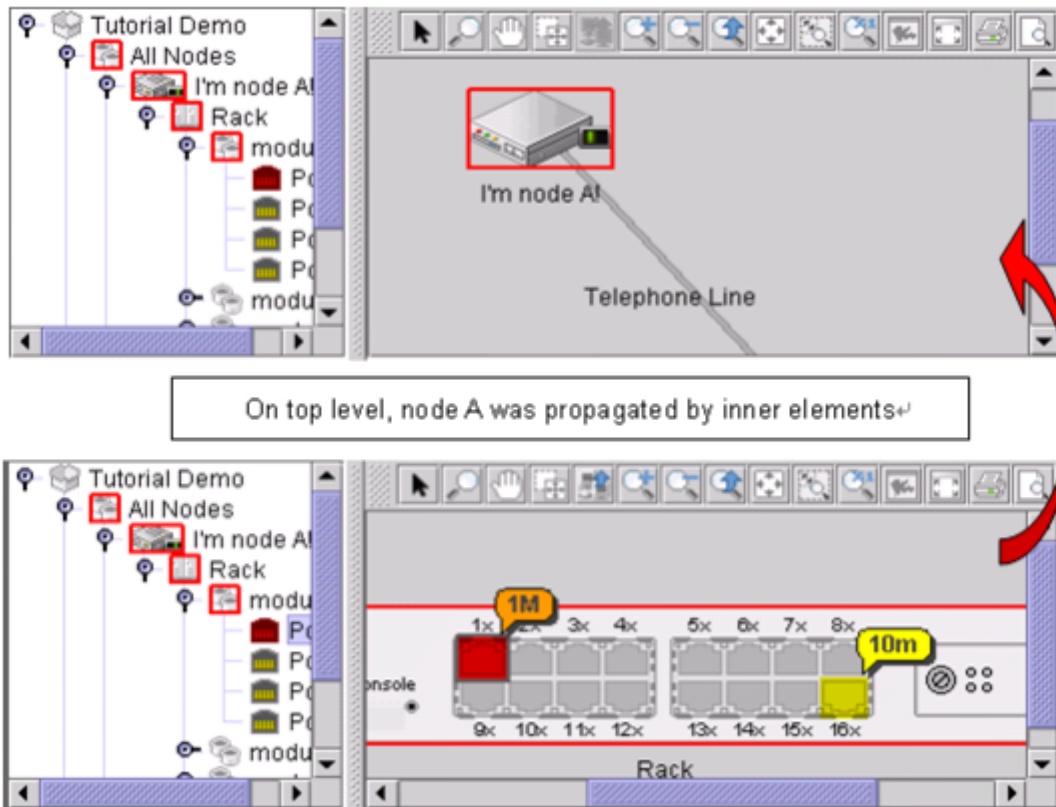
In addition, if you want the alarm state can be propagated to its parent element automatically, you can set a propagator to DataBox. TWaver provides a default propagator to propagate the most severe alarm severity to its parent element.

This section of the code is referred to as Step 7.

```
private void step7() {
    //create and set a summing propagator to the data source,
    //here will make the box propagate alarms to its parent.
    box.setAlarmPropagator(new SummingAlarmPropagator());

    //get a port in the equipment rack.
    Port nodeA = (Port) box.getElementByID("0:0");
    AlarmState alarmState = nodeA.getAlarmState();
    //add an acknowledged alarm with critical severity.
    alarmState.addAcknowledgeAlarm(AlarmSeverity.CRITICAL);
    //add and new alarm with major severity.
    alarmState.addNewAlarm(AlarmSeverity.MAJOR);


    //get another port.
    Port nodeB = (Port) box.getElementByID("3:3");
    alarmState = nodeB.getAlarmState();
    //add 10 new alarms with critical minor.
    alarmState.increaseNewAlarm(AlarmSeverity.MINOR, 10);
}
```



Elements with new alarms and acknowledge alarms

Adding Decorated Icons

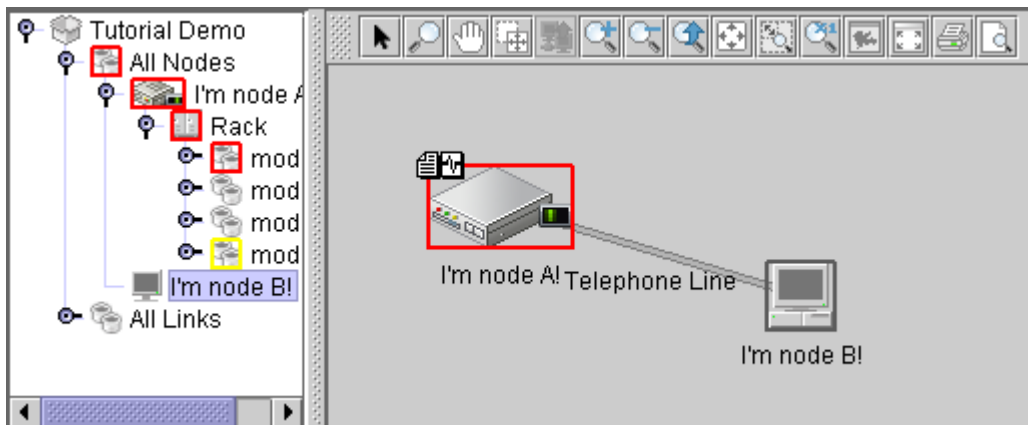
This section describes how to put an icon on the element objects. The icons can correspond to element secondary states or node and link decorations. TWaver provides a set of predefined icons to represent the element state that are managed in IconAttachmentHolder. It's very easy to customize new icon and add to the element objects.

To demonstrate this function, we put a predefined icon on the managed object. Then we create and publish a customized icon on the managed object. We need a small image icon like this: 

This section of the code is referred to as step 8.

```
//define a new LayoutedIconAttachment.
//It must define as public static class.
public static class MyIconAttachment extends IconAttachment{
    public MyIconAttachment(String name, ElementUI ui) {
        super(name, ui, TWaverUtil.getImageIcon("myIcon.png"));
    }
}

private void step8() {
    String iconName="document";
    TUIManager.registerAttachment(iconName, MyIconAttachment.class);
    //put a "document" icon on element B.
    Element element = box.getElementByID("A");
    element.addAttachment(iconName);
}
```



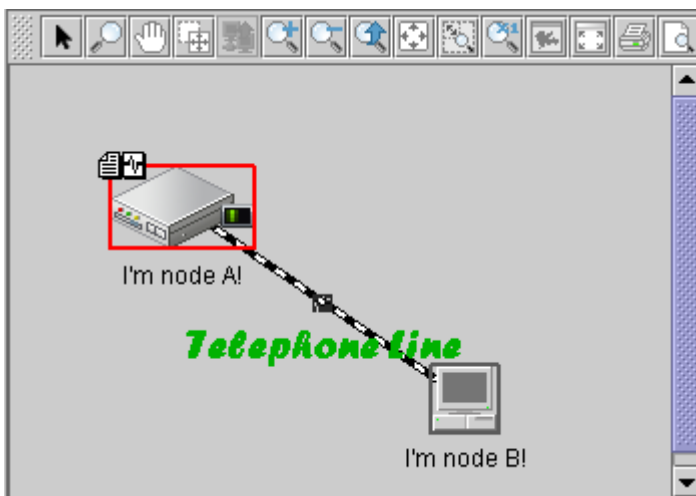
Adding Special Effect

TWaver provides rich rendering options to present an attractive look and can be customized. The following example shows you the specially effect for link element.

To demonstrate this function, we make the link shows as animating flowing effect, and set the colors of the link body, outline.

This section of the code is referred to as Step 9.

```
private void step9() {
    //get the link element.
    Element element = box.getElementByID("link");
    //make the link animating flowing
    element.putLinkFlowing(true);
    //set the link flowing color
    element.putLinkFlowingColor(Color.black);
    //set the link outline color
    element.putLinkOutlineColor(Color.black);
    //set the link body color.
    element.putLinkColor(Color.white);
    //set the link label font
    element.putLabelFont(new Font("Impact", 1, 20));
    //set the link label color
    element.putLabelColor(Color.MAGENTA);
}
```



Predefined Managed Object

This chapter explains how to use TWaver predefined elements for your application.

- [Common Features](#)
- [Using Node Element](#)
- [Using Link Element](#)
- [Using Group Element](#)
- [Using SubNetwork Element](#)
- [Using BTS & BTSAntenna Element](#)
- [Equipment Elements](#)
- [Using Follower Element](#)
- [Using Element Properties](#)
- [Element Self-Copy](#)

Common Features

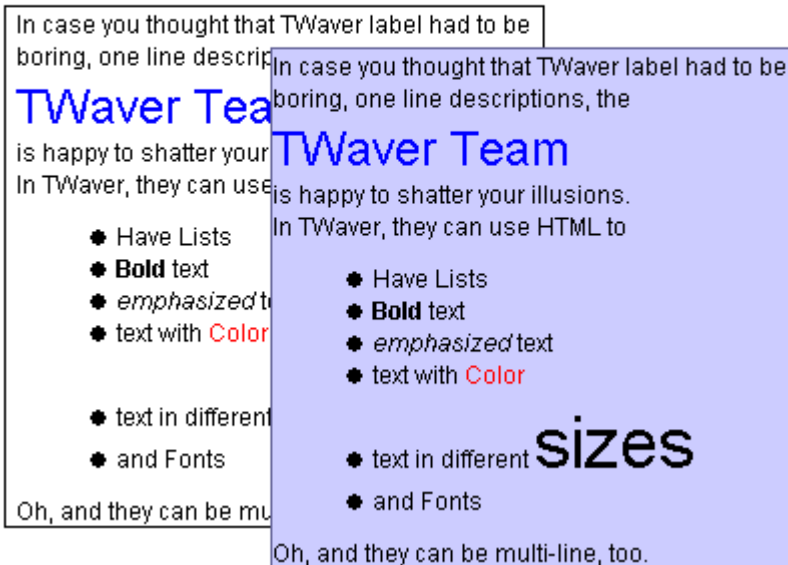
Common Features

- [Using HTML Label](#)
- [Using Alpha Transparent](#)

Using HTML Label

Each Element object in TWaver has a Name property. You can use setName method to specify a string text as the Element name. The name will be displayed as Element label on all TWaver components. You can use HTML code as the Element name to make more interesting label instead of a single line text:

```
Node node = new Node();
//set a long label for this node.
node.setDisplayName (" <html>In case you thought that TWaver label had to be " +
"<p>boring, one line descriptions, the " +
"<p><font color=blue size=+2>TWaver Team</font>" +
"<p> is happy to shatter your illusions.<p>" +
"In TWaver, they can use HTML to " +
"<ul><li>Have Lists<li><b>Bold</b>" +
"text<li><em>emphasized</em>" +
"text<li>text with <font color=red>Color</font>" +
"<li>text in different <font size=+3>sizes</font>" +
"<li>and <font face=AvantGarde>Fonts</font></ul>" +
"Oh, and they can be multi-line, too.</html>");
node.getAlarmState().addNewAlarm(AlarmSeverity.CRITICAL);
box.addElement(node);
```



Note

- By default, TWaver use Element name as its tool tip text, unless you use method setToolTipText specified new tip text.

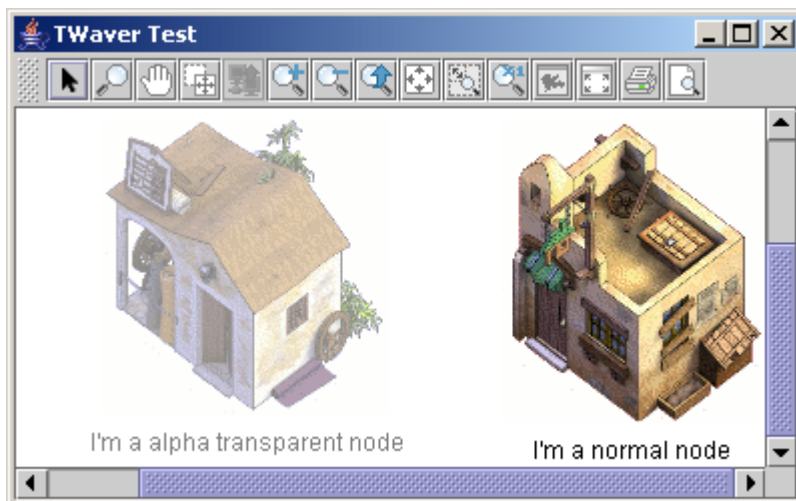
Using Alpha Transparent

You can specify the alpha transparent value for each Element.

```
Node node1 = new Node();
node1.setImage("/demo/network/1.png");
node1.setLocation(100, 100);
//set the alpha transparent value for node1.
node1.putRenderAlpha(0.5f);
node1.setName("I'm a alpha transparent node");
box.addElement(node1);
```

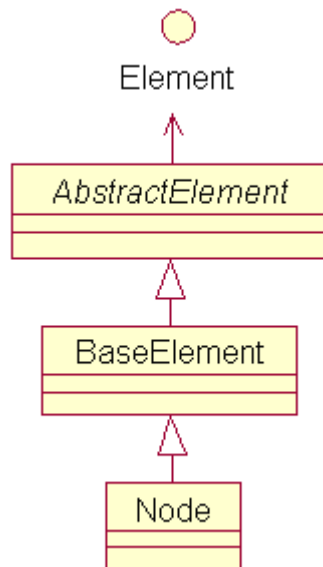
```
Node node2 = new Node();
node2.setImage("/demo/network/2.png");
node2.setLocation(300, 100);
node2.setName("I'm a normal node");
box.addElement(node2);
```

Run this program:

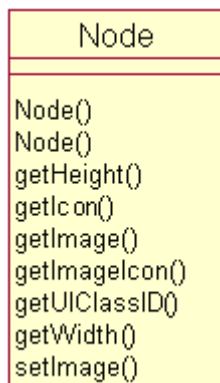


Using Node Element

Node element is the basic telecom managed object. It can present telecom Network Element.



Node extends BaseElement which are the simple implementation of Element interface. Node extends image property. Developers can specify and change node image to present all kinds of telecom equipment node.



Node Class Diagram

Node use NodeUI class to take the painting work.

- [Creating Node by API](#)
- [Creating Node by XML](#)
- [Using ShapeNode Element](#)


Creating Node by API

Creating Node by API

It is very easy to create a Node by API. You can create a Node instance and set the common properties and finally add it into a DataBox.

```
//create node without specified identifier
Node node1 = new Node();
//create node with specified identifier
Node node2 = new Node("Node2");
//create box, add all elements into box.
TDataBox box = new TDataBox();
box.addElement(node1);
box.addElement(node2);
//set node attributes.
node1.setLocation(100,100);
node1.setImage("node1.gif");
```

- Setting Node Image

By default, TWaver uses image  to paint Node instance. If you want to paint Node without image, just need to set the image to TWaverConst.BLANK_IMAGE, it will draw a rectangle. If you want to paint one kind node class without image, just invoke the method 'TUIManager.registerWithoutImage(Class elementClass)'.

- Node Size

By default, if Node has image, its size will be the image size. You can override getHeight/getWidth method to change the default size.

```
Node node = new Node() {
    public int getHeight() {
        return 50;
    }
    public int getWidth() {
        return 50;
    }
};
box.addElement(node);
```

Above code will scale the image to fit the specified size 50*50, it looks like:



- Node name:

You can set any string as the name of the Node. It will display below the image:

```
Node node = new Node() ;
node.setName("This is my name");
```



This is my name

Moreover, the name label has more properties which can be customized by a set of APIs. The names of these APIs are start with "putLabel" and follow a property name. For instance, "putLabelColor" is used to change the label color. See TWaver JavaDoc for more details.

Following codes show you how to use these client properties to customize a Node name lable:

```
Node node = new Node() ;
node.setName("This is my name");
node.putLabelFont(new Font("Forte",Font.ITALIC,20));
node.putLabelColor(Color.cyan);
node.putLabelVisible(true);
node.putLabelBorder(true);
box.addElement(node);
```



This is my name

Creating Node by XML

Use following code to load XML data from a XML file:

```
TDataBox box = new TDataBox();
box.parse("node.xml");
```

We compose a XML file to describe a Node instance:

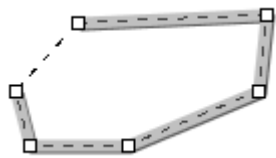
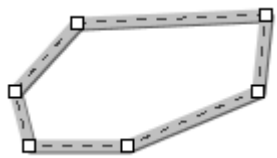
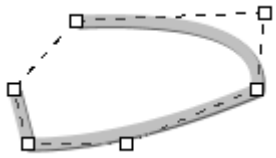
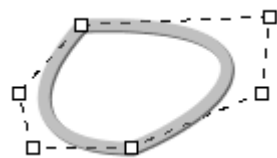
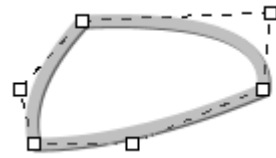
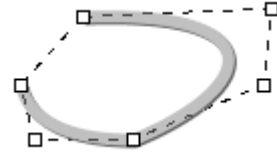

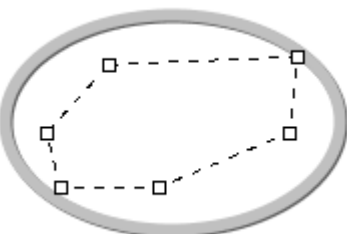
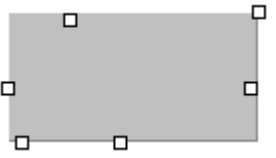
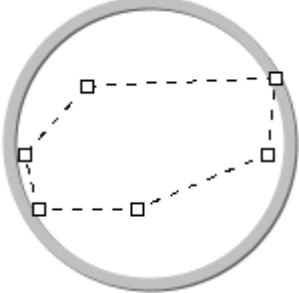
```
<?xml version="1.0" encoding="UTF-8"?>
<Java version="1.4.2_01" class="Java.beans.XMLDecoder">
  <object class="TWaver.Node">
    <void method="putClientProperty">
      <string>label.border</string>
      <boolean>true</boolean>
    </void>
    <void method="putClientProperty">
      <string>label.color</string>
      <object class="Java.awt.Color">
        <int>255</int>
        <int>0</int>
        <int>0</int>
        <int>255</int>
      </object>
    </void>
    <void method="putClientProperty">
      <string>label.font</string>
      <object class="Java.awt.Font">
        <string>Forte</string>
        <int>1</int>
        <int>30</int>
      </object>
    </void>
    <void property="location">
      <object class="Java.awt.Point">
        <int>310</int>
        <int>285</int>
      </object>
    </void>
    <void property="name">
      <string>This is my name</string>
    </void>
    <void property="selected">
      <boolean>true</boolean>
    </void>
    <void property="toolTipText">
      <string>This is my tip!</string>
    </void>
  </object>
</Java>
```



This is my name

Using ShapeNode Element

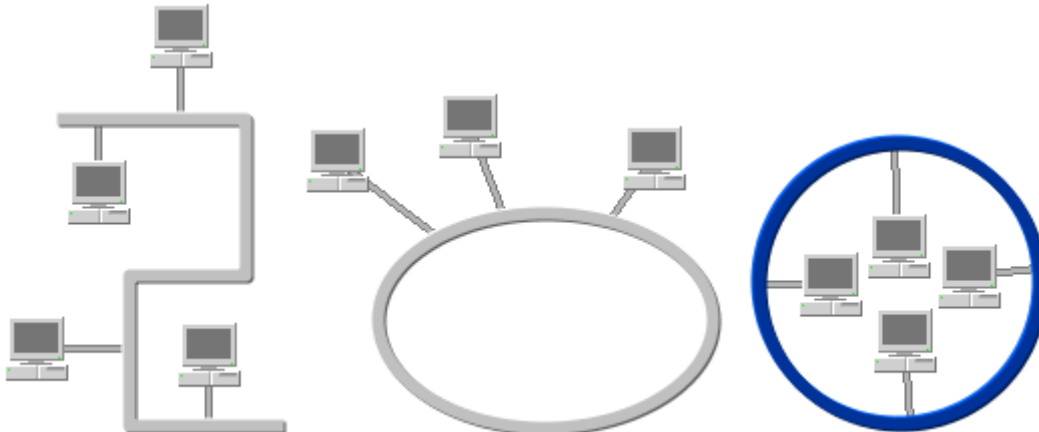
TWaver provides a special class `twaver.ShapeNode` to present some managed object with a complicated shape. `ShapeNode` is a special node combined with a set of points. Use `ShapeNode` you can present some special data on topology, such as bus network. Below are some examples:

 SHAPENODE_STRAIGHT_LINE	 SHAPENODE_CLOSE_STRAIGHT_LINE
 SHAPENODE_QUADRATIC_CURVE	 SHAPENODE_CLOSE_BEZIER_CURVE
 SHAPENODE_CLOSE_QUADRATIC_CURVE	 SHAPENODE_BEZIER_CURVE
 SHAPENODE_ORTHOGONAL_LINE	 SHAPENODE_ELLIPSE
 SHAPENODE_NONE	 SHAPENODE_ROUND

 Note

All ShapeNode above are in selected status. The small white blocks are indicating the control points of the shape.

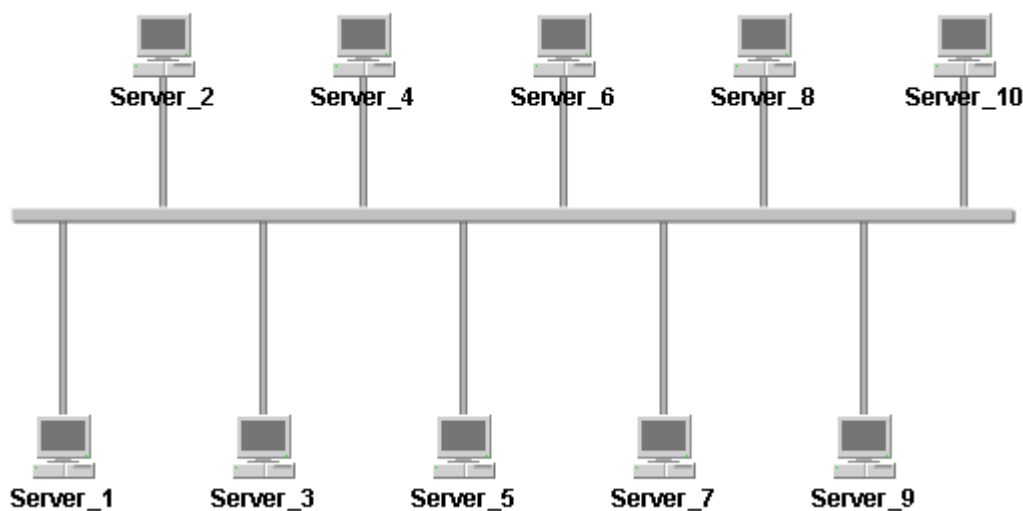
Below are several bus network examples built by twaver.ShapeNode.



Here is an example using orthogonal line in ShapeNode:

```
TDataBox box =new TDataBox();
TNetwork network=new TNetwork(box);
ShapeNode bus=new ShapeNode();
box.addElement(bus);
bus.addPoint(new Point(40,200));
for(int i=1;i<=10;i++){
    Node node=new Node();
    node.setName("Server_" +i);
    node.setLocation(50*i,100+200*(i%2));
    box.addElement(node);
    box.addElement(new Link(bus,node));
}
bus.addPoint(new Point(540,200));
//set type of ShapeNode to orthogonal line.
bus.setShapeNodeType(TWaverConst.SHAPENODE_ORTHOGONAL_LINE);
//set the joint point type to most close point
bus.putShapeNodeJointPoint(TWaverConst.JOINT_POINT_NEAR);
```

Run this example:

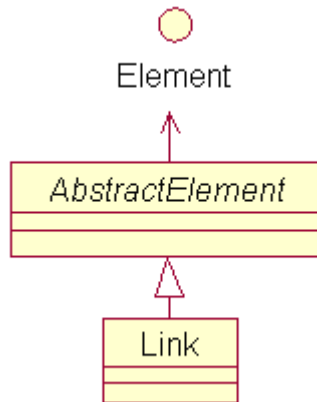


**Note**

In current TWaver, only types `ORTHOGONAL_LINE`, `ELLIPSE` and `ROUND` can be used in `ShapNode` for the "bus network".

Using Link Element

Link object can be used to represent many telecom resources like cable, circuit, fiber, route etc.



Link extends AbstractElement, provides more properties like start node, end node, link type etc. Link uses LinkUI to paint itself.



- [Creating Link by API](#)
- [Creating Link by XML](#)
- [Working with Link Bundle](#)
- [Using Link Bundle Agent](#)
- [Using Link Type](#)

Creating Link by API

To create a Link instance, you must specify the link start node and link end node:

```
Node fromNode = new Node();
fromNode.setName("I'm from node");
fromNode.setLocation(100, 100);
box.addElement(fromNode);

Node toNode = new Node();
toNode.setName("I'm to node");
toNode.setLocation(200, 100);
box.addElement(toNode);

Link link = new Link(fromNode, toNode);
link.addAttachment(TWaverConst.ATTACHMENT_LINK_FIBER);
link.putAttachmentPosition(TWaverConst.POSITION_CENTER);
box.addElement(link);
```

It will look like:



I'm from node I'm to node

- More Link Properties

All extends properties can be accessed by a set of APIs which start with "putLink" and follow a property name. See TWaver JavaDoc for more details.

Following code shows you how to use these client properties to control a link appearance:

```
Link link = new Link(fromNode, toNode);
link.addAttachment(TWaverConst.ATTACHMENT_LINK_FIBER);
link.putAttachmentPosition(TWaverConst.POSITION_CENTER);
box.addElement(link);

link.putLinkStyle(TWaverConst.LINK_STYLE_DASH);
link.putTextureFactory(TWaverConst.TEXTURE_GRID_SQUARE);
link.putLinkOutlineColor(Color.red);

link.getAlarmState().addNewAlarm(AlarmSeverity.CRITICAL);
```



I'm from node I'm to node

Creating Link by XML

Use following code to load XML data from a XML file:

```
TDataBox box = new TDataBox();
box.parse("link.xml");
```

We compose a XML file to describe a Link instance:

```
<?xml version="1.0" encoding="UTF-8"?>
<Java version="1.4.2_01" class="Java.beans.XMLDecoder">
  <object class="TWaver.Link">
    <void method="putClientProperty">
      <string>link.texture</string>
      <boolean>>false</boolean>
    </void>
    <void method="putClientProperty">
      <string>link.outline.color</string>
      <object class="Java.awt.Color">
        <int>0</int>
        <int>0</int>
        <int>0</int>
        <int>255</int>
      </object>
    </void>
    <void method="putClientProperty">
      <string>link.flowing.color</string>
      <object class="Java.awt.Color">
        <int>255</int>
        <int>255</int>
        <int>102</int>
        <int>255</int>
      </object>
    </void>
    <void method="putClientProperty">
      <string>link.width</string>
      <int>12</int>
    </void>
    <void method="putClientProperty">
      <string>link.flowing</string>
      <boolean>true</boolean>
    </void>
    <void method="putClientProperty">
      <string>link.color</string>
      <object class="Java.awt.Color">
        <int>0</int>
        <int>0</int>
        <int>255</int>
        <int>255</int>
      </object>
    </void>
    <void method="putClientProperty">
      <string>link.bundle.index</string>
      <int>0</int>
    </void>
    <void method="putClientProperty">
      <string>link.fromArrow</string>
      <boolean>>false</boolean>
    </void>
```

```

<void method="putClientProperty">
  <string>link.outline.width</string>
  <int>2</int>
</void>
<void method="putClientProperty">
  <string>link.flowing.converse</string>
  <boolean>true</boolean>
</void>
<void method="putClientProperty">
  <string>label.color</string>
  <object class="Java.awt.Color">
    <int>255</int>
    <int>0</int>
    <int>0</int>
    <int>255</int>
  </object>
</void>
<void method="putClientProperty">
  <string>link.flowing.width</string>
  <int>8</int>
</void>
<void method="putClientProperty">
  <string>link.dash</string>
  <boolean>false</boolean>
</void>
<void method="putClientProperty">
  <string>label.font</string>
  <object class="Java.awt.Font">
    <string>Berlin Sans FB Demi</string>
    <int>0</int>
    <int>28</int>
  </object>
</void>
<void method="putClientProperty">
  <string>link.bundle.size</string>
  <int>1</int>
</void>
<void property="from">
  <object id="Node0" class="TWaver.Node">
    <void property="location">
      <object class="Java.awt.Point">
        <int>58</int>
        <int>96</int>
      </object>
    </void>
    <void property="name">
      <string>Node</string>
    </void>
  </object>
</void>
<void property="name">
  <string>I&apos;m a link!</string>
</void>
<void property="to">
  <object id="Node1" class="TWaver.Node">
    <void property="location">
      <object class="Java.awt.Point">
        <int>380</int>
        <int>95</int>
      </object>
    </void>
    <void property="name">
      <string>Node</string>
    </void>
  </object>
</void>

```

```

</object>
<object idref="Node0"/>
<object idref="Node1"/>
</Java>

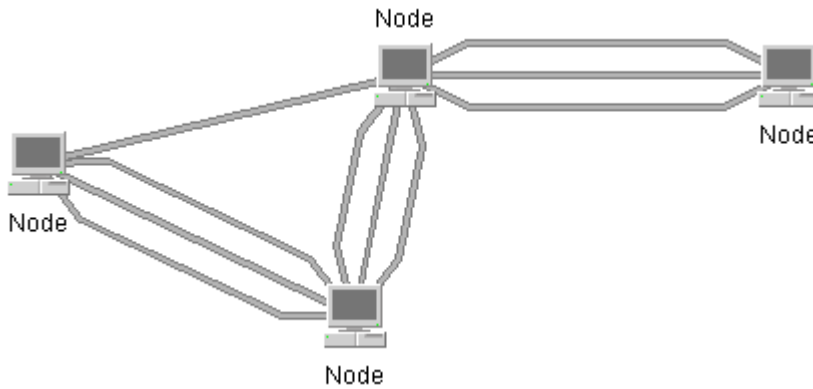
```



Working with Link Bundle

For the links which have the same start/end nodes, network components will bundle all of them into one link automatically. When the bundle link was double clicked, it will expand to display each link with the normal appearance.

Here is what the link bundle looks like in TWaver:



You can double click to expand or collapse the link bundles.

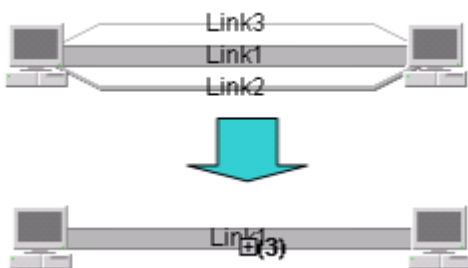
Using Link Bundle Agent

By default TWaver picks a random Link from the Link set to represent the link bundle when the bunch of Links is bundled. A Generator can be set to determine which Link should be the 'agent' of the Link set. In this example, we create three Links between two Nodes:

```
//create link1.
final Link link1 = new Link(from, to);
link1.putLinkWidth(10);
link1.putLabelPosition(TWaverConst.POSITION_CENTER);
link1.setName("Link1");
box.addElement(link1);
//create link2.
final Link link2 = new Link(from, to);
link2.putLink3D(true);
link2.putLabelPosition(TWaverConst.POSITION_CENTER);
link2.setName("Link2");
box.addElement(link2);
//create link3.
final Link link3 = new Link(from, to);
link3.putLinkWidth(0);
link3.putLinkOutlineWidth(0);
link3.putLabelPosition(TWaverConst.POSITION_CENTER);
link3.setName("Link3");
box.addElement(link3);
```

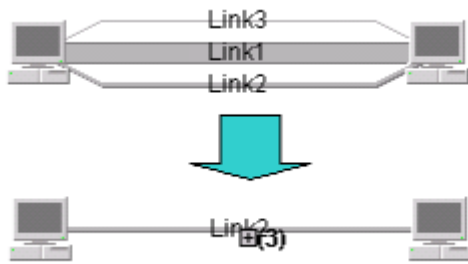
We use link1 as link bundle agent:

```
box.setLinkBundleAgentGenerator(new Generator() {
    public Object generate(Object object) {
        return link1;
    }
});
```



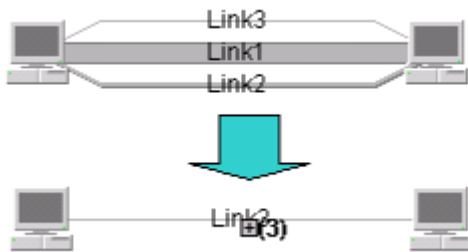
Use link2 as link bundle agent:

```
box.setLinkBundleAgentGenerator(new Generator() {
    public Object generate(Object object) {
        return link2;
    }
});
```



Use link3 as link bundle agent:

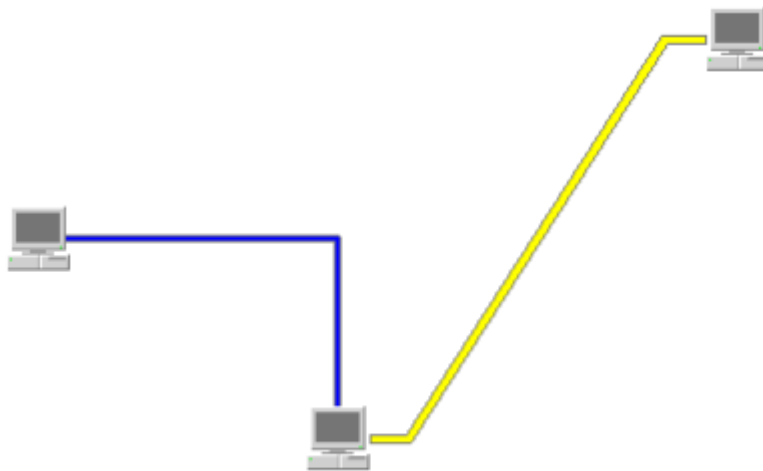
```
box.setLinkBundleAgentGenerator(new Generator() {
    public Object generate(Object object) {
        return link3;
    }
});
```



Using Link Type

From 1.5.0, TWaver use link type to distinguish different links which have different shapes. Users can find the definitions in TWaverConst. They are indicated with prefix 'LINK_TYPE_*'.

```
Node node1 = new Node();
node1.setLocation(400, 200);
box.addElement(node1);
Node node2 = new Node();
node2.setLocation(200,400);
box.addElement(node2);
Node node3 = new Node();
node3.setLocation(50,300);
box.addElement(node3);
Link link1 = new Link(node1,node2);
Link link2 = new Link(node2,node3);
link1.setLinkType(twaver.TWaverConst.LINK_TYPE_FLEXIONAL);
link2.setLinkType(twaver.TWaverConst.LINK_TYPE_ORTHOGONAL);
link1.putLinkWidth(4);
link2.putLinkWidth(3);
link1.putLinkColor(Color.YELLOW);
link2.putLinkColor(Color.BLUE);
box.addElement(link1);
box.addElement(link2);
```



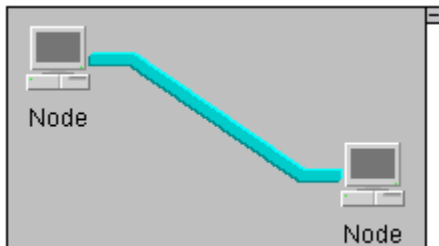
See [Link Property List](#) for more information

Using Group Element

Group is an Element container. Group can be used to contain other node elements. You can close or open Group by double click Group. When a group closed, it displayed like a normal node and all internal elements are hidden. When it opens, all internal elements will display normally and group displayed as a rectangle region which can cover all contained elements position.



Closed Group

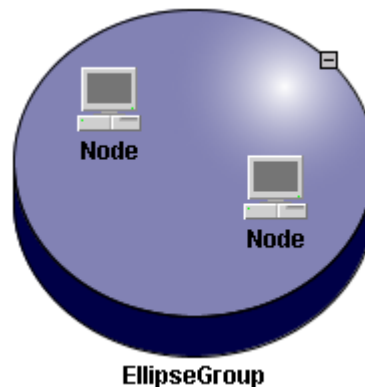
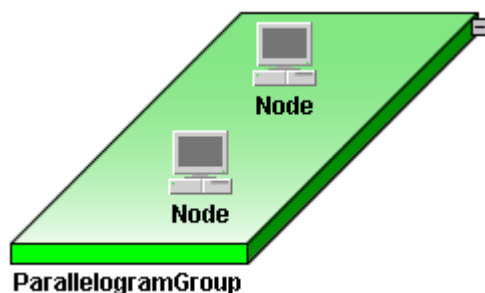


Expanded Group

- AbstractElement -> Element
- ● BaseElement -> Cloneable
- ● Node
- ● Group

Group Hierarchy

Group is derived from Node class. The Group has its own body shape such as rectangle, circle or other shapes. No matter what shape, it will always cover all children elements.



Group use a type property to determinate the shape of Group.

- TWaverConst.GROUP_TYPE_REGULAR
 - TWaverConst.GROUP_TYPE_ELLIPSE
 - TWaverConst.GROUP_TYPE_PARALLELOGRAM
 - TWaverConst.GROUP_TYPE_ROUND
 - TWaverConst.GROUP_TYPE_ROUND_RECTANGLE
 - TWaverConst.GROUP_TYPE_OCTAGON
-
- [Using Group](#)
 - [Creating Group by XML](#)
 - [Group Properties](#)

Using Group

You can create a Group instance like a Node. Then create some children elements and add them into this group. You can do that by either `child.setParent` or `parent.addChild`:



```
Node fromNode = new Node();
fromNode.setLocation(100, 100);
box.addElement(fromNode);

Node toNode = new Node();
toNode.setLocation(200, 100);
box.addElement(toNode);




Link link = new Link(fromNode, toNode);
link.addAttachment(TWaverConst.ATTACHMENT_LINK_FIBER);
link.putAttachmentPosition(TWaverConst.POSITION_CENTER);
box.addElement(link);

Group group=new Group ();
group.addChild(fromNode);
group.addChild(toNode);
group.addChild(link);
box.addElement(group);
```

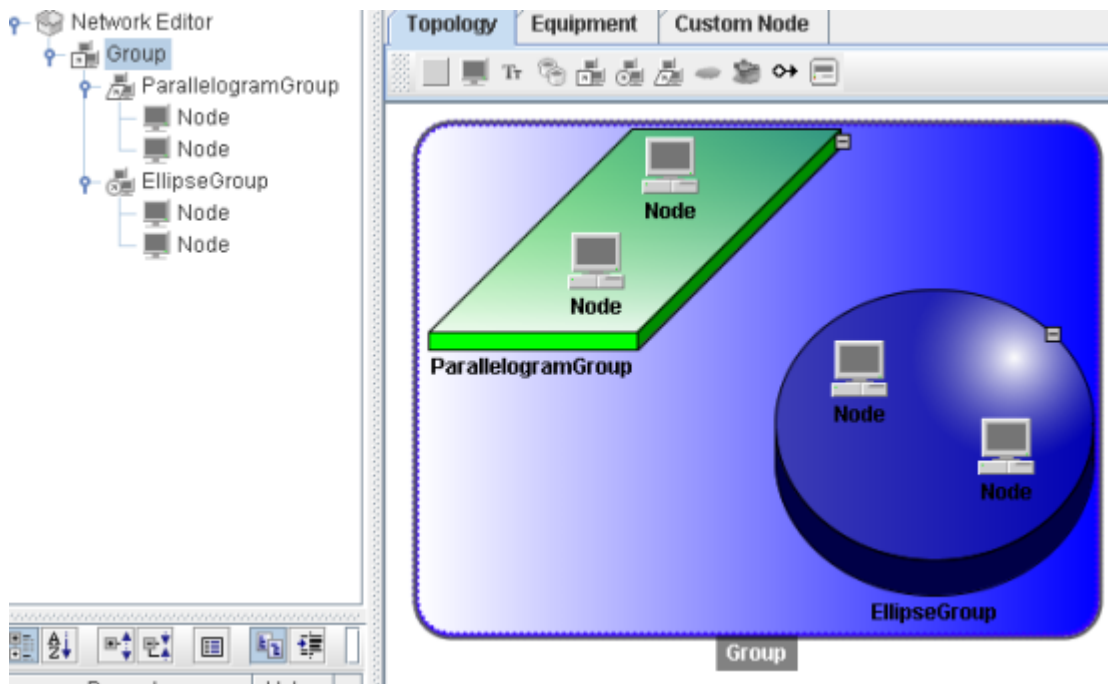
You can open or close a Group by:

- Click group;
- Click group handler  or ;
- Invoke method `public void setExpand(boolean expand)`

Group show a small icon in the right-top corner to show its current status. It can be:

-  Closed status. Click can be opened.
-  Opened. Click can be closed.
-  Empty Group. No element inside. No response to Click.

TWaver also support nest Group. One Group can contain another Group object.



Nest Group

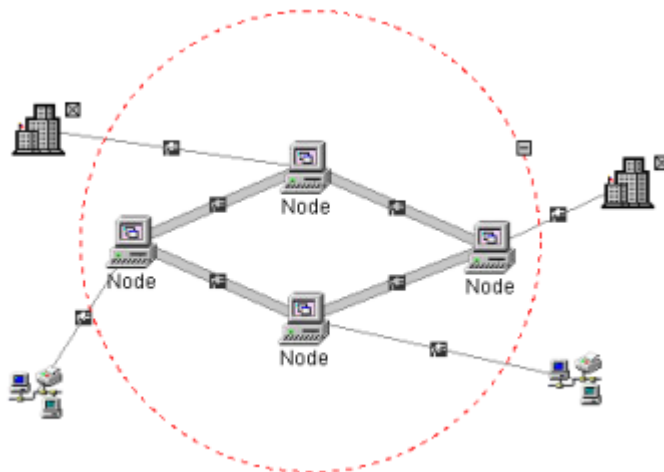
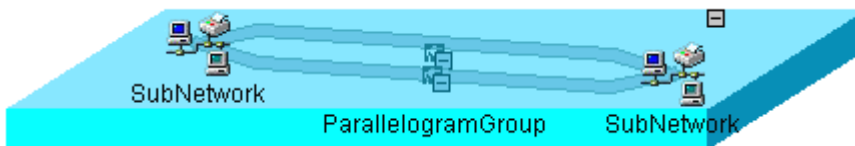
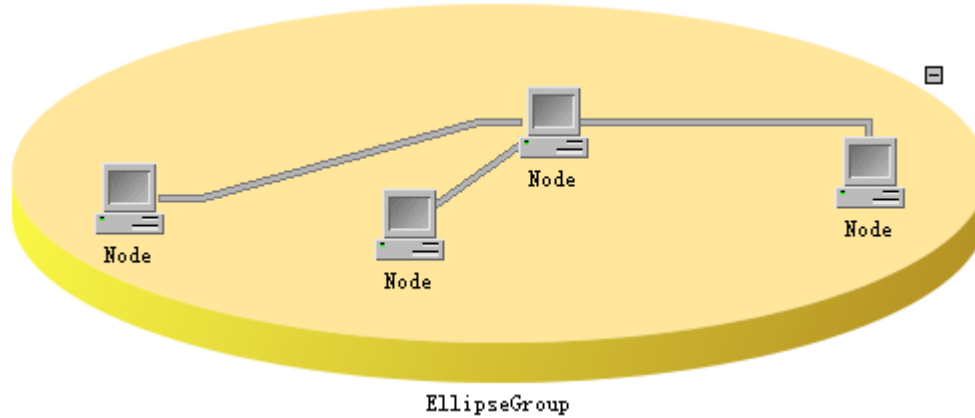
Creating Group by XML

Creating Group by XML

Create Group by XML is very similar with creating a Node by XML. Here we no longer give more examples.

Group Properties

Group extends a set of APIs, all these methods are start with "putGroup" and plus the property name. You can use them to change the outline, color, fill or other properties for Group. See TWaver JavaDoc for more details. These are some examples for Group:



Using SubNetwork Element

- [What is SubNetwork](#)
- [Difference between SubNetwork and Group](#)
- [Inheritance of SubNetwork](#)
- [Using TSubNetwork Background](#)
- [Using SubNetwork Data File](#)
- [SubNetwork Client Properties](#)
- [Creating SubNetwork via Shape](#)

What is SubNetwork

SubNetwork is a special element container. Like Group, it can contain many other elements inside and displayed like a Node. But SubNetwork has its own map context such as XML data file, background map, location of scroll view, zoom value etc. That is, when you double click a SubNetwork, you will get into a new map and all internal elements will be displayed.

SubNetwork is a very useful element. It can organize the whole network into multi-layers which can be drill-down. With SubNetwork you can very easy to organize large network data with an easy-understanding structure.

In TWaver, TSubNetwork is an Interface which defines the behaviour of a SubNetwork. Any elements implement this interface will be able to drill-down. That means you can customize a Node or a Link to a SubNetwork.

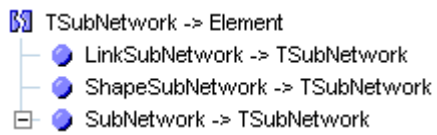
Please check [TSubNetwork Interface Method list](#) for more details.

Difference between SubNetwork and Group

SubNetwork and Group are both element containers. Both of them can be closed or opened. When closed, they can hide internal elements and displayed like a normal node. The difference is that Group will display children elements in current sub network view. TSubNetwork element will display children elements inside its sub network view.

Inheritance of SubNetwork

SubNetwork is an interface, it defines the Element that can be drill down.



Using TSubNetwork Background

Using TSubNetwork Background

Background image are designed for Network component. But it can be used in TSubNetwork objects with setBackground method:

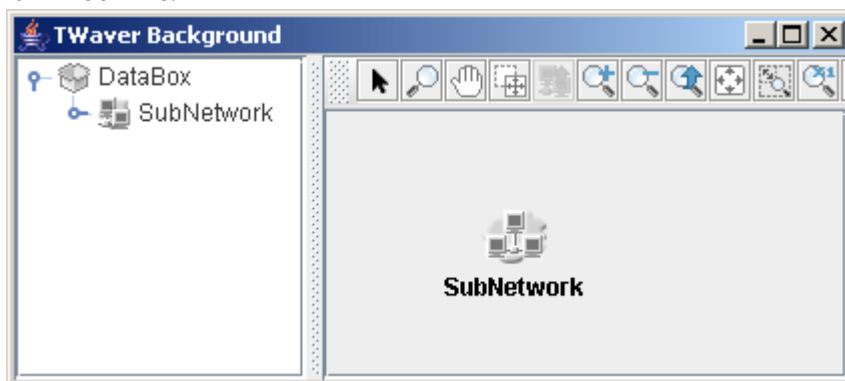
```
Node node1 = new Node();
node1.setName("A");
node1.setLocation(10, 50);
box.addElement(node1);

Node node2 = new Node();
node2.setName("B");
node2.setLocation(100, 50);
box.addElement(node2);

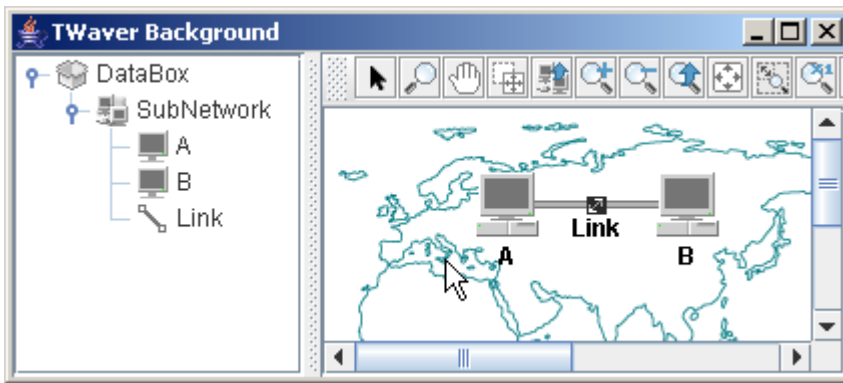
Link link = new Link(node1, node2);
link.setName("Link");
link.addAttachment(TWaverConst.ATTACHMENT_LINK_FIBER);
link.putAttachmentPosition(TWaverConst.POSITION_CENTER);
box.addElement(link);

SubNetwork subnetwork=new SubNetwork();
subnetwork.setLocation(50,50);
subnetwork.setName("SubNetwork");
String url= "file:/c:/mydir/map.png";
subnetwork.setBackground(new ImageBackground(url));
subnetwork.addChild(node1);
subnetwork.addChild(node2);
subnetwork.addChild(link);
box.addElement(subnetwork);
```

It will look like:



Top level topology



Drill-down SubNetwork

Using SubNetwork Data File

The other important feature of TSubNetwork is that you can set its own datasource for a TSubNetwork object. It can be a remote XML datasource, or a XML file on the disk.

The standalone datasource of the TSubNetwork can provide another important feature: network data on-demand-load. That is, the elements of a TSubNetwork can be loaded only when the TSubNetwork firstly drilled down. Once its data initialized, TWaver will not load data again. The benefit is, for a large carrier-class network, you don't need to create all elements or data when you launch the program. You can only create the top level topology:

- Reduce element creation, speed up program launch time.
- Reduce memory consumption.
- Simply your codes and design.

You can consider TSubNetwork datasource in following situation:

- Chassis of the equipment.
- TSubNetwork which the topology seldom changed.

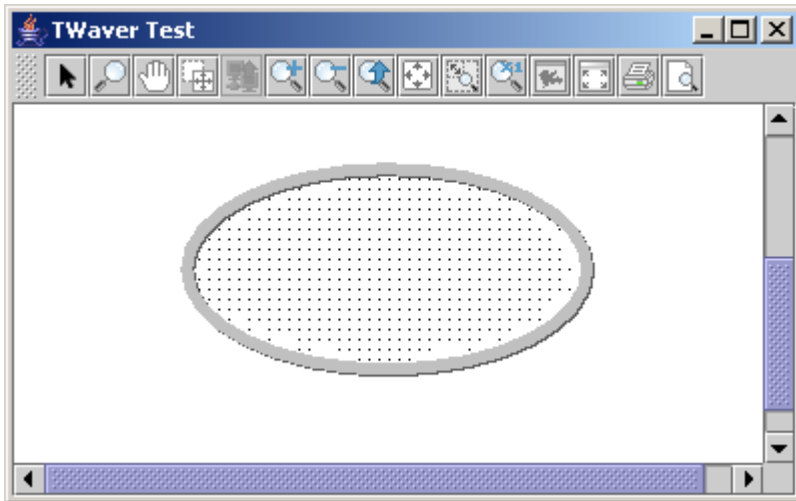
SubNetwork Client Properties

Currently TWaver doesn't define any client properties for TSubNetwork.

Creating SubNetwork via Shape

ShapeSubNetwork is a special TSubNetwork that can be created by a given shape. That means you can create ShapeSubNetwork with any Java2D shape you want.

```
Ellipse2D ellipse = new Ellipse2D.Float(100, 100, 200, 100);
ShapeSubNetwork subnetwork = new ShapeSubNetwork(ellipse);
box.addElement(subnetwork);
```



Using BTS & BTSAntenna Element

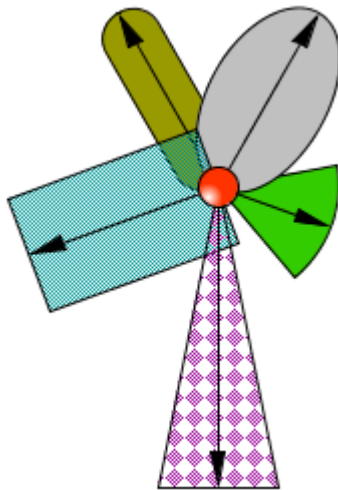
BTS (Base Transceiver Station) is a mobile phone base station. A BTS contains transmit and receive technology and also the aerals to supply a radio cell. Several BTSs are administered by a BSC (Base Station Controller), which is in turn under an MSC (Mobile Switching Center). Existing BSCs and BTSs can be extended for new radio technology to allow the network operator to reuse existing aerial sites for UMTS radio networks. TWaver provides BTS element and BTSAntenna element to present BTS equipment.

Use `BTS.addAntenna` or `BTSAntenna.setBTS` to connect a BTS and a `BTSAntenna`. TWaver provides several types for BTS Antenna. Each type will display in different shape.

- `TWaverConst.ANTENNA_TYPE_SECTOR`
- `TWaverConst.ANTENNA_TYPE_ELLIPSE`
- `TWaverConst.ANTENNA_TYPE_TRIANGLE`
- `TWaverConst.ANTENNA_TYPE_RECTANGLE`
- `TWaverConst.ANTENNA_TYPE_ROUND_RECTANGLE`

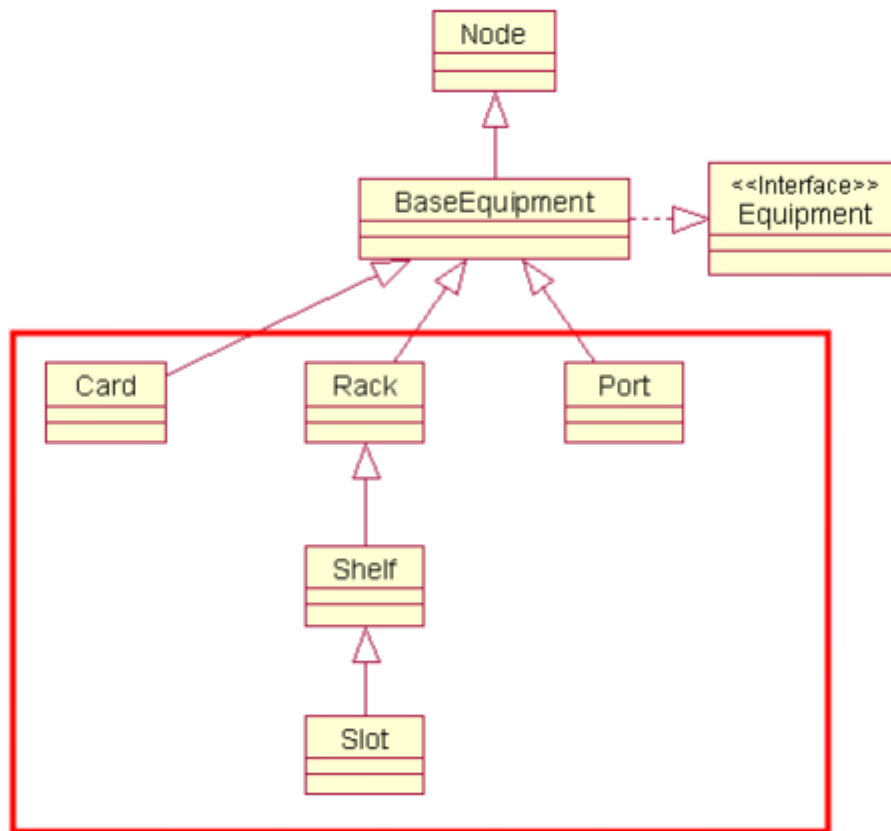
More properties of `BTSAntenna`:

- `beemAlpha`: alpha transparency of the fill color
- `beemDirection`: antenna start angle
- `beemWidth`: antenna span angle
- `power`: antenna radius

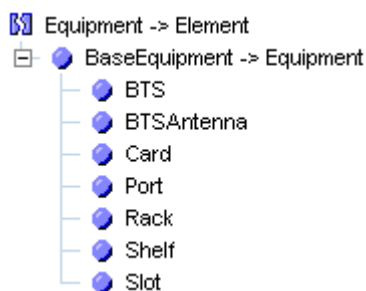


Equipment Elements

TWaver defined a set of elements used to create telecom equipment chassis:



All these elements are derived from Node and implements Equipment interface:



For the OSS system, the most popular equipment elements include Rack, Shelf, Slot, Card, Port etc. But telecom equipments vary widely. It's very hard to cover all kinds of equipment chassis no matter how many predefined elements are provided by TWaver. So, TWaver tries to provide a flexible, extendable framework. Developer can extend more chassis elements easily.

TWaver has not specified the containable rules for TWaver elements. You can design an equipment chassis without Rack, or Rack contains another Rack as the sub rack as long as it fits your equipment physical structure.

- [Grid Element](#)
- [Chassis Element](#)
- [Rack Element](#)
- [Shelf Element](#)
- [Slot Element](#)
- [Card Element](#)
- [Port Element](#)

- [Extending Equipment Element](#)

Grid Element

Grid is a predefined element object for editing equipment panel efficiently and rapidly. It is very easy to use it to create a network equipment panel . It contains some properties as following:

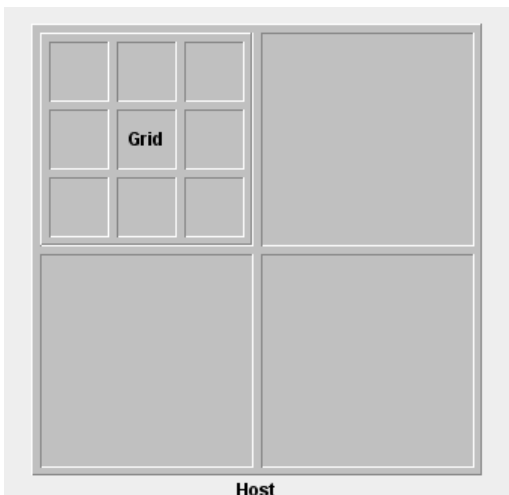
- (1)rowCount/columnCount: The row count or the column count that the grid is divided.
- (2)rowIndex/columnIndex: The row location and column location that the grid is hosted by its parent Grid. This is used to put a Grid in another Grid.
- (3)columnPercents/rowPercents: Assign the widths/heights for columns/rows of a Grid by percentage. For the rows/columns which not specified by this method parameters, it will be zero if the total percentage of the specified rows/columns is greater than or equal 100%; otherwise, the rest space will be assigned to the rest rows/columns evenly.

Create a Grid by API is easy. Here is an example:

```
Grid host = new Grid();
host.setRowCount(2);
host.setColumnCount(2);
host.setSize(300, 300);
host.setName("Host");

Grid grid = new Grid();
grid.setRowIndex(0);
grid.setRowCount(3);
grid.setColumnCount(3);
grid.setColumnIndex(0);
grid.putLabelPosition(TWaverConst.POSITION_CENTER);
grid.setName("Grid");

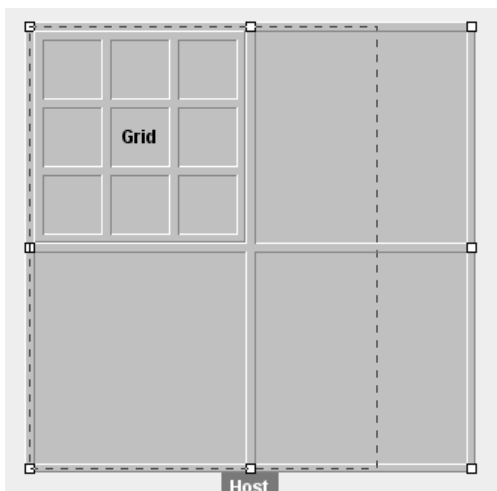
grid.setHost(host);
```



Quick Edit

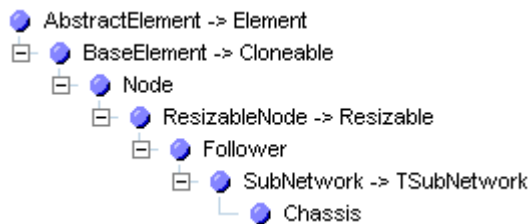
If the TNetwork is editable, then Grid is editable. In this case, a dashed rectangle will appear when the mouse is moved to the border area. You can drag & drop to change the Grid size.

Check the edit function in TWaver demo "editor/ChassisEditorDemo".

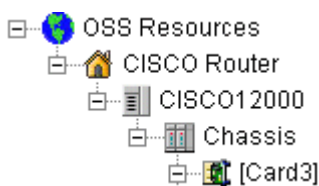


Chassis Element

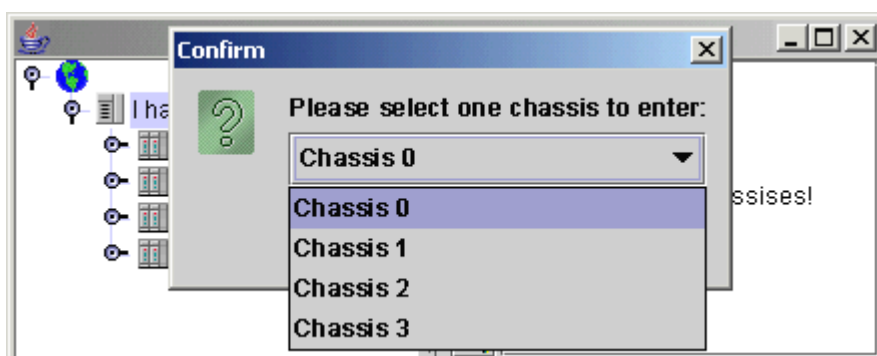
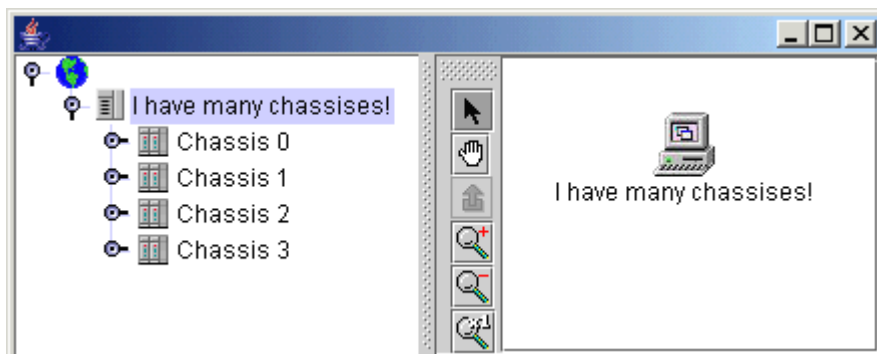
Chassis is an invisible elements container which represents one chassis panel of equipment:

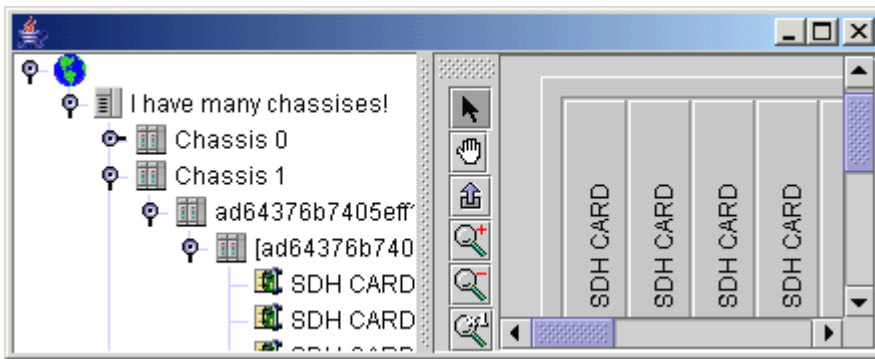


Chassis are not instance of Equipment interface. It is an equipment container. It extends SubNetwork and displays all elements in a standalone layer. As an abstract element, chassis is invisible on Network component. But you find it on the Tree component:



Chassis represents one telecom equipment chassis panel. Normally we add it to a Node instance as the child element. When the Node double clicked, TWaver will automatically drill into the chassis view. Some complicated telecom equipment has more than one chassis. They can have rear chassis and front chassis etc. TWaver supports multi-chassis equipment. That is, when you double click the equipment, TWaver will check all chassis listed in the dialog. User can select the chassis they want to see:





Multi-chassis equipment

Chassis use ChassisUI to paint itself. In fact they do nothing because chassis are invisible on network component. Chassis extends SubNetwork



Rack Element

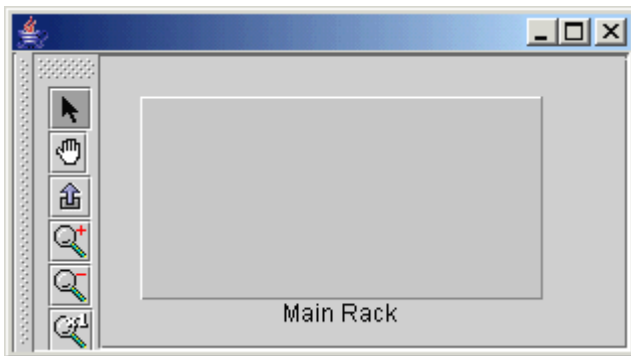
Rack defined as the equipment rack. Generally Rack is only a physical rack used to support telecommunication equipments. You can add Rack element as the child element of a Chassis. Some small equipment like modem may have no rack. If so, you can just add card or port into chassis.

Create a Rack instance like following:

```
Node node = new Node();
node.setLocation(10, 50);
box.addElement(node);

Chassis chassis = new Chassis();
chassis.setParent(node);
box.addElement(chassis);

Rack rack = new Rack();
rack.setName("Main Rack");
rack.setLocation(20, 20);
rack.setSize(200, 100);
rack.setParent(chassis);
box.addElement(rack);
```



By default, rack will be shown as a 3D rectangle. You can set a rack image to make it more intuitive.

Shelf Element

Shelves are used as an equipment shelf.

Shelf normally is a physical container, no further properties provided. You can specify its position, size, image or child elements. You can add slot, card or port into shelf according to your equipment physical structure.

Following code shows you how to use Rack and Shelf:

```
Rack rack = new Rack();
rack.setName("Main Rack");
rack.setLocation(20, 20);
rack.setSize(200, 100);
rack.setParent(chassis);
box.addElement(rack);
```

```
Shelf shelf1=new Shelf();
shelf1.setLocation(22,22);
shelf1.setSize(196,43);
shelf1.setParent(rack);
box.addElement(shelf1);
```

```
Shelf shelf2=new Shelf();
shelf2.setLocation(22,74);
shelf2.setSize(196,43);
shelf2.setParent(rack);
box.addElement(shelf2);
```



Main Rack

Slot Element

Slot element represents one empty card carrier. It can contain Card element. Similar to Rack and Shelf classes, Slot is just a physical container. You can use it like this:

```
for(int i=0;i<8;i++){
    Slot slot=new Slot();
    slot.setLocation(24+i*24,24);
    slot.setSize(20,38);
    slot.setParent(shelf1);
    box.addElement(slot);
}
```



Main Rack

Developers can use Rack, Shelf, Slot and other carrier elements to describe many telecom equipment structures. To specify the location, size, image properties to render equipment rack, sub rack, transverse card, half size slot etc.

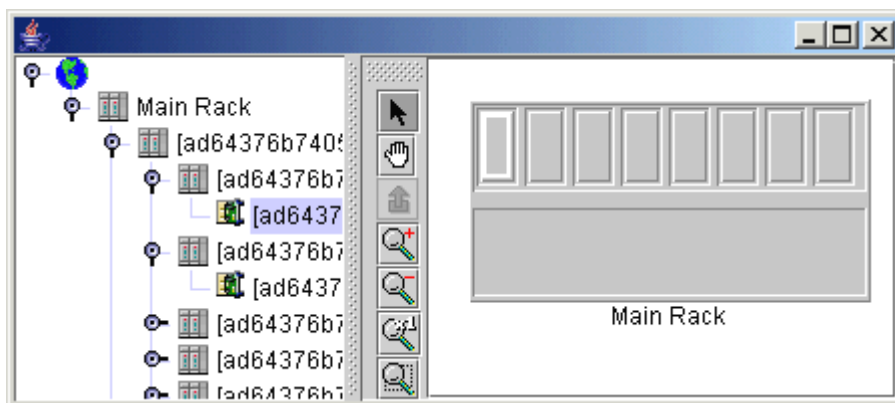
Card Element

Card element can be used to represent one equipment card item. Card is a functional physical and logical entry which can have much runtime information like alarms and events. A Card element can be in Rack, Shelf, Slot or other physical containers. You can also put it into a Chassis instance directly.

Create cards by API is very easy. For example:

```
for (int i = 0; i < 8; i++) {
    Slot slot = new Slot();
    slot.setLocation(24 + i * 24, 24);
    slot.setSize(20, 38);
    slot.setParent(shelf1);
    box.addElement(slot);

    Card card = new Card();
    card.setLocation(26 + i * 24, 26);
    card.setSize(16, 34);
    card.setParent(slot);
    box.addElement(card);
}
```



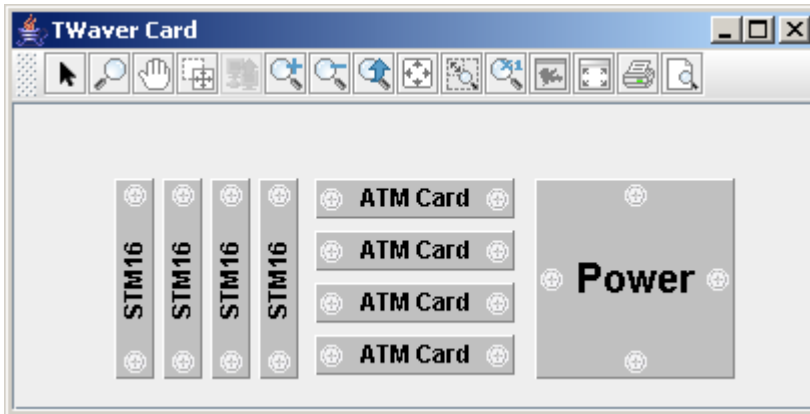
You can put a "bolt" on Card borders. Following codes show you how to use these client properties:

```
for (int i = 0; i < 4; i++) {
    Card card = new Card();
    card.setLocation(50 + i * 24, 100);
    card.setName("STM16");
    card.setSize(20, 100);
    card.putCardBoltTop(true);
    card.putCardBoltBottom(true);
    box.addElement(card);
}

for (int i = 0; i < 4; i++) {
    Card card = new Card();
    card.setLocation(150, 100 + i * 26);
    card.setName("ATM Card");
    card.setSize(100, 20);
    card.putLabelDirection(Direction.HORIZONTAL);
    card.putCardBoltLeft(true);
    card.putCardBoltRight(true);
    box.addElement(card);
}

Card card = new Card();
```

```
card.setLocation(260, 100);
card.setSize(100, 100);
card.setName("Power");
card.putLabelFont(new Font("Arial",Font.BOLD,20));
card.putLabelDirection(Direction.HORIZONTAL);
card.putCardBoltTop(true);
card.putCardBoltBottom(true);
card.putCardBoltLeft(true);
card.putCardBoltRight(true);
box.addElement(card);
```



Port Element

Port element can be used to render one equipment physical or logical port. Such as fiber, circuit, cable port, LED port etc. Normally Port elements are contained by Card element. You can put it to other equipment elements like Rack or Slot.

There are various telecom equipments and ports. Each of them may have different function and appearance. Instead of try to define all kinds of ports or other equipment unit, TWaver just provides a framework to developers. You can customize the image or other properties to represent different hardware units. In following codes, we specify the port different image to represent different port:

```
Card card = new Card();
card.setLocation(50, 100);
card.setSize(200, 60);
box.addElement(card);
for (int i = 0; i < 8; i++) {
    Port port = new Port();
    port.setImage("port_rj45f.png");
    port.setParent(card);
    port.setLocation(70+ i * 20, 110 );
    box.addElement(port);
}
for (int i = 0; i < 8; i++) {
    Port port = new Port();
    port.setImage("port_rj45f1.png");
    port.setParent(card);
    port.setLocation(70+ i * 20, 135 );
    box.addElement(port);
}
```



Extending Equipment Element

TWaver provides many equipment elements, but you may want to extend your own equipment element to represent your special equipment unit. Developers can extend their own equipment elements easily.

In this section we show user an example to explain how a new equipment element extends. Let's consider TimeSlot. In the service of T1 and E1, on TimeSlot are normally refer to one 64kbps channel. For the E1 port, it contains 32 TimeSlots. We hope that when user double click one E1 port, all TimeSlot inside can be displayed in a new subgraph. Each TimeSlot should render the reserved status.

We define a TimeSlot class which derived from BaseEquipment and extends several properties:

- Index: the index number. It will be displayed as label.
- Used: usage status. If it has been used the color should blue, otherwise green.
- Reserved: reserve status. If this TimeSlot was reserved, the color would gray.

TimeSlot.Java is as follows:

```
public class TimeSlot extends BaseEquipment {
    private static final Color NORMAL_COLOR = Color.green.darker();
    private static final Color USED_COLOR = Color.cyan.darker();
    private static final Color RESERVED_COLOR = Color.gray;
    private static final ImageIcon icon = TWaverUtil.getImageIcon("/resource/image/TWaver/timeslot.gif");
    private static final int DEFAULT_WIDTH = 20;
    private static final int DEFAULT_HEIGHT = 100;
    private boolean reserved = false;
    private boolean used = false;
    private int index=0;

    public TimeSlot() {
        super();
        init();
    }

    public TimeSlot(Object id) {
        super(id);
        init();
    }

    protected void init() {
        setReserved(false);
        setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
    }

    public ImageIcon getIcon() {
        return icon;
    }

    public void setReserved(boolean reserved) {
        this.reserved = reserved;
        updateColor();
    }

    public boolean isReserved() {
        return reserved;
    }

    public boolean isUsed() {
        return used;
    }

    public void setUsed(boolean used) {
        this.used = used;
    }
}
```

```

        updateColor();
    }

    private void updateColor() {
        if (reserved) {
            putBodyColor(RESERVED_COLOR);
        } else {
            if (used) {
                putBodyColor(USED_COLOR);
            } else {
                putBodyColor(NORMAL_COLOR);
            }
        }
    }

    public int getIndex() {
        return index;
    }

    public void setIndex(int index) {
        this.index = index;
        //set the index as name.
        this.setName("" + index);
    }
}

```

Now let write some code to check whether it works well:

```

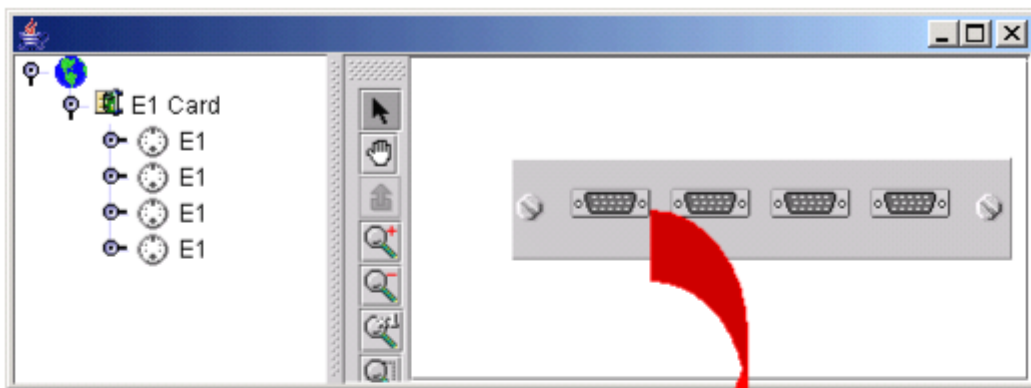
Card card =newCard();
card.setLocation(50, 50);
card.setSize(250, 50);
card.setName("E1 Card");
card.putLabelVisible(false);
card.putCardBoltLeft(true);
card.putCardBoltRight(true);
box.addElement(card);
for(int i = 0; i < 4; i++) {
    Port port =newPort();
    port.setImage("c:/temp/e1.png");
    port.setParent(card);
    port.setName("E1");
    port.putLabelVisible(false);
    port.setLocation(80 + i * 50, 65);
    box.addElement(port);

    Chassis chassis =newChassis();
    chassis.setParent(port);
    box.addElement(chassis);

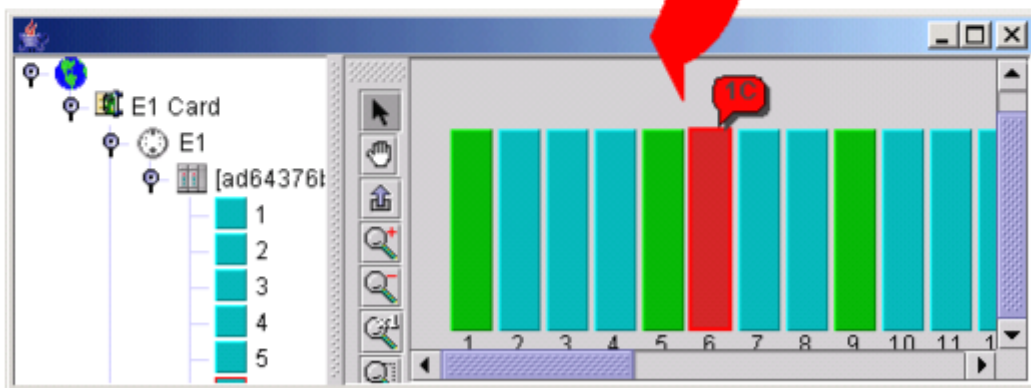
    for(int j = 0; j < 32; j++) {
        TimeSlot timeslot =newTimeSlot();
        timeslot.setParent(chassis);
        timeslot.setUsed(!(j % 4 == 0));
        timeslot.setReserved(j == 15);
        timeslot.setIndex(j + 1);
        timeslot.setLocation(20 + j * (timeslot.getWidth() + 4), 50);
        if(j == 5) {
            timeslot.getAlarmState().addNewAlarm(AlarmSeverity.CRITICAL);
        }
        box.addElement(timeslot);
    }
}

```

```
}
```



Chassis and Port



Double-Click Port, show TimeSlot graph

Using Follower Element

Follower is a special Node. Follower can have a 'host' Node, it can move along with its host, just like the Follower is part of the host Node. The Follower can be moved separately.

However you can make two Follower are hosted each other. In this case the two Followers will be always moved together, just like one Element.

Things can be more complicated. You can create a hosted queue, ring or tree. The following codes simulate these cases. Please note that the Link with arrow is created separately to show the host direction.

```
Follower node1 = new Follower();
node1.setLocation(100, 50);
box.addElement(node1);

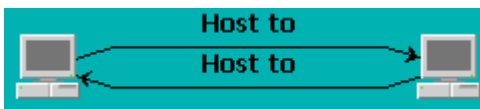
Follower node2 = new Follower();
node2.setLocation(300, 50);
box.addElement(node2);

node2.setHost(node1);
node1.setHost(node2);
```

This will look like:



The two Nodes are hosted each other (see the relationship below)



Host Pair

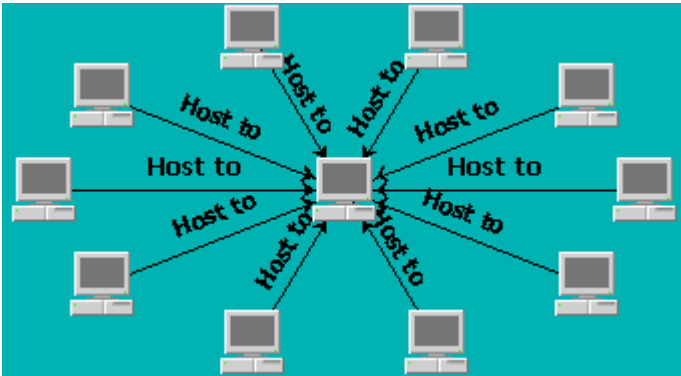


the line and text of these pictures does not exist in the actual example. It is only used to explain the relationship of these Nodes.

```
Follower center = new Follower();
center.setLocation(200, 200);
box.addElement(center);

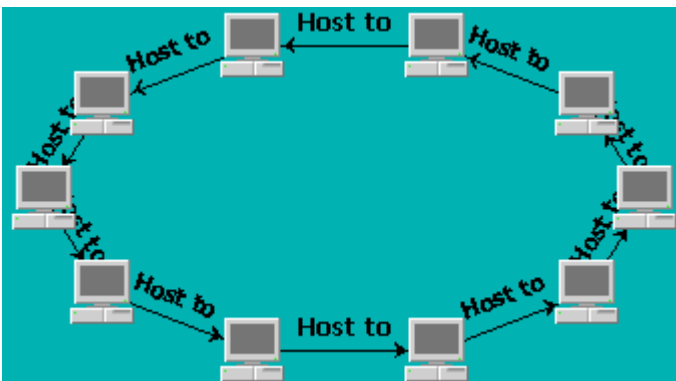
int count = 10;
for (int i = 0; i < count; i++) {
    Follower node = new Follower();
    int x = center.getLocation().x + (int) (150 * Math.cos(Math.PI * 2 / count * i));
    int y = center.getLocation().y + (int) (80 * Math.sin(Math.PI * 2 / count * i));
    node.setLocation(x, y);
    box.addElement(node);

    node.setHost(center);
}
```



Host Tree

```
int count = 10;
for (int i = 0; i < count; i++) {
    Follower node = new Follower("n" + i);
    int x = 200 + (int) (150 * Math.cos(Math.PI * 2 / count * i));
    int y = 400 + (int) (80 * Math.sin(Math.PI * 2 / count * i));
    node.setLocation(x, y);
    box.addElement(node);
    if (i > 0) {
        node.setHost((Follower) box.getElementByID("n" + (i - 1)));
    }
    if (i == count - 1) {
        (Follower) box.getElementByID("n0").setHost(node);
    }
}
```



Host Ring

```
int count = 5;
for (int i = 0; i < count; i++) {
    Follower node = new Follower("m" + i);
    int x = 100 + i * 80;
    int y = 500;
    node.setLocation(x, y);
    box.addElement(node);
    if (i > 0) {
        node.setHost((Follower) box.getElementByID("m" + (i - 1)));
    }
}
```

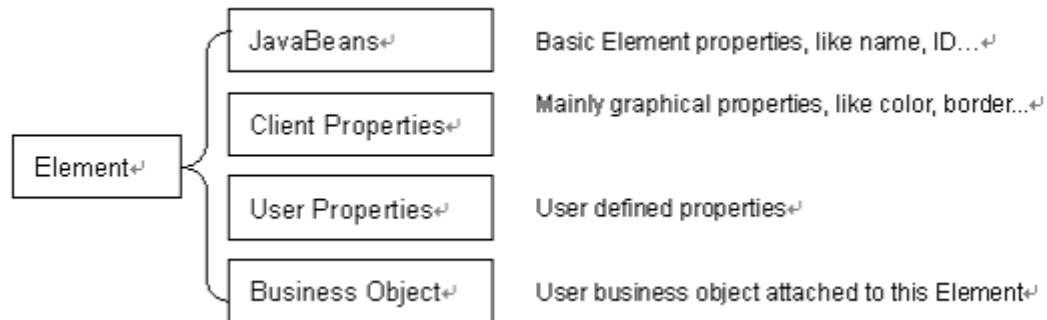


```
}
```



Host Queue

Using Element Properties



- [Using User Properties](#)
- [Using Business Object](#)

Using User Properties

In TWaver, the extended Element properties can be saved as ClientProperties or UserProperties. ClientProperties mainly used to save TWaver internal defined graphical properties, such as color, border etc. To void confliction with these TWaver defined properties, you should put all your defined extended properties in UserProperties because it is a place totally clean. If you want display all these UserProperties in Table or PropertySheet, use following code:

With API:

```
ElementAttribute attribute=new ElementAttribute();  
attribute.setUserPropertyKey("description");
```

With XML:

```
<attribute userPropertyKey="description"  
...  
>
```

Using Business Object

Business Object is a new added feature from TWaver 1.5.6. Business Object is an arbitrary object attached to TWaver Element as a business object with user properties on it. The properties on business object can be displayed on TWaver or PropertySheet components. Before, you can put a new property into ClientProperty or UserProperty of Element. However with this way, you have to iterate all properties of the server side object and put them into Element. Now you can directly attach the server side object on Element as a Business Object with code `element.setBusinessObject(businessObject)`. TWaver provide the ability to visit, manage and monitoring the properties of the business object.



The sense of Business Object is: you can make any server side object work with TWaver Element directly without converting objects or coping properties between business data and TWaver data.

To config a property on Business Object shown in Table or PropertySheet, use following way to config the XML:

With XML:

```
<attribute businessProperty="true"
...
/>
```

With API:

```
attribute=new ElementAttribute();
...
attribute.setBusinessProperty(true);
```

Please note business object can be Any Java Object. Below is a good example:

```
public class BusinessObject {
    private Object javaBeanPropertyA;
    protected Map clientProperties = new LinkedHashMap();

    public void putClientProperty(Object key, Object value) {
        clientProperties.put(key, value);
    }
    public Object getClientProperty(Object key){
        return clientProperties.get(key);
    }
    public Object getJavaBeanPropertyA() {
        return javaBeanPropertyA;
    }
    public void setJavaBeanPropertyA(Object javaBeanPropertyA) {
        this.javaBeanPropertyA = javaBeanPropertyA;
    }
}
```

By default, TWaver will use standard JavaBean methods or `get/putClientProperty` to visit the properties. Developers can also use following way to tell the read/write methods:

```
ElementAttribute.setReadMethod(String readMethod)
```

```
ElementAttribute.setWriteMethod(String writeMethod)
```

Following way can change the read/write methods globally:

```
TWaverUtil.setBocpReadMethod(Method bocpReadMethod)
TWaverUtil.setBocpWriteMethod(Method bocpWriteMethod)
```

This example use MyObject as the business object connects to Element, and use TElementTable display its properties:

```
public class BusinessObjectTest {
    public static void main(String[] args) {
        TDataBox box=new TDataBox();
        TElementTable table=new TElementTable(box);
        table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

        List attributes=new ArrayList();
        //TWaver will get 'ID' of Element by default,
        //because it is not a Business Object property
        ElementAttribute attribute=new ElementAttribute();
        attribute.setName("ID");
        attribute.setDisplayName("ID");
        attributes.add(attribute);

        //Define a new property, tells TWaver it is a Business Object property,
        //So TWaver will get it from Business Object
        attribute=new ElementAttribute();
        attribute.setBusinessProperty(true);
        attribute.setName("salary");
        attribute.setDisplayName("Salary");
        attributes.add(attribute);
        attribute=new ElementAttribute();
        attribute.setBusinessProperty(true);
        attribute.setName("married");
        attribute.setDisplayName("Married");
        attributes.add(attribute);

        //Define a property of Business Object, and use a 'ClientProperty' key.
        //in this case, TWaver will get this property from business object
        //in this way:
        //element.getBusinessObject().getClientProperty("description")
        attribute=new ElementAttribute();
        attribute.setBusinessProperty(true);
        attribute.setClientPropertyKey("description");
        attribute.setDisplayName("Description");
        attributes.add(attribute);
        table.setElementClass(Node.class);
        table.registerElementClassAttributes(Node.class, attributes);

        for(int i=0;i<20;i++){
            Node element=new Node();
            MyObject bo=new MyObject();
            bo.setSalary((TWaverUtil.getRandomInt(10) + 2) * 1000);
            bo.setMarried(TWaverUtil.getRandomBool());
            bo.putClientProperty("description",""+element.getID() + " with monthly salary of $" + bo.getSalary());
            //set business object
            element.setBusinessObject(bo);
            box.addElement(element);
        }
        JFrame frame = new JFrame("Business object demo");
```

```

frame.getContentPane().add(new JScrollPane(table));
frame.setSize(800, 600);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
TWaverUtil.centerWindow(frame);
frame.setVisible(true);
}
public static class MyObject {
    private double salary;
    private boolean married;
    protected Map clientProperties = new LinkedHashMap();

    public void putClientProperty(Object key, Object value) {
        clientProperties.put(key, value);
    }
    public Object getClientProperty(Object key){
        return clientProperties.get(key);
    }
    public boolean isMarried() {
        return married;
    }
    public void setMarried(boolean married) {
        this.married = married;
    }
    public double getSalary() {
        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }
}
}

```

Run this example:

ID	Salary	Married	Description
56d35d2f221c8d0e...	4000.0	<input checked="" type="checkbox"/>	'56d35d2f221c8d0e...
56d35d2f221c8d0e...	10000.0	<input checked="" type="checkbox"/>	'56d35d2f221c8d0e...
56d35d2f221c8d0e...	10000.0	<input type="checkbox"/>	'56d35d2f221c8d0e...
56d35d2f221c8d0e...	3000.0	<input checked="" type="checkbox"/>	'56d35d2f221c8d0e...
56d35d2f221c8d0e...	4000.0	<input type="checkbox"/>	'56d35d2f221c8d0e...
56d35d2f221c8d0e...	11000.0	<input type="checkbox"/>	'56d35d2f221c8d0e...
56d35d2f221c8d0e...	3000.0	<input type="checkbox"/>	'56d35d2f221c8d0e...
56d35d2f221c8d0e...	8000.0	<input checked="" type="checkbox"/>	'56d35d2f221c8d0e...

Element Self-Copy

Element Self-Copy

- [Element Self-Copy Overview](#)
- [Using Interceptor](#)

Element Self-Copy Overview

TWaver Element defines self-copy methods to support element copy. Use these copy methods to make a field-for-field copy of instances of that class.

```
//copy a new instance with a auto-gen id.
public Element copy();
//copy a new instance with given id.
public Element copy(Object id);
//copy a new instance and put it into the given DataBox.
public Element copy(TDataBox box);
//copy a new instance with given id, and put it into the given DataBox
public Element copy(Object id , TDataBox box);
```

For Link, TWaver provides extra copy methods for convenience.

```
//copy a new Link with specified from Node and end Node
public Element copy(Node from, Node to);
//copy a new Link with specified id, from Node and end Node
public Element copy(Object id, Node from, Node to)
```

Following code show you how to methods copy and exportValues work together. Understanding this will greatly help you how to override them to control the whole copy process:

```
public Element copy(TDataBox box){
    try {
        Element newElement = (Element)this.getClass().newInstance();
        this.exportValues(newElement, box);
        return newElement;
    }catch (Exception ex) {
        ex.printStackTrace();
        return null;
    }
}
```

```
protected void exportValues(Element element, TDataBox box) {
    ExportValuesInterceptor interceptor = TWaverUtil.getExportValuesInterceptor();
    interceptor.exportValues(this, element, box);
}
```


Using Interceptor

In this section, you will learn how to use `ExportValuesInterceptor` and the default value interceptor to customize the Element copy process.

When you copy a new Element, you may need to control which value is need to copy and how to copy a complicated value. TWaver allow you set a interceptor to control the copy process. You can use method `TWaverUtil.setExportValuesInterceptor` to set your interceptor, you can also use the default interceptor in `TWaver DefaultExportValuesInterceptor`.

The default interceptor implementation `DefaultExportValuesInterceptor` can copy all predefined JavaBean values and `ClientProperty` values in a default copy rule. You can change the default copy rule by method `registerPredefinedPropertyCopied`:

```
TUIManager.registerPredefinedPropertyCopied(String propertyName, boolean copyOrNot)
```

Normally you just extends class `DefaultExportValuesInterceptor` and override method `exportValues` to change the default copy rule.

For example, `twaver.Follower` has a property "host", it is a `Node` value. By default, TWaver only copy the reference of this `Node`:

```
targetElement.setHost(sourceElement.getHost());
```

If you want make a deeply copy for the host `Node`, and set it to the new copied Element as host, you can use below code:

```
public static class MyExportValuesInterceptor extends DefaultExportValuesInterceptor {
    public void exportValues(Element source, Element target, TDataBox box) {
        if(box!=null&&target!=null){
            box.addElement(target);
        }
        super.exportValues(source, target, box);
    }
    protected void exportHost(Follower source, Follower target, TDataBox box) {
        if (source.getHost() == null) {
            target.setHost(null);
        } else {
            Node newHost;
            try {
                newHost = (Node) (source.getHost().getClass().newInstance());
                if (box != null) {
                    box.addElement(newHost);
                }
                target.setHost(newHost);
            } catch (InstantiationException e) {
                e.printStackTrace();
            } catch (IllegalAccessException e) {
                e.printStackTrace();
            }
        }
    }
}
```

```
TDataBox box = new TDataBox();
Follower follower = new Follower();
Node host = new Node();
follower.setHost(host);
box.addElement(host);
box.addElement(follower);

TWaverUtil.setExportValuesInterceptor(new MyExportValuesInterceptor());
TUIManager.registerPredefinedPropertyCopied(TWaverConst.PROPERTYNAME_HOST, true);
TDataBox box2 = new TDataBox();
follower.copy(box2);
System.out.println(box2.getAllElements().size());
```

Using DataBox

In TWaver, all managed objects displayed on a telecom OSS GUI are contained and managed by a DataBox.

DataBox is a data container. You can consider DataBox as a more complex java collection or java object array. DataBox can handle all managed objects which instanceof TWaver Element interface. For details, see [Predefined Managed objects](#).

A DataBox is a collection of managed object of the class Element that notifies its listeners when an object is added, removed, cleared, layer changed or any property value changed. The DataBox can easily connect to back-end data in a large variety of formats. It can also connect to XML data streams. That is, you can use XML streams to add, remove objects or change some object's property value.

DataBox has been well designed to load and manage large scale network data. DataBox runs extremely fast and consumes little memory.

- [Changing Data of DataBox](#)
- [Visiting Data of DataBox](#)
- [Using ElementCallbackHandler Simplify Traversal DataBox](#)
- [Using DataBox Listener](#)
- [Using Element Property Change Listener](#)
- [Using SelectionChangeInterceptor](#)
- [Using DataBox LayerModel](#)
- [On-Demand-Load](#)
- [DataBox and Multi-Thread](#)
- [Customizing Alarm Propagator](#)
- [Using QuickFinder](#)
- [Undo and Redo Support](#)

Changing Data of DataBox

Add Element into DataBox:

```
box.addElement(Element element)
```

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.2_04" class="java.beans.XMLDecoder">
  <object id="Rack0" class="twaver.Node"/>
</java>
```

Remove Element from DataBox:

```
box.removeElement(Element element);
box.removeElementByID(Object elementID);
```

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.2_04" class="java.beans.XMLDecoder">
  <removeObject id="Rack0"/>
</java>
```

Clear DataBox:

```
box.clear();
```

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.2_04" class="java.beans.XMLDecoder">
  <clearObject/>
</java>
```

Change Element property:

```
box.addElement(node1);
...
node1.setLocation(100,100);//change node location value.
```

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.2_04" class="java.beans.XMLDecoder">
  <updateObject id="node1">
    <void property="location">
      <object class="java.awt.Point">
        <int>100</int>
```

```
<int>100</int>  
</object>  
</void>  
</updateObject>  
</java>
```

Visiting Data of DataBox

Method	Description
<code>box.getAllElements()</code>	Retrieve all elements of the DataBox
<code>box.getAllElementsReverse()</code>	Retrieve all elements of the DataBox in a reverse order
<code>box.getElementByID(Object id)</code>	Get element by ID
<code>box.getElementByName(String name)</code>	Get all elements with given name
<code>box.getElementByTag(String tag)</code>	Get all elements with given tag
<code>box.iterator()</code>	Get an Iterator of all data
<code>box.iterator(Class clazz)</code>	Get an Iterator of all specified type data
<code>box.iteratorReverse()</code>	Get an Iterator in a reverse order
<code>box.iteratorSelection(ElementCallbackHandler handler)</code>	Iterate all selected elements and pass each element into a handler to process
<code>box.iteratorReverse(ElementCallbackHandler handler)</code>	Iterate all elements in this box in a reverse order and pass each element to a handler to process
<code>box.iteratorReverseByLayer(ElementCallbackHandler handler)</code>	Iterate all element by layer reverse order, and pass each element to a handler to process
<code>box.iteratorByLayer(ElementCallbackHandler handler)</code>	Iterate all element by layer order and pass each element into a handler to process
<code>box.getRootElements()</code>	Get all top-level elements. A top-level element is an element without parent
<code>box.getRootElementsReverse()</code>	Get all top-level elements in reverse order. A top-level element is an element without parent
<code>box.depthFirstEnumeration()</code>	Get an enumeration visit all elements in depth-first order.
<code>box.depthFirstEnumeration(Element root)</code>	Get an enumeration visit all children elements in depth-first order.
<code>box.breadthFirstEnumeration()</code>	Get an enumeration visit all elements in breadth-first order.
<code>box.breadthFirstEnumeration(Element root)</code>	Get an enumeration visit all children elements in breadth-first order.



You can't remove the element from box iterator. The returned iterator has been encapsulated, an exception will be thrown if you try to remove element from the iterator. Use `removeElement` method of DataBox instead.

Using ElementCallbackHandler Simplify Traversal DataBox

The twaver.ElementCallbackHandler is used to process element when elements' iteration for data box.
twaver.ElementCallbackHandler

```
public interface ElementCallbackHandler {
    public boolean processElement(Element element);
}
```

It will be called in twaver.TDataBox like this

```
public void iterator(ElementCallbackHandler handler){
    Iterator it = elements.iterator();
    while(it.hasNext()){
        Element element = (Element)it.next();
        if(!handler.processElement(element)){
            return;
        }
    }
}
```

You can use it by follows methods

- twaver.TDataBox.iterator(ElementCallbackHandler)
- twaver.TDataBox.iteratorByLayer(ElementCallbackHandler)
- twaver.TDataBox.iteratorReverse(ElementCallbackHandler)
- twaver.TDataBox.iteratorReverseByLayer(ElementCallbackHandler)
- twaver.TDataBox.iteratorSelection(ElementCallbackHandler)

You can create a twaver.ElementCallbackHandler instance like this

```
ElementCallbackHandler handler=new ElementCallbackHandler(){
    int i=0;
    public boolean processElement(Element element) {
        element.setName("node_"+i++);
        if(i>10){
            //return false , end traversal
            return false;
        }
        return true;
    }
};
box.iterator(handler);
```

Use twaver. ElementReturnCallbackHandler for a return value
twaver. ElementReturnCallbackHandler

```
public interface ElementReturnCallbackHandler extends ElementCallbackHandler {
    public Object getReturnValue();
}
```

Use as follows

```
ElementReturnCallbackHandler handler=new ElementReturnCallbackHandler(){
    int i=0;
    private List list = new ArrayList();
    public boolean processElement(Element element) {
        element.setName("node_"+i++);
        list.add(element);
        if(i>10){
            return false;
        }
        return true;
    }
    public Object getReturnValue(){
        return list;
    }
};
box.iterator(handler);
List processedElements=handler.getReturnValue();
```


Using DataBox Listener

Use DataBoxListener interface to receive DataBox element change event. A DataBox event will be sent to DataBox listener on an element added, removed or layer changed. Listener can also receive event on the DataBox cleared.

Please note that DataBox listener is a coarse-grained interface used to detect element add, removed, cleared or layer changed. If you want detect property change event of each Element, please use another fine-grained interface at [Using Element Property Change Listener](#).

Below example show you how to monitor the DataBox element changing.

```
TDataBox box = new TDataBox();
box.addDataBoxListener(new DataBoxAdapter() {
    public void elementAdded(DataBoxEvent dataBoxEvent) {
        System.out.println(dataBoxEvent.getElement() + " added");
    }

    public void elementRemoved(DataBoxEvent dataBoxEvent) {
        System.out.println(dataBoxEvent.getElement() + " removed");
    }

    public void elementsCleared(DataBoxEvent dataBoxEvent) {
        System.out.println(dataBoxEvent.getDataBox() + " cleared");
    }
});
```

Using Element Property Change Listener

Once an Element was added into the DataBox, it will listen to the element property change event. You can hook the event by adding a `java.beans.PropertyChangeListener` instance.

Below example show you how to monitor the DataBox element property value change event.

```
TDataBox box = new TDataBox();
box.addElementPropertyChangeListener(new PropertyChangeListener(){
    public void propertyChange(PropertyChangeEvent e){
        Element element=(Element)e.getSource();

        System.out.println("Element:" +element.getID().toString());
        System.out.println("Property name:" +e.getPropertyName());
        System.out.println("Old value:" +e.getOldValue());
        System.out.println("New value:" +e.getNewValue());
    }
});
```

Using SelectionChangeInterceptor

SelectionChangeInterceptor is used to intercept the selection change events, the following two methods can be implemented to do something when selection is changed:

```
public void beforeSelectionChanged(TDataBox box, int type, Collection elements);
public void afterSelectionChanged(TDataBox box, int type, Collection elements);
```

As follows:

```
box.getSelectionModel().setSelectionChangedInterceptor(
    new SelectionChangedInterceptor(){
        public void beforeSelectionChanged(TDataBox box, int type, Collection elements){
            selectionArea.append(">> Before: " + getName(type) + " | Effected Count:" + elements.size());
        }
        public void afterSelectionChanged(TDataBox box, int type, Collection elements){
            selectionArea.append("<< After: " + getName(type) + " | Effected Count:" + elements.size());
            if(selectionArea.getLineCount() > 300){
                selectionArea.setText(null);
            }
        }
        public String getName(int type){
            if(type == SelectionChangedInterceptor.APPEND_SELECTION_COLLECTION){
                return "APPEND_SELECTION_COLLECTION";
            }
            if(type == SelectionChangedInterceptor.APPEND_SELECTION_ELEMENT){
                return "APPEND_SELECTION_ELEMENT";
            }
            if(type == SelectionChangedInterceptor.CLEAR_SELECTION){
                return "CLEAR_SELECTION";
            }
            if(type == SelectionChangedInterceptor.REMOVE_SELECTION_COLLECTION){
                return "REMOVE_SELECTION_COLLECTION";
            }
            if(type == SelectionChangedInterceptor.REMOVE_SELECTION_ELEMENT){
                return "REMOVE_SELECTION_ELEMENT";
            }
            return "***";
        }
    }
);
```

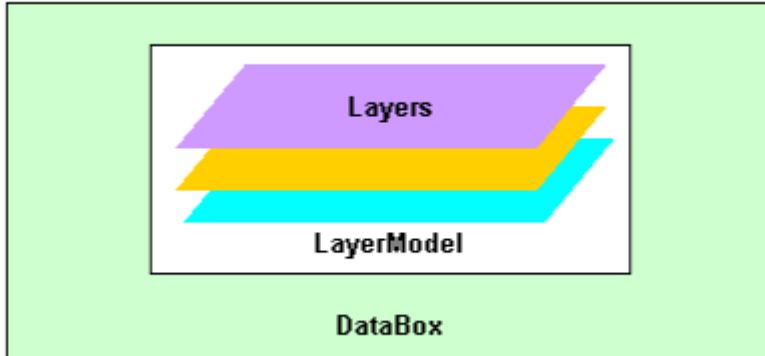
Using DataBox LayerModel

LayerModel is a container managing element layers.

- [LayerModel and Layer](#)
- [Managing Layer via LayerModel](#)
- [Using LayerModelListener](#)
- [Using Layer](#)
- [Element and Layer](#)
- [Serializing Layer to XML](#)
- [Layer on Network](#)
- [Using LayerManagerPane](#)

LayerModel and Layer

TWaver DataBox is a layered data container. Any element in DataBox should be in a layer. By default, all elements are located in the default layer, a predefined, irremovable layer. As a part of DataBox, LayerModel is the manager of all layers. You can access LayerModel by method TDataBox.getLayerModel.



Managing Layer via LayerModel

This is what you can do by LayerModel

- Add Layer (duplicate Layer ID will cause an exception)
- Remove Layer (remove default Layer will cause an exception)
- Clear Layer (exclude default Layer)
- Retrieve Layer (get, iterate)
- Move Layer (via moveTo*** methods)

Using LayerModelListener

You can monitor the change by installing a listener on LayerModel object:

- `addLayerModelListener`: add a layer model listener
- `removeLayerModelListener`: remove an installed layer model listener

With the installed listener, you will aware following change of LayerModel:

- Layer added (method `layerAdded` called back)
- Layer removed (method `layerRemoved` called back)
- Layer index changed (method `layerIndexChanged` called back)

Using Layer

- [Predefined properties](#)
- [Layer Client Properties](#)
- [Monitoring Layer Property Change Event](#)

Predefined properties

Layer object defines all properties for a network topology layer:

- id: the identifier of the layer, can not duplicate
- name: layer name
- visible: whether the layer (and elements on it) is visible
- selectable: whether elements on this layer is selectable by mouse
- movable: whether elements on this layer is movable by mouse
- resizable: whether elements on this layer is resizable by mouse
- alpha: the alpha value of the transparency
- description: description of this layer

Layer Client Properties

Layer object support ClientProperty facility. Following methods are supported to use ClientProperty:

- putClientProperty: put into Layer a new client property
- getClientProperty: get a client property from Layer object
- getClientProperties: get all client properties from this Layer object

Monitoring Layer Property Change Event

Layer is a standard JavaBean object. It support fire bound property change event mechanism. You can use method `getPropertyChangeSupport` to get the utility class that used by beans to fire value change event of the bound properties. The following definitions are used to indicate the Layer properties:

layer property	Defination
Id	Layer.PROPERTY_ID
Name	Layer.PROPERTY_NAME
description	Layer.PROPERTY_DESCRIPTION
Visible	Layer.PROPERTY_VISIBLE
Selectable	Layer.PROPERTY_SELECTABLE
Movable	Layer.PROPERTY_MOVABLE
Resizable	Layer.PROPERTY_RESIZABLE
Alpha	Layer.PROPERTY_ALPHA
client properties	TWaverConst.CLIENT_PROPERTY_PREFIX+key

Element and Layer

The Layer of an Element can be specified via method `setLayerID` and `getLayerID`. Once the Element enter into `DataBox`, TWaver will check the layered of this element and process it as following rules:

1. If no `LayerID` found, put it into default layer
2. If `LayerID` found, but the indicated Layer is not exist, put it into default layer
3. If `LayerID` found, and the indicated Layer is exist, put it into the Layer
4. If `LayerID` value changed, process it as above rules 1, 2 and 3
5. If new Layer added, related Elements will be moved to the Layer from default layer
6. If Layer removed, all related Elements will be moved to the default layer

Serializing Layer to XML

Just like element data contains in DataBox, Layer also can be serialized into XML. Following codes shows you how to create a Layer and serialize the Layer into XML:

```
TDataBox box=new TDataBox();
Layer layer=new Layer("layer1");
box.getLayerModel().addLayer(layer);

Node node=new Node("node1");
node.setLayerID("layer1");
box.addElement(node);

DataBoxOutputSetting setting=new DataBoxOutputSetting();
setting.setWithLayers(true);
box.output("c:/a.xml",setting);
```

The serialized XML will looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.5.0_03" class="java.beans.XMLDecoder">
  <object class="twaver.Node">
    <void property="layerID">
      <string>layer1</string>
    </void>
  </object>
  <object class="twaver.Layer">
    <null/>
    <void property="name">
      <string>Default Layer</string>
    </void>
  </object>
  <object class="twaver.Layer">
    <string>layer1</string>
  </object>
</java>
```

Also, DataBox can read Layer objects from XML as following rules:

1. If Layers in XML no conflict to the Layers in DataBox, all Layers will be saved by DataBox
2. If Layers in XML conflict to the Layers in DataBox, properties of duplicated Layers will be merged, and all Layers will be saved. When merge properties for a Layer, if property exist in both XML Layer and DataBox Layer, then the property of XML Layer will be used.

Layer on Network

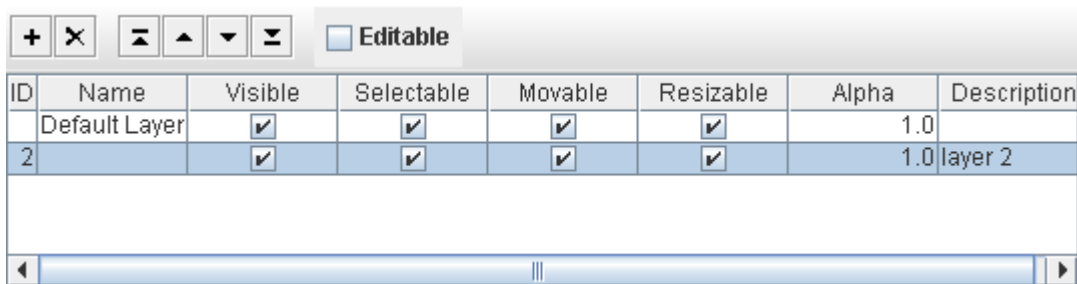
LayerModel gives each Layer an index. Network will paint all elements by the index order. The lower Layer will be paint on the network first and the upper Layer will be paint on the network later.

Using LayerManagerPane

LayerManagerPane is a predefined Swing panel. You can use it to manage layers. You can create LayerManagerPane by LayerModel, or by network.

```
LayerManagerPane pane=new LayerManagerPane(network);
```

Here is a quick look at LayerManagerPane. The toolbar provides you buttons to add, delete and move layers.



On-Demand-Load

Normally, all network datas are created by APIs and added into the DataBox. But sometimes you don't want to create all element instances when application starts. TWaver provides the ability to load data from anywhere only when we need to load it.

For instance, you may have 1000 routers on the network. Each router node has a chassis which contains racks, cards and ports. In order to speedup the startup time and save memory, the application can only create the 1000 router node and display it on network component. The chassis and all contained elements like rack, card and port will be created dynamically only when the equipment is drilled-down. Now you may need the on-demand-load ability of TWaver.

TWaver data can be created by API, XML or any customized data loader. We know how to create data by API from above chapters. You can check [Comparing XML and API](#) to learn more details. You can learn how to define a customized data loader in this section.

By default, TWaver uses a default data loader to load data, which is contained in TWaverUtil. You can get this instance by TWaverUtil.getDataLoader and specify a new data loader by TWaverUtil.setDataLoader. The default data loader will try to load data from an URL. It can be a HTTP URL or file name. But you can also define a new data loader to load data as your own way.

To illustrate this feature, we assume that there are three subnetworks on the network. Each subnetwork has a lot of children nodes. The children come from a file, XML, and database.

```
TDataBox box = new TDataBox();
//this subnetwork data comes from a file.
SubNetwork sub1=new SubNetwork();
sub1.setDataSource("file://myData.txt");
box.addElement(sub1);
//this subnetwork data comes from a socket connection.
SubNetwork sub2=new SubNetwork();
sub2.setDataSource("Socket://127.0.0.1:6789/ID=679");
box.addElement(sub2);
//this subnetwork data comes from a database SQL.
SubNetwork sub3=new SubNetwork();
sub3.setDataSource("DataBase://sql=select * from ElementTable");
box.addElement(sub3);
```

Then we define a new data loader to deal with the datasource of the subnetworks.

```
class MyDataLoader extends DefaultDataLoader{
    public boolean loadData(String dataSource, SubNetwork subNetwork, TDataBox box) {
        //deal with 'file' datasource.
        if(subNetwork.getDataSource().startsWith("file://")){
            return loadFromFile(dataSource, subNetwork, box);
        }
        //deal with 'socket' datasource.
        if(subNetwork.getDataSource().startsWith("Socket://")){
            return loadFromSocket(dataSource, subNetwork, box);
        }
        //deal with 'database' datasource.
        if(subNetwork.getDataSource().startsWith("DataBase://")){
            return loadFromDataBase(dataSource, subNetwork, box);
        }
        return false;
    }
}
```



```
//to load data from XML file, we can use the default data loader.
private boolean loadFromFile(String dataSource,SubNetwork subNetwork,TDataBox box){
    return super.loadData(subNetwork.getDataSource(), subNetwork, box);
}
```

```
//load data from socket
private loadFromSocket(String dataSource,SubNetwork subNetwork,TDataBox box){
//here we can use the URL to create socket connection,
//receive data, create element objects,
//set their parent as the subnetwork,
//and put them into the DataBox
...
}
```

```
//load data from database
private boolean loadFromDataBase(String dataSource,SubNetwork subNetwork,TDataBox box){
//here we can use the URL to create database connection,
//use the sql to query data from table, create element objects,
//set their parent as the subnetwork,
//and put them into the DataBox
...
}
```

Finally we set this data loader as the default TWaver data loader:

```
MyDataLoader loader = new MyDataLoader();
TWaverUtil.setDataLoader(loader);
```

Also On-Demand-Load can be realized by rewriting these two methods:

```
twaver.TDataBox
public void loadSubNetwork(TSubNetwork subNetwork)
public void loadSubNetwork(TSubNetwork subNetwork, Component component)
```

DataBox and Multi-Thread

DataBox and Multi-Thread

DataBox is not thread-safe. You should use `SwingUtilities.invokeLater` or `SwingUtilities.invokeAndWait` to operate DataBox in Swing Event-Dispatching-Thread (EDT). See following code:

```
TDataBox box=new TDataBox();
Thread thread =new Thread(){
public void run(){
//long task here
...
box.addElement(new Node()); //wrong
```

```
TDataBox box=new TDataBox();
Thread thread =new Thread(){
public void run(){
//long task here
...
SwingUtilities.invokeLater(new Runnable(){
public void run(){
box.addElement(new Node()); //right
}
});
}
}
```

Please see JDK documents [at here](#) for more details about how to use Thread.

Customizing Alarm Propagator

TWaver provide a default alarm propagator for developer. The default propagator propagates the most severe alarm severity to its parent Element. In fact you can define a new alarm propagator with different propagate rules.

For example, we can define a new alarm propagator which onlyl propagate critical alarms to parent and ignore other severities. We can extend a new class from TWaver default alarm propagator 'SummingAlarmPropagator' and override its method to change the propagate rule:

```
class CriticalAlarmPropagator extends SummingAlarmPropagator{
    protected void propagateToParent(Element element, Element parent){
        //figure out parent's highest severity
        AlarmSeverity result = null;
        Iterator it = parent.children();
        while (it.hasNext()) {
            Element child = (Element) it.next();
            AlarmSeverity childHighest = child.getAlarmState().getHighestNewAlarmSeverity();
            if (childHighest == AlarmSeverity.CRITICAL) {
                result = AlarmSeverity.CRITICAL;
                break;
            }
            if(child.getAlarmState().getPropagateSeverity() == AlarmSeverity.CRITICAL){
                result = AlarmSeverity.CRITICAL;
                break;
            }
        }
        // propagate this result to parent.
        parent.getAlarmState().setPropagateSeverity(result);
    }
}
//use the propagator.
box.setAlarmPropagator(new CriticalAlarmPropagator());
```

Using QuickFinder

QuickFinder will help you create an index on the property which you are interested in. With the finder you can quickly get a list filled with the elements you want to get by the specified property value or the last element of the list.

In TWaver, there are two types of finders. One is used to index client property, and the other is used to index bean property. The following codes will show you how to use these two kinds of finders.

```
TDataBox box = new TDataBox ("Finder");
DataBoxQuickFinder finder = box.createClientPropertyFinder("age");
System.out.println (finder.find (new Integer (12)));
box.getElementByID ("B").putClientProperty ("age", new Integer (20));
System.out.println (finder.find (new Integer (20)));

box.removeElement (box.getElementByID ("D"));
System.out.println (finder.find (new Integer (15)));

Node node=null;
for(int i=0;i<50;i++) {
    node = new Node("guy"+i);
    node.setUserObject("user "+i);
    box.addElement(node);
}

finder = box.createJavaBeanFinder("userObject");

List list = finder.find("user 9");
Iterator iterator = list.iterator();
while(iterator.hasNext()) {
    node = (Node)iterator.next();
    System.out.println("the correct element ID is :"+node.getID());
}
```

Undo and Redo Support


Undo and Redo Support

TWaver DataBox allows developers to provide support for undo/redo in applications via following methods:

```
box.getUndoRedoManager().undo(); //undo
box.getUndoRedoManager().redo(); //redo
```

TWaver defines buttons for undo/redo in Network toolbar:

- `twaver.network.toolbar.NetworkUndoButton`
- `twaver.network.toolbar.NetworkRedoButton`

TWaver Network toolbar factory can creates toolbar with a set of buttons. The predefined "editor toolbar" will contain undo/redo buttons: 


You can also use APIs to enable the undo/redo with following code:

```
//use edit toolbar
network.setToolBar(NetworkToolBarFactory.getToolBar(TWaverConst.EDITOR_TOOLBAR, network));
//set the limit of undo/redo steps
box.getUndoRedoManager().setLimit(200);
//enable the keyboard for undo/redo ("ctrl+z" , "ctrl+y")
network.setEnableUndoRedoWithKeyboard(true);
```

- [Using UndoRedoManager](#)
- [Using UndoRedoListener](#)
- [Using UndoRedoInterceptor](#)

Using UndoRedoManager

Class UndoRedoManager is the manager which controls undo/redo action for a DataBox. It works with DataBox and provides a set of APIs for developers to control the undo/redo:

 UndoRedoManager
<ul style="list-style-type: none"> • <<create>> UndoRedoManager(in box: TDataBox) • getDataBox(): TDataBox • addUndoRedoListener(in l: UndoRedoListener): void • removeUndoRedoListener(in l: UndoRedoListener): void • getUndoRedoListeners(): List • reset(): void • canUndo(): boolean • canRedo(): boolean • undo(): void • redo(): void • getLimit(): int • setLimit(in limit: int): void • getPosition(): int • getInterceptor(): UndoRedoInterceptor • setInterceptor(in interceptor: UndoRedoInterceptor): void • isUndoRedoing(): boolean • isEnabled(): boolean • setEnable(in enable: boolean): void • addEvent(in event: Object): void • getEvents(): LinkedList

Some methods are explained here:

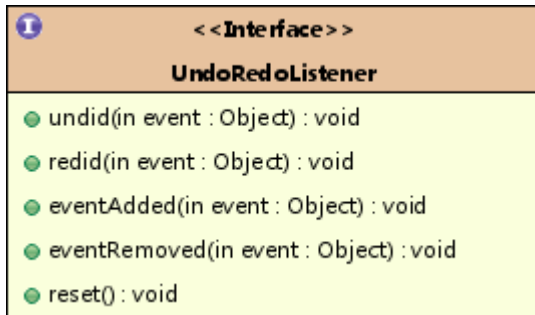
```
//Set undo/redo step limit. The default value is -1, that is, no step limit.
public void setLimit(int limit)
//Set the undo/redo interceptor. The default undo/redo interceptor is DefaultUndoRedoInterceptor.
//Use this method to change the default behaviour of the undo/redo.
//This is not recommended unless you know what are you doing.
public void setInterceptor(UndoRedoInterceptor interceptor)
//Whether enable undo
public boolean canUndo()
//Whether enable redo
public boolean canRedo()
//Add a undo/redo listener. When any undo/redo action performed, the listener will be invoked
public void addUndoRedoListener(UndoRedoListener l)
//Add a undo/redo event, including PropertyChangeEvent,
//DataBoxEvent and other TWaver internal undo/redo events.
public void addEvent(Object event)
//Reset undo/redo manager, all recorded steps will be cleared.
//This will be invoked when the DataBox is cleared.
public void reset()
```

Using in dataBox:

```
dataBox.getUndoRedoManager().setLimit(100);
```

Using UndoRedoListener

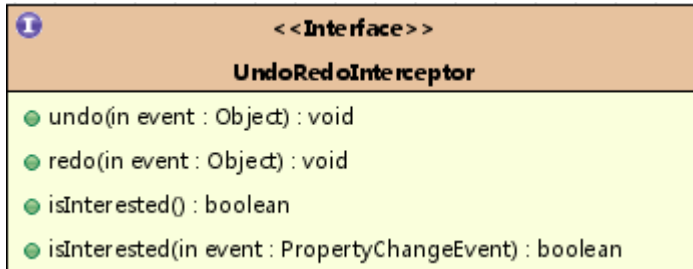
Class UndoRedoListener used to monitoring the actions of UndoRedoManager. When undo/redo actions performed, it will be called back:



- undid : called when action UndoRedoManager.undo() performed
- redid : called when action UndoRedoManager.redo() performed
- eventAdded : called when UndoRedoManager.eventAdded(event) performed
- eventRemoved : called when UndoRedoManager.eventRemoved(event) performed
- reset : called when UndoRedoManager.reset() performed

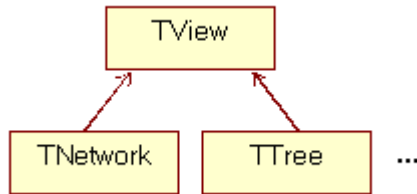
Using UndoRedoInterceptor

Use interceptor you can intercept the undo/redo action and change to your customized action instead. TWaver inside also uses a default interceptor to implement the default undo/redo actions (twaver.DefaultUndoRedoInterceptor).



Using Components

TWaver provides several Swing-Based components to display the business data contained in DataBox. All these components act as the View of MVC architecture. All TWaver components implement the interface TView and provide coincident operations for developers.



TView interface provides numerous options that enable you to fine-tune control to fit your application's needs. To make optimum use of these options, please see the javadoc of twaver.TView class.

- [TWaver Resources Redirection](#)
- [TWaver Task Scheduler](#)
- [Components Comment Features](#)
- [Using Network Component](#)
- [Using Tree Component](#)
- [Using Table Component](#)
- [Using Element Table Component](#)
- [Using TreeTable Component](#)
- [Using Alarm Table Component](#)
- [Using AlarmOverview Component](#)
- [Using List Component](#)
- [Using PropertySheet Component](#)
- [Using Chart Component](#)

TWaver Resources Redirection

TWaver resources include all images, icons, XML files and i18n property files. TWaver use a class as the resource agent to load them. By default TWaver will load all resources from TWaver.jar/resource package.

You can change the default resources location of TWaver. To do that, you need to set a new resource agent class with TWaverUtil.setResourceAgent(Class) method. If new resource agent was set, TWaver will try to load the resources from the new location. If resource was not found, then TWaver use the default resource.

By this mechanism, you can redefine all resources of TWaver used to create a brand new look-and-feel. You can also redefine part of them to customize the default style, such as change a default icon, text, or visible properties of an element.

The following table lists all kinds of the resource relative path:

Resource	Path	
	TWaver Default Path	Customized Path
image	TWaver.jar/resources/image/	—
i18n	TWaver.jar/resources/i18n/	ResourceAgent.class/i18n/
bean info	TWaver.jar/resources/bean/	ResourceAgent.class/bean/

TWaver Task Scheduler

- [Scheduler and Task](#)
- [Swing and Threads](#)

Scheduler and Task

In order to manage all periodic and lightweight tasks, TWaver provides task scheduler and task definition. You can define a new task and register the task into task scheduler to execute it. All codes defined in a task while be executed in the event-dispatching thread. That means you can handle all TWaver components safely in a TWaver task.

```
//define a cyclical task with 1 second interval.
Task task=new TaskAdapter(){
    public void run(long clock) {
        //do something here.
    }
    public int getInterval() {
        return 1000;
    }
}
TaskScheduler.getInstance().register(task);
```

Please note that an only lightweight task is properly to be executed by task scheduler. Heavyweight task will block the event-dispatching thread.
Predefined TaskScheduler

Two predefined TaskScheduler work inside TWaver:

1. This scheduler controls the alarm blinking, link flowing, and all tasks registered by developers via method TaskScheduler.getInstance().register(task). Default minimum time interval of this scheduler is 500ms.
2. This scheduler controls the animate GIF image painting. Default minimum time interval of this scheduler is 50ms.

You can see the schedulers above all have default minimum time interval. If the interval in your task is smaller than the scheduler's, use following method to make an adjustment:

```
public static void setAnimateBlinkInterval(int interval)
public static void setAnimateGIFInterval(int interval)
```



- Please be very careful when adjust scheduler A because this interval will affect the whole system on data blinking, link flowing, customer defined tasks, so it will affect the performance of your application.
- Adjust the interval of the schedulers property may improve the performance of TWaver and your application. You can even disable animation for a network (include blinking, flowing, animate GIF) by network.setEnableAnimation(false) and does not affect the customer registered tasks.

Swing and Threads

If your program creates and refers to its GUI the right way, you might not need to worry about threads. However, if your program creates threads to perform tasks that affect the GUI, or if it manipulates the already-visible GUI in response to anything but a standard event, then you should know the single-thread rule is as follows:

Once a Swing component has been realized, all code that might affect or depend on the state of that component should be executed in the event-dispatching thread.

You can find thousands of articles that discuss how to use `SwingUtilities.invokeLater` & `invokeAndWait` methods to execute tasks in the event-dispatching thread.

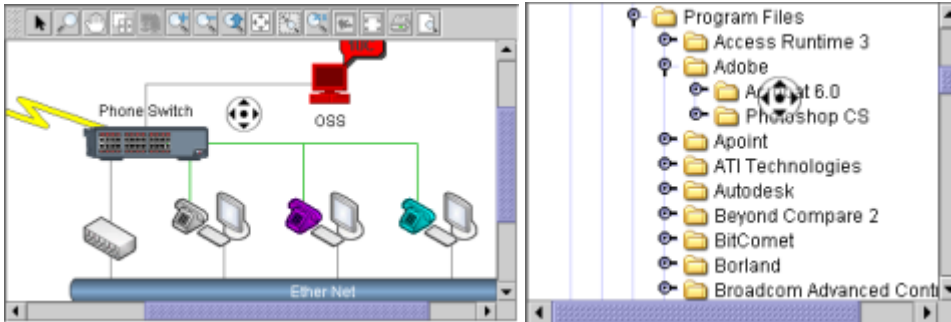
Components Comment Features

- [AutoScroll](#)
- [Exporting Images](#)
- [Popup Component](#)

AutoScroll

All TWaver components can be automatically scrolling. The method 'TWaverUtil.setEnableAutoScroll(true)' is provided to enable this feature.

With automatically scroll enabled, press mouse middle button to start automatically scroll, move mouse cursor to pan the component.



Exporting Images

TWaver can export images for TWaver components or any Swing components. Class TWaverUtil provides the method to export images for Swing components, also can specify the image bounds, scale and other parameters:

```
public static boolean exportImage(Component component,
String fileName,
String formatName,
double zoom,
Rectangle logicalBounds)
```

For TNetwork, this method is provided to export images dividually.

```
public static boolean exportDividedImages(String fileName,
String format,
int rowCount,
int columnCount,
double zoom)
```

You can also click the button on Network toolbar to export network canvas into an image file:

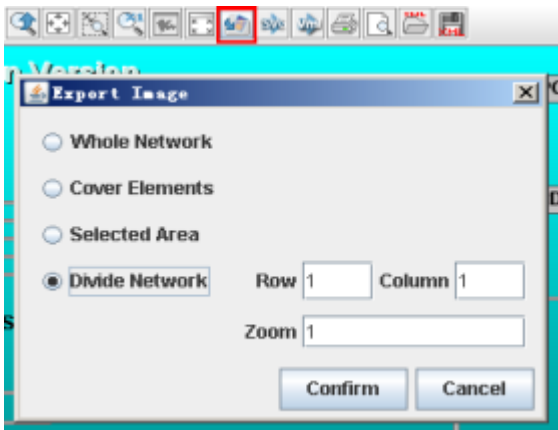
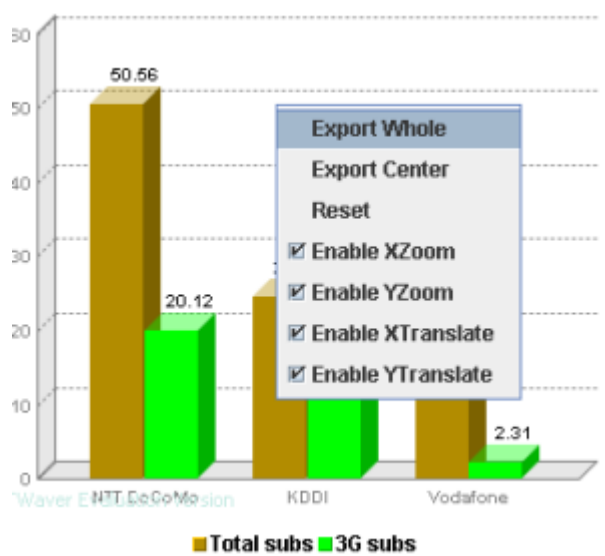


Chart Component provides another method to export the whole image or the center graphics image.

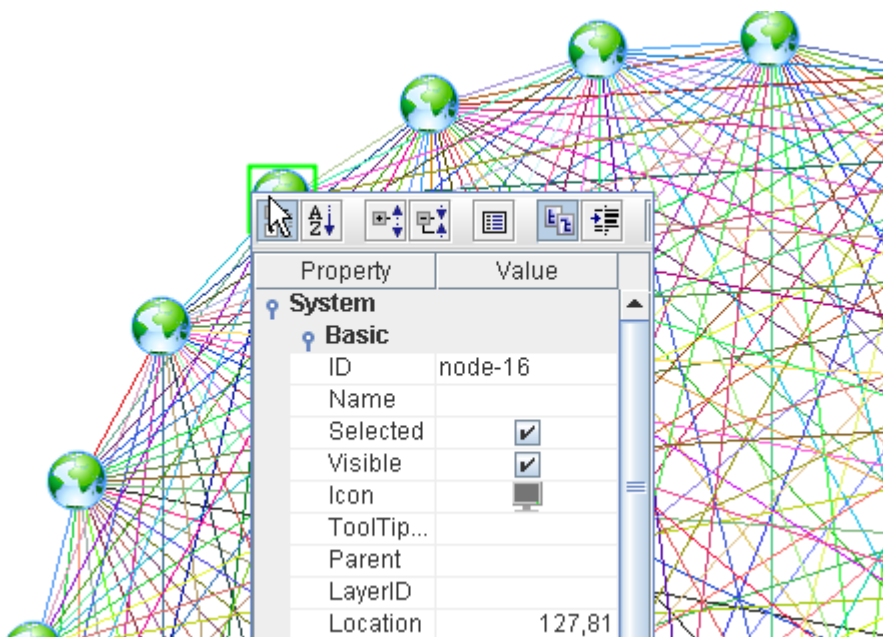
```
public static boolean exportImage(String fileName,
String formatName,
Boolean wholeChart),
```



Popup Component

TWaver components can popup menu, and also can popup any Swing component. Use `TWaverUtil.showPopupComponet` to popup component.

```
network.getCanvas().addMouseListener(new MouseAdapter() {
    public void mouseReleased(MouseEvent e) {
        if (e.isPopupTrigger()) {
            Element element = network.getElementPhysicalAt(e.getPoint());
            if (element != null) {
                TPropertySheet sheet = new TPropertySheet(box);
                sheet.setElementEditable(true);
                JScrollPane sp = new JScrollPane(sheet);
                sp.setPreferredSize(new Dimension(200, 300));
                box.getSelectionModel().setSelection(element);
                TWaverUtil.showPopupComponet(sp, network.getCanvas(), e.getPoint());
            }
        }
    }
});
```



Using Network Component

Network component is the major Swing component in TWaver product. This chapter introduces how to use TNetwork components to build your application.

- [Customizing Toolbar](#)
- [Network Component Hierarchy](#)
- [Network Component MVC Model](#)
- [Network Coordinate System](#)
- [Network Printing](#)
- [Network Representation](#)
- [Operations for Network](#)
- [Using Blinking Rule](#)
- [More Network Component Features](#)

Customizing Toolbar

The default toolbar of Network is displayed on the top position. It only contains the essential buttons like zoom, select and other. Developers can customize the toolbar buttons. You can add more predefined buttons and its order, or add new buttons.

You can visit toolbar by `TNetwork.getToolBar` and `TNetwork.setToolBar`. TWaver provides a toolbar factory class `NetworkToolBarFactory` to create toolbar easily. `NetworkToolBarFactory` accepts new button registered with a string key. After that, you can create new toolbar by giving a key string array. Following code shows you how to use `NetworkToolBarFactory` to create new toolbar:

Create simple toolbar:

```
network.setToolBar(NetworkToolBarFactory.getSimpleToolBar(network));
```



Create default Toolbar:

```
network.setToolBar(NetworkToolBarFactory.getDefaultToolBar(network));
```



Create editor Toolbar:

```
network.setToolBarByName(TWaverConst.EDITOR_TOOLBAR);
//or
network.setToolBar(NetworkToolBarFactory.getToolBar(TWaverConst.EDITOR_TOOLBAR, network));
```



Create toolbar with separator:

```
String[] ids = new String[] {
    "Selection",
    "$SEPARATOR",
    "ZoomIn",
    "ZoomOut",
    "ZoomToRectangle",
};
network.setToolBar(NetworkToolBarFactory.getToolBar(ids, network));
```



Create toolbar with customized button:

```
public class MyButton extends twaver.network.toolbar.BaseNetworkButton {
public class MyButton extends TWaver.network.toolbar.BaseNetworkButton {
    public MyButton(TNetwork network) {
        super(network);
        this.setText("My Button");
    }
}
```

```
NetworkToolBarFactory.registerButton("MyButton",MyButton.class);
\\
String[] ids = new String[] {
    "Selection",
    "$SEPARATOR",
    "MyButton",
    "$SEPARATOR",
    "ZoomIn",
    "ZoomOut",
    "ZoomToRectangle",
};
network.setToolBar(NetworkToolBarFactory.getToolBar(ids, network));
```



Here show you how to add a button into toolbar and add it into the default ButtonGroup:

```
JRadioButton myButton=new JRadioButton("MyButton");
JRadioButton myButton2=new JRadioButton("MyButton2");
//Get the default ButtonGroup
ButtonGroup group =(ButtonGroup)
network.getToolBar().getClientProperty(TWaverConst.TOOLBAR_BUTTON_GROUP_KEY_PREFIX
+TWaverConst.DEFAULT_BUTTON_GROUP_NAME);
//add it into the default button group
group.add(myButton);
group.add(myButton2);
//add it into the network toolbar
network.getToolBar().add(myButton);
network.getToolBar().add(myButton2);
```



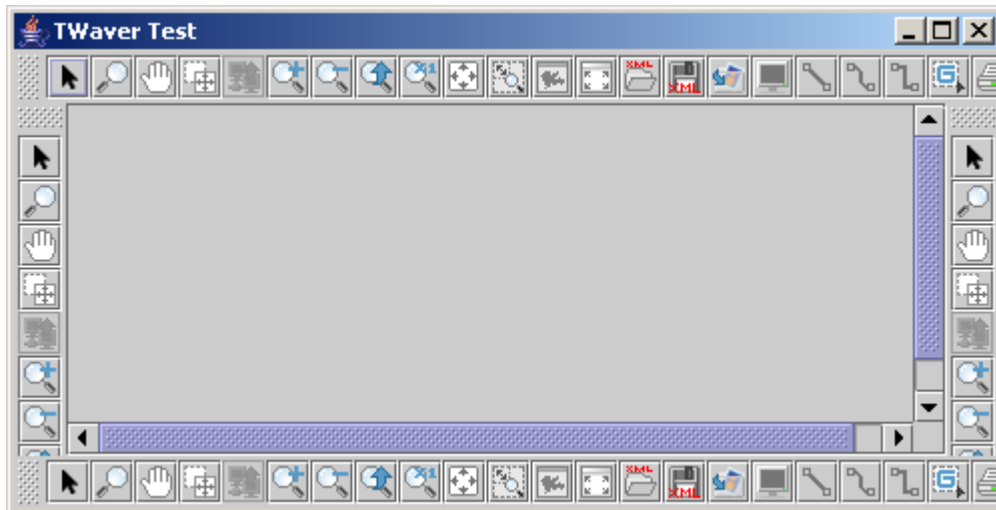
Create multiple toolbars:

```
JToolBar toolbar=NetworkToolBarFactory.getSimpleToolBar(network);
```










```





















toolbar.setOrientation(SwingConstants.VERTICAL);
network.add(toolbar, BorderLayout.WEST);
\\
toolbar=NetworkToolBarFactory.getSimpleToolBar(network);
toolbar.setOrientation(SwingConstants.VERTICAL);
network.add(toolbar, BorderLayout.EAST);
\\
toolbar=NetworkToolBarFactory.getDefaultToolBar(network);
network.add(toolbar, BorderLayout.NORTH);
\\
toolbar=NetworkToolBarFactory.getDefaultToolBar(network);
network.add(toolbar, BorderLayout.SOUTH);

```



Network component provides many predefined buttons used to operate network canvas like selection, zoom, pan etc. this table list all predefined buttons and its functions:

Button	Key	Icon	Usage
NetworkSelectButton	Selection		Selection mode
NetworkMagnifierButton	Magnifier		Network Magnifier
NetworkPanButton	Pan		Pan mode
NetworkUpButton	Up		To upper subnetwork
NetworkZoomInButton	ZoomIn		Zoom in
NetworkZoomOutButton	ZoomOut		Zoom out
NetworkZoomBackButton	ZoomBack		Zoom back
NetworkZoomResetButton	ZoomReset		Reset zoom
NetworkZoomToRectangleButton	ZoomToRectangle		Zoom to specified rectangle area

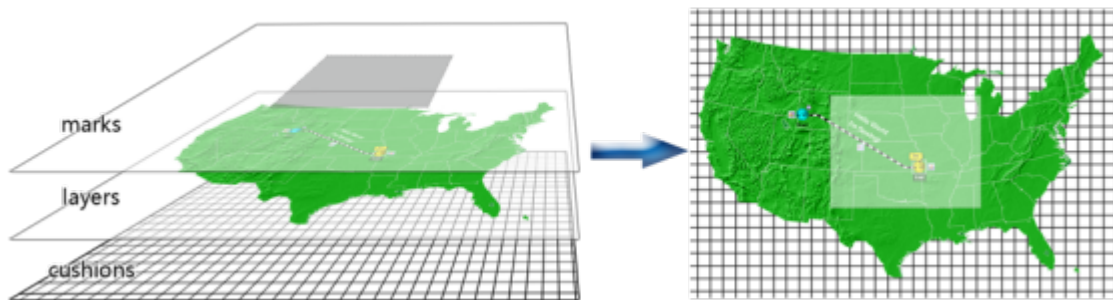
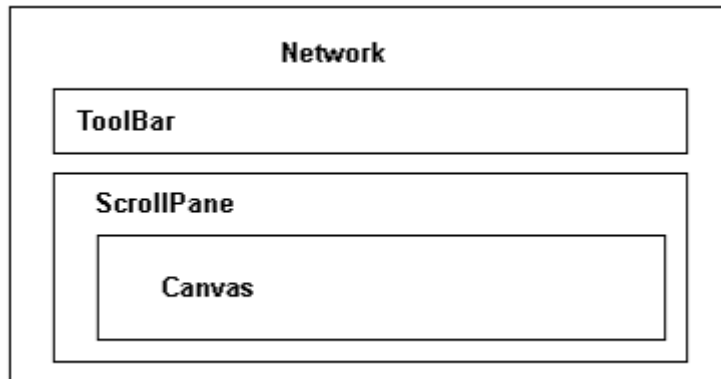
NetworkZoomToOverviewButton	ZoomToOverView		Zoom to Overview
NetworkOverViewButton	OverView		Turn on/off overview window
NetworkOpenDataFileButton	OpenDataFile		Open data file
NetworkSaveDataFileButton	SaveDataFile		Save data to file
NetworkExportImageButton	ExportImage		Export network canvas to image file
NetworkFullScreenButton	FullScreen		Network Full Screen
NetworkPrintButton	Print		Print network
NetworkPrintPreviewButton	PrintPreview		Print preview
NetworkCreateNodeButton	CreateNode		Create node
NetworkCreateLinkButton	CreateLink		Create link
NetworkCreateTextButton	CreateText		Create Text Element
NetworkUndoButton	Undo		Undo
NetworkRedoButton	Redo		Redo
NetworkCreateFlexionLinkButton	CreateFlexionLink		Create Flexion Link
NetworkCreateLinkSubNetworkButton	CreateLinkSubNetwork		Create Subnetwork Link
NetworkCreateOrthogonalLinkButton	CreateOrthogonalLink		Create Orthogonal Link
NetworkCreateShapeLinkButton	CreateShapeLink		Create shape link
NetworkCreateShapeNodeButton	CreateShapeNode		Create shape node
NetworkCreateShapeSubNetworkButton	CreateShapeSubNetwork		Create shape subnetwork
NetworkLazyMoveButton	LazyMove		Move elements lazily
	\$SEPARATOR		Button separator

Network Component Hierarchy

From the appearance, network contains following parts:

- A customizable toolbar
- A scrollable panel
- A canvas used to paint whole network map

The structure is as follows:



Network Canvas Hierarchy

- Cushion Layer: A layer displayed on the bottom, that is, painted first. It can paint any information "below the network data" on this layer. Please note Network displays in front of Cushion Layer as part of Data Layer.
- Data Layer: This layer displays all network business data. Elements layer, Attachments Layer, Alarm Layer etc. You can use LayerManager to manage the layer hierarchy.
- Marker Layer: A layer display extra information on top of the network.



Background Layer (Color Background, TextureBackground, Image Background etc) is under the Cushion Layer.

Adding Cushion layer

TNetwork

```
public void addCanvasCushion(CanvasCushion canvasCushion)
```

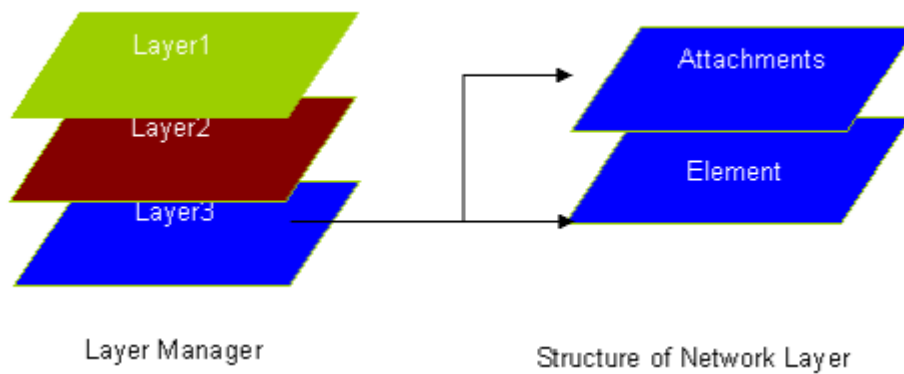
```
network.addCanvasCushion(new CanvasCushion(){
    public void paint(Graphics2D g2d) {...}
});
```

Adding MarkerLayer

```
TNetwork
public void addCanvasMarker (CanvasMarker m)
```

```
network.addCanvasMarker (new CanvasMarker (){
public void mark (Graphics2D g2d) {...}
});
```

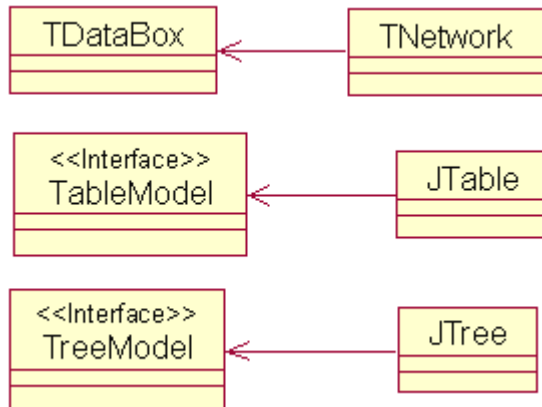
More information about management of Network Data Layer please check [Using DataBox LayerModel](#).



Network Component MVC Model

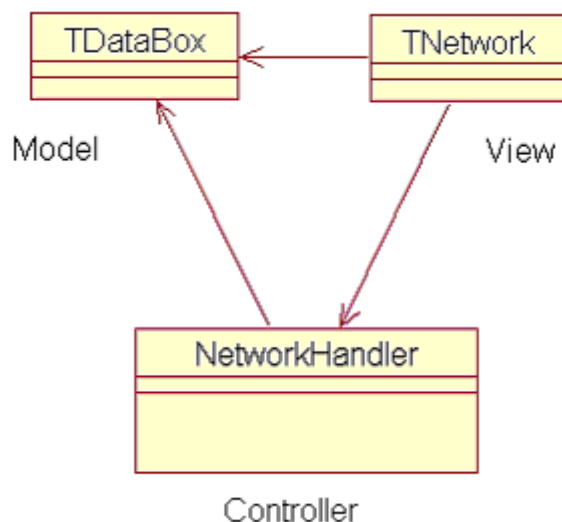
Network component derived from Swing JComponent. You can create and use it as a Swing component.

To make network component working, you should set the DataBox of the network component. All network datas are contained by DataBox container. According to the MVC architecture, DataBox acts as the model and components act as the view. This is similar to Swing TableModel or TreeModel:



The data in DataBox (TDataBox) can be shared by multi components. For example, many network components share the same DataBox, or network component and tree component share the same DataBox instance. Sharing data can make sure all components display all data consistently.

Considering to MVC architecture, network component acts as view, DataBox acts as model, and NetworkHandler acts as controller:



MVC architecture of network component

Network Coordinate System

TWaver Network component use Java Graphics Coordinate System which start from left-top corner as (0,0). X axis start from left to right, Y axis start from top to bottom. It does not support negative number. The elements beyond coordinate system will invisible or partial invisible.

Furthermore, TWaver Network support physical coordinate and logical coordinate. Physical coordinate is the actual physical location of your screen. Logical coordinate is the location of an element.

When you zoom in or zoom out network canvas, the location value of an element will never be changed. But for your screen, the element looks move to more right-bottom corner. In this case, if you want get an element at a point, you have to consider what coordination needs to use.



For example:

A node with location (100,100) added into a Network and you zoom in the network in some scale and the element displays in (300,300) of your screen. To get the element by location, you have two ways:

- `TNetwork.getElementPhysicalAt(300,300)`
- `TNetwork.getElementLogicalAt(100,100)`

To know more information, please check following parts:

- [Getting Elements By Location](#)
- [Network Viewport & Other Methods](#)

Getting Elements By Location

Normally we get element data from DataBox. However you can also get element from Network canvas by location. Network provides following methods to manipulate elements displayed on network canvas.

- List `twaver.network.TNetwork.getElementsAt(Rectangle rect, boolean isIntersectSelection)`

Gets all visible elements in specified rectangle.

Parameters: `rect` the conditional rectangle `isIntersectSelection` True if select all intersected elements, false select all contained elements.

Returns: all elements contained in this rectangle.

- Element `twaver.network.TNetwork.getElementLogicalAt(Point2D p)`

Gets the first element contains specified point.

Parameters: `p` a logical location

Returns: the element located at the specified logical location Null if nothing there.

- Element `twaver.network.TNetwork.getElementLogicalAt(int x, int y)`

Gets the first element contains specified point.

Parameters: `x` logical x value `y` logical y value

Returns: the element located at the specified logical location. Null if nothing there.

- List `twaver.network.TNetwork.getElementsLogicalAt(Point p)`

Gets all elements located at the given logical point. Logical point is come from physical point and network scale.

Parameters: `p` the given logical point.

Returns: all elements there

- List `twaver.network.TNetwork.getElementsLogicalAt(int x, int y)`

Gets all elements located at the given logical point. Logical point is come from physical point and network scale.

Parameters: `x` logical x value `y` logical y value

Returns: all elements there.

- Element `twaver.network.TNetwork.getSelectedElementLogicalAt(int x, int y)`

Gets the first selected element at the logical position. Logical point is come from physical point and network scale. Parameters: `x` the x coordinate x logical value `y` the y coordinate. y logical value Returns: the result element the first selected element there, null if nothing there

- Element `twaver.network.TNetwork.getSelectedElementPhysicalAt(Point p)`

Gets the first selected element at the given location. Physical point is refer to the absolute screen pixel location.

Parameters: `p` the given location

Returns: the first selected element there

- Element `twaver.network.TNetwork.getSelectedElementPhysicalAt(int x, int y)`

Gets the first selected element at the given location. Physical point is refer to the absolute screen pixel location.

Parameters: `xx` location value `yy` location value

Returns: the first selected element there

- Element `twaver.network.TNetwork.getElementPhysicalAt(Point p)`

Gets element located given physical location. Physical point is refer to the absolute screen pixel location.

Parameters: `p` the given physical location

Returns: the element located there

- Element `twaver.network.TNetwork.getElementPhysicalAt(int x, int y)`

Gets element located given physical location. Physical point is refer to the absolute screen pixel location.

Parameters: `xx` location `yy` location

Returns:element located there

- List `twaver.network.TNetwork.getElementsPhysicalAt(Point p)`

Gets all elements located at given physical point. Physical point is refer to the absolute screen pixel location.

Parameters:pgiven physical point

Returns:all elements there

- List `twaver.network.TNetwork.getElementsPhysicalAt(int x, int y)`

Gets all elements located at given physical location. Physical point is refer to the absolute screen pixel location.

Parameters:xx locationyy location

Returns:all elements there

Network Viewport & Other Methods

Setting Viewport Location for Network

Get or set the logical coordinate location for network scroll viewport:

- `public Point getLogicalCenterPoint()`
- `public void setLogicalCenterPoint(Point centerPoint)`

Get or set the physical coordinate location for network scroll viewport:

- `public Point getPhysicalCenterPoint()`
- `public void setPhysicalCenterPoint(Point centerPoint)`


Network Size Functions

- `public Dimension getLogicalSize()`

Get network canvas logical size

- `public boolean adjustCanvasSize()`

Pack to adjust network canvas by all element locations. If network canvas size changed, return true. Call this method you can adjust network canvas size to its preferred size. This is the minimum rectangle to cover all elements on canvas. For example, call this to pack network canvas when remove some elements.

 Note that if `isEnabledAutoAdjustCanvasSize` is false, then this method will do nothing.

See Also:


`forceAdjustCanvasSize()`

Returns:

If the size changed, return true. Otherwise return false.

- `public boolean forceAdjustCanvasSize()`

Force to resize network canvas. Calls this method to adjust network canvas size to its preferred size. For example, call this to pack network canvas when remove some elements.

 Note that this method will not consider `isEnabledAutoAdjustCanvasSize` value.

See Also:

`forceAdjustCanvasSize()`

Returns:

If the size changed, return true. Otherwise return false.

- `enableAutoAdjustCanvasSize`

Whether resize network canvas automatically according to current elements locations. Default value is true.

See Also:

`isEnabledAutoAdjustCanvasSize()`

`setEnabledAutoAdjustCanvasSize(boolean enableAutoAdjustCanvasSize)`

- `enableIllegalLinkVisible`

Whether the illegal links are visible on network canvas. Illegal links are the links do not have start node or end node. Default value is false.

See Also:

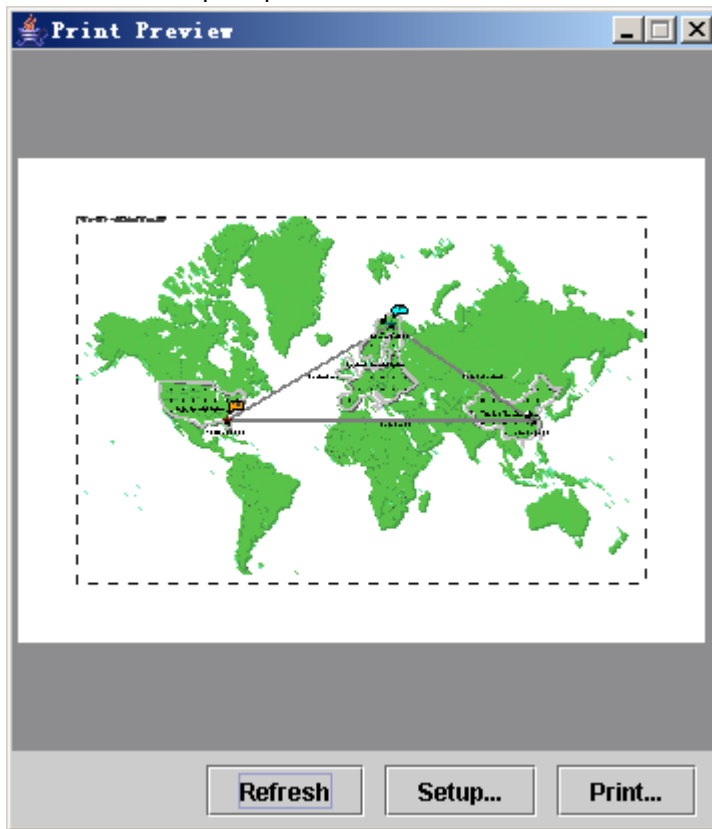
```
isEnabledIllegalLinkVisible()  
setEnabledIllegalLinkVisible(boolean enableIllegalLinkVisible)
```


Network Printing

The easy way to print a network view is clicking the print or print preview button on network toolbar.



Print button and print preview button



Print preview dialog

Network Representation

- [Attaching Swing Components in Network](#)
- [Customizing Network Background](#)
- [Network Canvas Interaction](#)
- [Using Alpha Transparent.](#)
- [Using Attachment](#)
- [Using ElementUI](#)
- [Using Filters](#)
- [Using Generator](#)
- [Using Link Layouter](#)
- [Using Network Layout](#)

Attaching Swing Components in Network

You can use `twaver.network.ui.ComponentAttachment` to attach any Swing component as a Attachment for any Element.

Define a customized ComponentAttachment:

```
public class MyAttachment extends ComponentAttachment {
    public MyAttachment(String name, ElementUI ui) {
        super(name, ui);

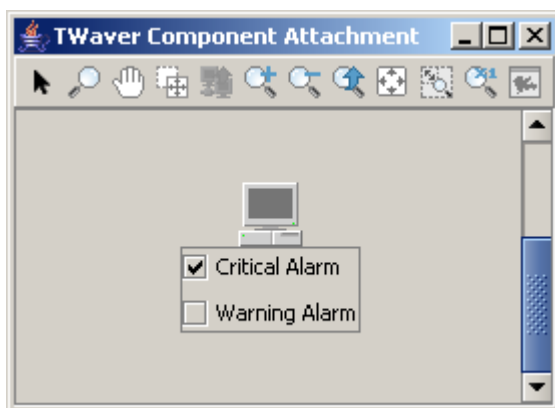
        JPanel panel =
new JPanel(new SingleFileLayout(SingleFileLayout.COLUMNN));
        panel.setOpaque(false);

        JCheckBox cb1 = new JCheckBox("Critical Alarm");
        cb1.setOpaque(false);
        cb1.setSelected(true);
        panel.add(cb1);
        JCheckBox cb2 = new JCheckBox("Warning Alarm");
        cb2.setOpaque(false);
        cb2.setSelected(false);
        panel.add(cb2);
        panel.setBorder(BorderFactory.createLineBorder(Color.gray));
        this.setComponent(panel);
        this.setSize(panel.getPreferredSize());
    }
}
```

Then we put this customized attachment to a Node:

```
Node node = new Node();
node.setLocation(200, 200);
box.addElement(node);

TUIManager.registerAttachment("att", MyAttachment.class);
node.addAttachment("att");
```

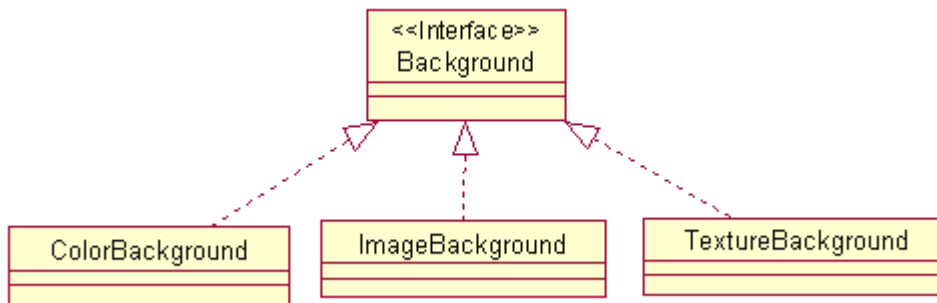


Note: You can use '`ComponentAttachment.setMinimizedIcon()`' method to specify a icon, when you double click the attachment, attachment will be collapsed into the icon.

Customizing Network Background

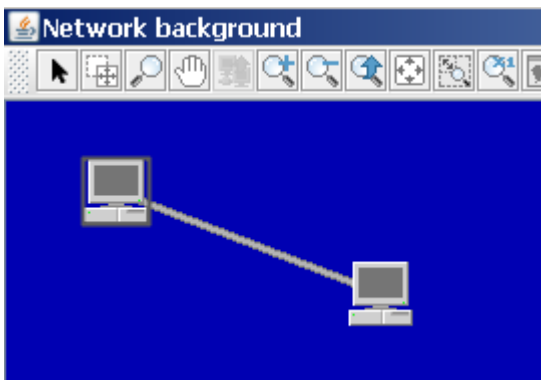
Network can display a background image behide all elements. Background can be an image, texture image, one color or vector graph (SHP/MapInfo). You can customize you own Background. TWaver defines Background interface, all kinds' background classes extends from this interface.

- ColorBackground: Single color background.
- ImageBackground: Single image background.
- TextureBackground: Texture image background.



Following codes show you how to use these backgrounds.
Single color background:

```
network.setBackground(new ColorBackground(Color.blue.darker()));
```



Single image as background:

```
network.setBackground(new ImageBackground("usa.gif"));
```

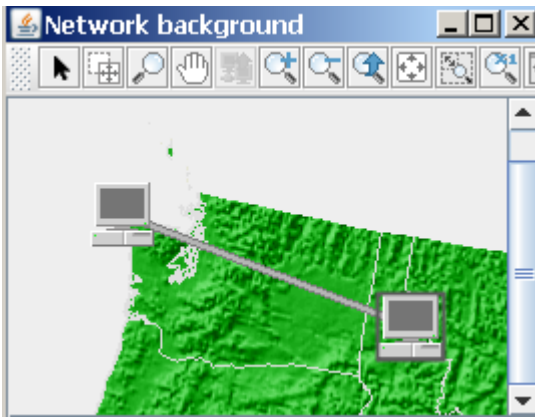
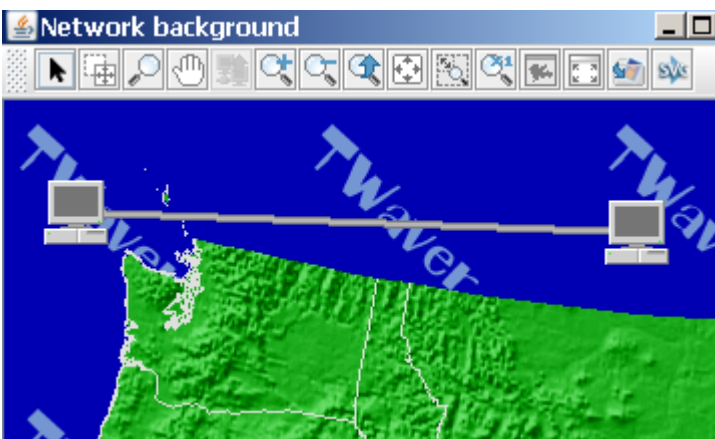


Image background with color and texture:

```
ImageBackground background=new ImageBackground("/demo/resource/images/usa.gif");
background.setColor(Color.blue.darker());
background.setTextureURL("/demo/resource/images/twaver_back.png");
network.setBackground(background);
```



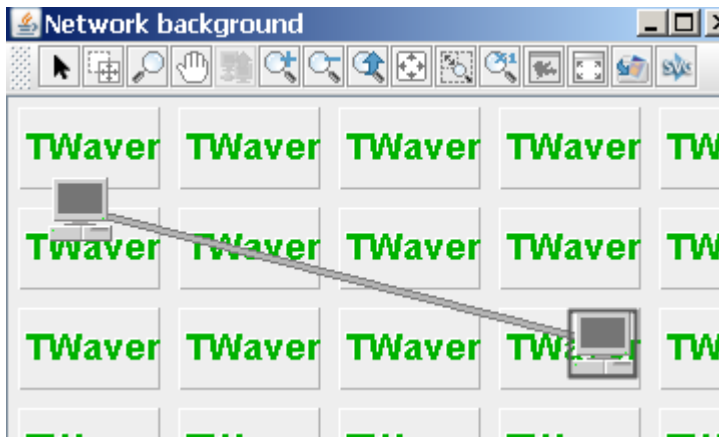
Texture image background:

```
network.setBackground(new TextureBackground("twaver_back.png"));
```



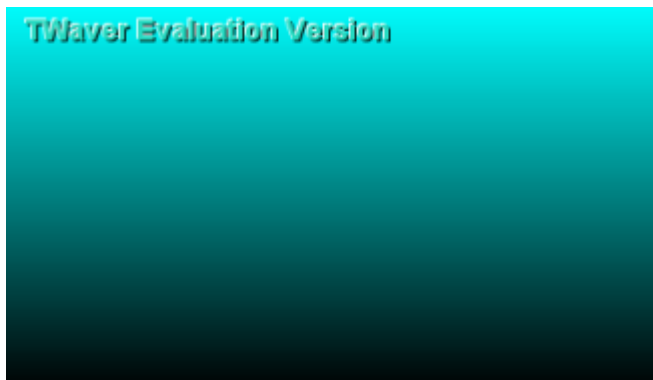
Customized texture background:

```
BufferedImage bufferedImage = new BufferedImage(80,50,BufferedImage.TYPE_INT_ARGB);
Graphics2D g2d = bufferedImage.createGraphics();
try {
    g2d.draw3DRect(5, 5, 70, 40, true);
    Font font = new Font("Arial", Font.BOLD, 18);
    g2d.setFont(font);
    g2d.setColor(Color.white);
    g2d.drawString("TWaver", 8, 30);
    g2d.setColor(Color.green.darker());
    g2d.drawString("TWaver", 9, 31);
}
finally {
    g2d.dispose();
}
//Create a texture paint from the buffered image
Rectangle rect = new Rectangle(0,0,80,50);
TexturePaint texture = new TexturePaint(bufferedImage, rect);
network.setNetworkBackground(new TextureBackground(texture));
```



Gradient color background:

```
ColorBackground background = new ColorBackground(Color.cyan);
background.setGradientColor(Color.black);
background.setGradientFactory(TWaverConst.GRAIENT_LINE_S);
background.setGradient(true);
network.setBackground(background);
```



Gradient image background:

```
ImageBackground background = new ImageBackground("/demo/resource/images/usa.gif");
background.setColor(Color.black);
background.setGradientColor(Color.cyan);
background.setGradientFactory(TWaverConst.GRADIENT_EXTEND_S);
background.setGradient(true);
network.setBackground(background);
```



Network Canvas Interaction

- [Customizing Default Mouse Action](#)
- [Customizing Popup Menu](#)
- [Interaction Mode](#)
- [Listening Keyboard & Mouse Event](#)
- [Using Copy & Paste Shortcut Key](#)

Customizing Default Mouse Action

All elements displayed in network component can specify a default action. Default actions are composed of action definition and action trigger. Action definition defines the action contents. Action trigger defines the action performed trigger.

TWaver elements are painted by ElementUI classes. All UI classes can receive all kinds user input events. TWaver defines several event types. You can also put more complex event types in MouseEvent and pass it in. The predefined event types are listed here:

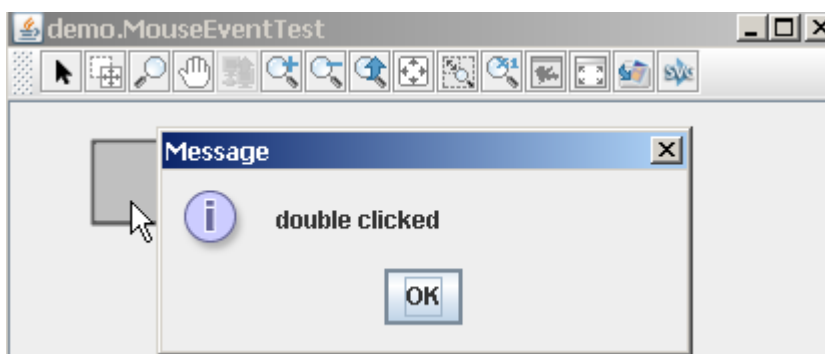
Name	Gesture
TWaverConst.MOUSE_LEFT_CLICKED	Click mouse left button
TWaverConst.MOUSE_LEFT_DOUBLE_CLICKED	Double click mouse left button
TWaverConst.MOUSE_RIGHT_CLICKED	Click mouse right button
TWaverConst.MOUSE_RIGHT_DOUBLE_CLICKED	Double click mouse right button

Now we create an element, when we double click it will display a message dialog. We define a new ElementUI extends from BaseElementUI, and rewrite the method "performAction":

```
public static class MyElementUI extends BaseElementUI {
    public MyElementUI(TNetwork network, BaseElement element) {
        super(network, element);
    }
    public void performAction(int gesture, MouseEvent e) {
        if (gesture == TWaverConst.MOUSE_LEFT_DOUBLE_CLICKED) {
            JOptionPane.showMessageDialog(network, "double clicked");
        }
    }
}
```

Then we define a new element which using this new ElementUI as the renderer:

```
BaseElement node = new BaseElement() {
    public String getUIClassID() {
        return MyElementUI.class.getName();
    }
};
node.setLocation(50, 50);
box.addElement(node);
```



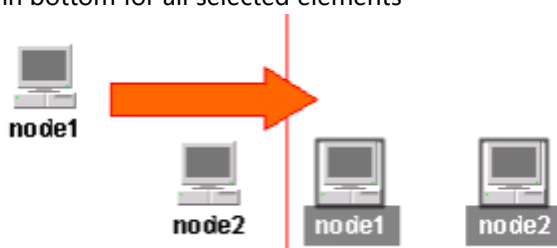
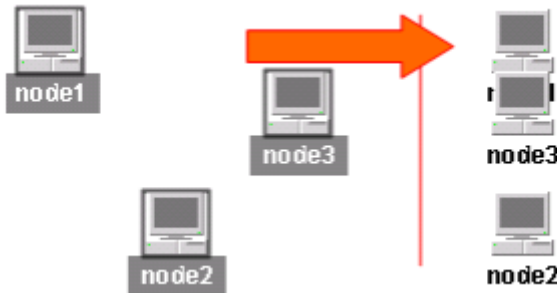

Double click element


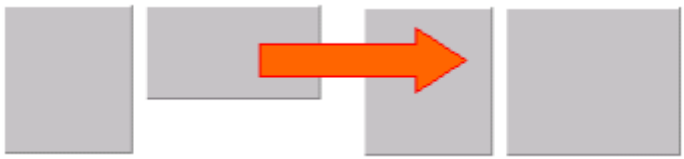

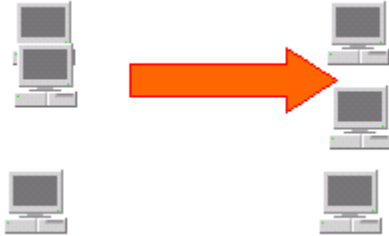
Several TWaver elements define the default action. For example, double click SubNetwork will drill down this subnetwork. Developers can set or override element default action.

Network Actions

TWaver inside defines a set of operations for Elements, include alignment, copy, paste, vertical pile up, horizontal pile up, resize, etc. You can call these actions directly or add these actions on your JButton to use them.

All defined Actions are listed here:

Action Name	Description
AlignBottomAction	Align in bottom for all selected elements 
AlignCenterAction	Align in horizontal for all selected elements 
AlignLeftAction	Align in left for all selected elements
AlignRightAction	Align in right for all selected elements
AlignTopAction	Align in top for all selected elements
LeftPileAction	Pile all selected elements up from left to right (gap=0) 
RightPileAction	Pile all selected elements up from right to left (gap=0)
TopPileAction	Pile all selected elements up from top to bottom (gap=0)

	
BottomPileAction	Pile all selected elements up from bottom to top (gap=0)
SameHeightAction	Same height for all selected elements 
SameWidthAction	Same width for all selected elements
EvenHSpaceAction	Same horizontal gap for all selected elements 
EvenVSpaceAction	Same vertical gap for all selected elements 
MoveFirstAction	Move to top
MoveLastAction	Move to bottom
ClearDataBoxAction	Clear all elements
CopyAction	Copy selected elements
PasteAction	Paste copied elements

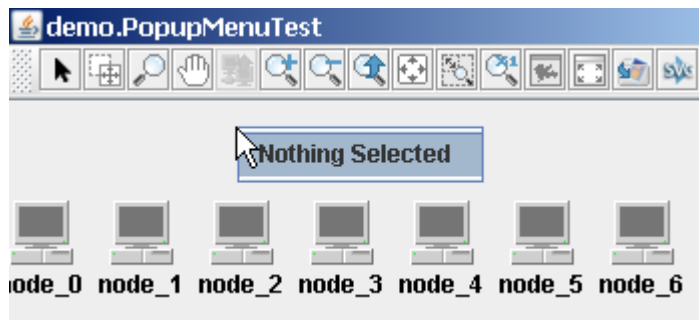
Customizing Popup Menu

Network provides the customizable popup menu. Network component itself doesn't create any popup menu. It delegates a menu generator (PopupMenuGenerator) to create popup menu. Once network components receive any popup menu trigger event like mouse right click, it will call PopupMenuGenerator to create menu and display it. The menu was totally created by PopupMenuGenerator.

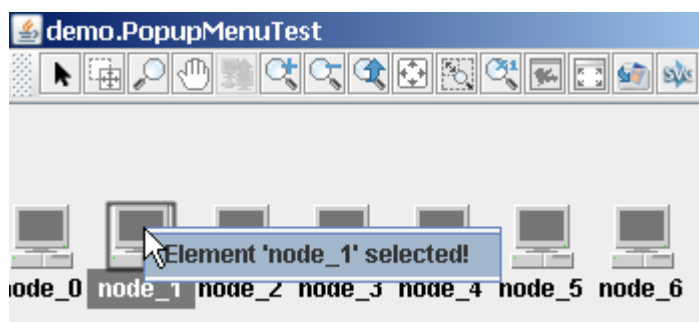
When network creates popup menus, The TView interface and mouse event object will be passed into the generator. Developers can use it to access the component and user mouse event to generate menus.

Now we give a sample to show you how to use popup menu generator. We assume that when popup menu creating, it display all selected elements with menu items. The code looks like this:

```
network.setPopupMenuGenerator(new PopupMenuGenerator(){
    public JPopupMenu generate(TView tview, MouseEvent mouseEvent) {
        JPopupMenu menu=new JPopupMenu();
        if(tview.getDataBox().getSelectionModel().isEmpty()){
            menu.add(new JMenuItem("Nothing Selected"));
        }else{
            Iterator it = box.getSelectionModel().selection();
            while(it.hasNext()){
                Element element=(Element)it.next();
                String menuText;
                if(element.getName()!=null){
                    menuText="Element '"+element.getName()+"' selected!";
                }else{
                    menuText=element.getID().toString();
                }
                menu.add(new JMenuItem(menuText));
            }
        }
        return menu;
    }
});
```



No element selected



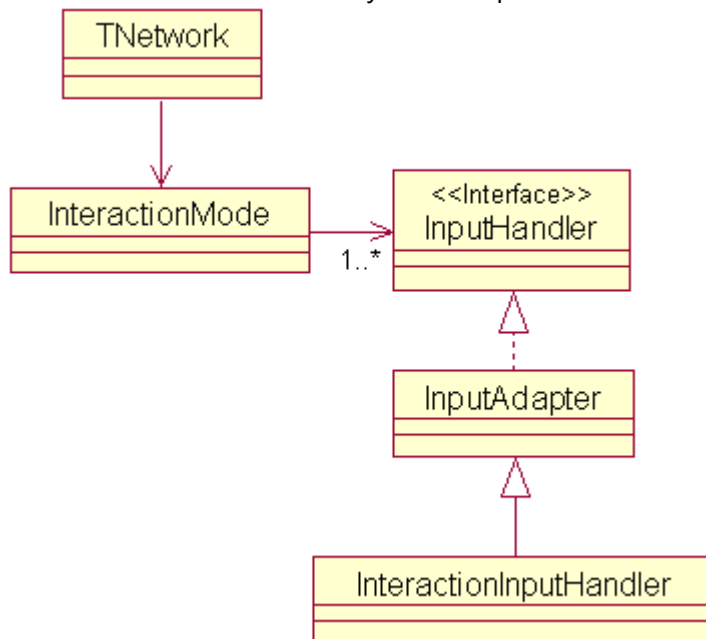
Node_1 selected

Interaction Mode

Network component needs to respond to user input events when user interacts with network component like selecting, pan, move, zoom etc. In different interactive modes, network component needs to respond to mouse or keyboard events differently. For example, when you drag to select a rectangle area on network by mouse, if currently is selecting mode, network should select all elements contains in this area; if currently is zoom to rectangle mode, then network component will zoom selected area to whole network view. In order to deal with all kinds' interaction, network component will switch to different interaction mode to respond to user with different action.

- Inside InteractionMode

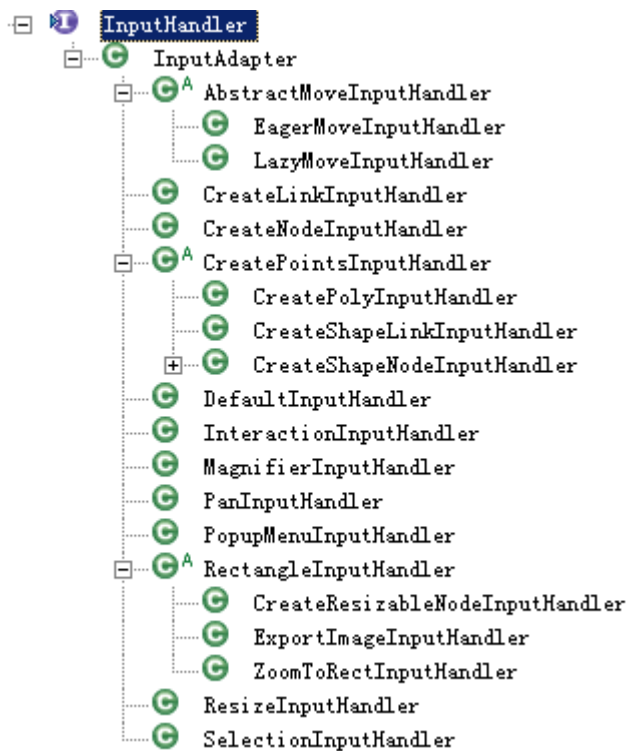
InteractionMode is combined by a lot of InputHandler instance:



- Inputhandler

Inputhandler defines an input event listener include both mouse and keyboard input event. No any new method defined.

All these InputHandler instances are responsible for different input event. InputHandler is a responder of one specified input event which is used to response one input event like mouse selection or view pan etc. TWaver provides following predefined InputHandlers:



Name	Usage
MoveInputHandler EagerMoveInputHandler LazyMoveInputHandler	Move all selected elements TWaver default mode Drag/drop elements lazily
CreateLinkInputHandler	Create link
CreateNodeInputHandler	Create node
CreatePolyInputHandler	Create Polygon
CreateShapeLinkInputHandler	Create ShapeLink
CreateShapeNodeInputHandler	Create ShapeNode
DefaultInputHandler	Default mode
InteractionInputHandler	Trigger default action
MagnifierInputHandler	Magnifier mode
PanInputHandler	Pan whole network view
PopupMenuInputHandler	Show popup menu
CreateResizableNodeInputHandler	Create a new resizable element
ExportImageInputHandler	Select the bound to export
ZoomToRectInputHandler	Zoom to rectangle area
ResizeInputHandler	Resize selected element

SelectionInputHandler

Elements selecting

You can combine several InputHandler instance to form a new interaction mode. Consider default interaction mode: by default, network component need to response user input event, it should be able to do:

- Select elements freely (SelectionInputHandler)
- Move elements freely (MoveSelectionInputHandler)
- Provide default action (InteractionInputHandler)
- Provide popup menu (PopupMenuInputHandler)

So we can create a new interaction mode like this:

```
InputHandler[] listeners = new InputHandler[] {
    //install a selection handler, make this network selectable.
    new SelectionInputHandler(network),
    //install a selection move handler, make this network movabl.
    new EagerMoveInputHandler (network),
    //install an action input handler, make this network active.
    new InteractionInputHandler(network),
    //install a popup menu input handler, make this network can
    //show a popup menu when user right clicks the mouse button.
    new PopupMenuInputHandler(network),
};
InteractionMode defaultMode = new InteractionMode(listeners);
```

You can create a new IntereActionMode instance by an array of InputHandler.

- Using InteractionModeFactory

TWaver provides many predefined InteractionMode, developers can use it directly. A factory class InteractionModeFactory can be used to create these interaction mode instances easily. All available interaction modes are listed in this table. Please note every interaction mode can include one or more InputHandler.

InteractionModeFactory	
getCreateLinkMode(in linkclass : Class)	
getCreatePolyMode()	
getCreateShapeLinkMode()	
getCreateNodeMode()	
getCreateNodeMode(in elementType : Class)	
getPanMode()	
getMagnifierMode()	
getZoomToRectMode()	
getDefaultMode()	
getDefaultLazySelectionMode()	

Name	Method	Description
CreateLinkMode	getCreateLinkMode (TNetwork network,	Create new links on canvas

	Class linkclass)	
CreateNodeMode	getCreateNodeMode (TNetwork network)	Create new nodes on canvas
	getCreateNodeMode (TNetwork network, Class elementType)	
DefaultMode	getDefaultMode (TNetwork network)	Default mode. Elements can selected or moved
PanMode	getPanMode (TNetwork network)	Pan canvas
ZoomToRectMode	getZoomToRectMode (TNetwork network)	Enlarge selected rectangle area to whole network view
CreatePolyMode	getCreatePolyMode (TNetwork network)	This is used for creating polygon on network canvas
MagnifierMode	getMagnifierMode (TNetwork network)	Magnifier mode
DefaultLazySelectionMode	getDefaultLazySelectionMode (TNetwork network)	Lazily move selected elements when drag/drop
CreateShapeLinkMode	getCreateShapeLinkMode (TNetwork network)	Used when create link on network canvas

Name	Included InputHandlers
CreateLinkMode	new DefaultInputHandler(network), new CreateLinkInputHandler(network, linkclass),
CreateNodeMode	new DefaultInputHandler(network), new CreateNodeInputHandler(network) OR new DefaultInputHandler(network), new CreateNodeInputHandler(network, elementType)
DefaultMode	new DefaultInputHandler(network), //install resize input handler. new ResizeInputHandler(network), //install a selection handler, make this network selectable. new SelectionInputHandler(network), //install a selection move handler, make this network movabl. new EagerMoveInputHandler(network), //install an action input handler, make this network active. new InteractionInputHandler(network), //install a popup menu input handler, make this network can //show a popup menu when user right click the mouse button. new PopupMenuInputHandler(network)
PanMode	new DefaultInputHandler(network),

	new PanInputHandler(network)
ZoomToRectMode	new DefaultInputHandler(network), new ZoomToRectInputHandler(network)
CreatePolyMode	new DefaultInputHandler(network), new CreatePolyInputHandler(network)
MagnifierMode	new DefaultInputHandler(network), new ResizeInputHandler(network), new SelectionInputHandler(network), new EagerMoveInputHandler(network), new InteractionInputHandler(network), new MagnifierInputHandler(network)
DefaultLazySelectionMode	new DefaultInputHandler(network), new ResizeInputHandler(network), new SelectionInputHandler(network), new LazyMoveInputHandler(network), new InteractionInputHandler(network), new PopupMenuInputHandler(network),
CreateShapeLinkMode	new DefaultInputHandler(network), new CreateShapeLinkInputHandler(network)

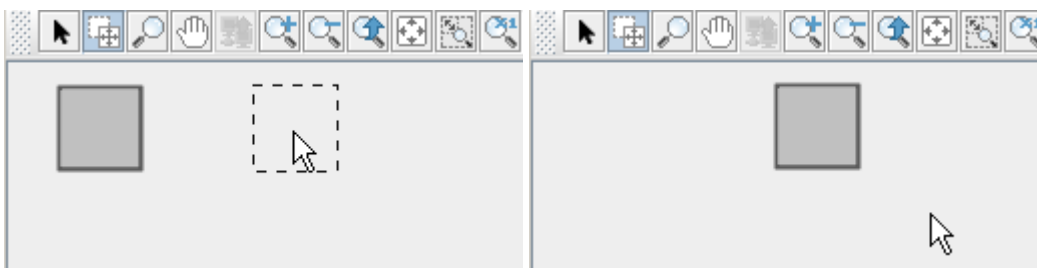
- Use InteractionMode

Use setInteractionMode method of TNetwork class to let new interaction mode take effect:


```
network.setInteractionMode(defaultMode);
```

Use lazyMoveInoutHandler change the default drag/drop behavior.

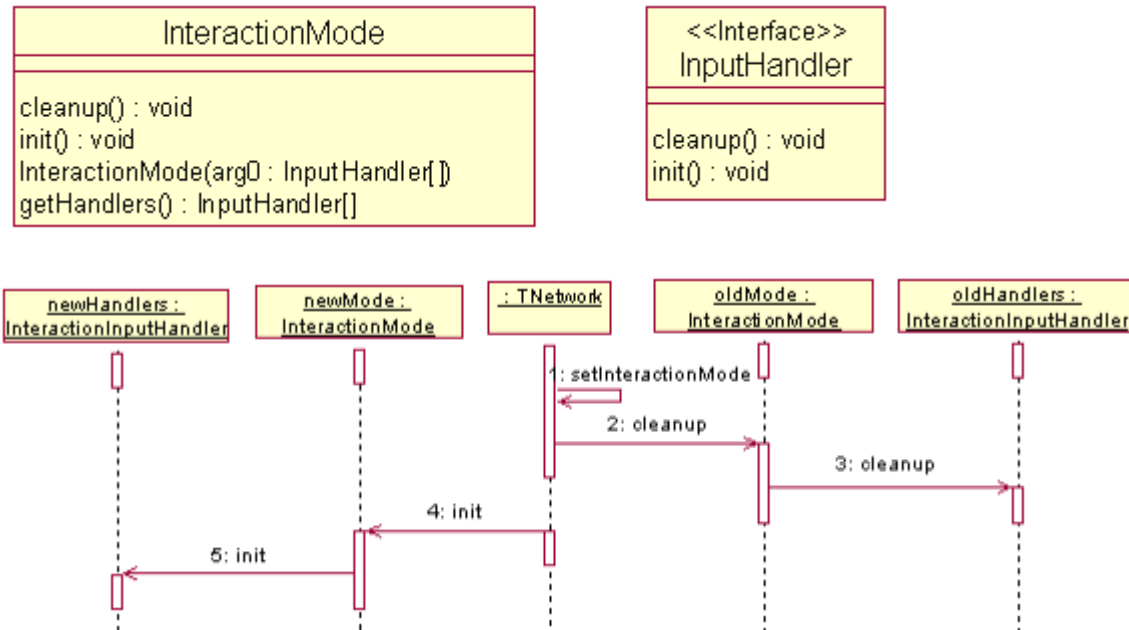
```
InteractionMode m
=InteractionModeFactory.getDefaultLazySelectionMode(network);
network.setInteractionMode(m);
```



- InteractionMode initialize/cleanup

InteractionMode can initialize and cleanup each InputHandler instance when interaction mode was setted or removed from network component. Developers can provide extra actions in method "init" and "cleanup". For example, when PanInteractionMode instance was setted to network component, we need change the mouse cursor to the pan icon  ; when it was removed from network component, we need to reset the default cursor





Because InteractionMode will call "init" method when it loaded, we can write following code:

```

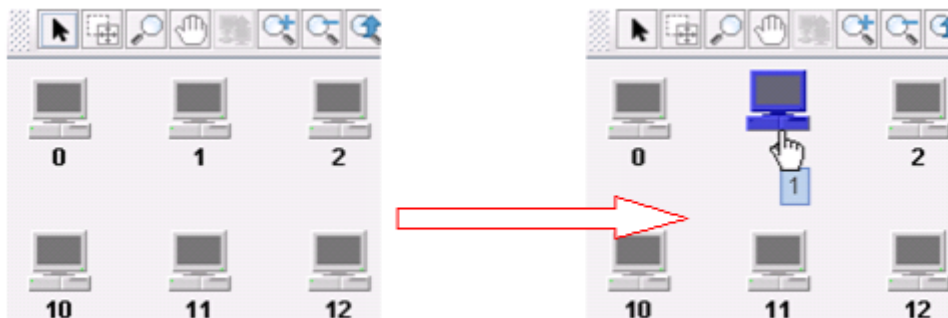
public void init() {
    network.getCanvas().setCursor(
        Cursor.getPredefinedCursor(Cursor.MOVE_CURSOR));
}
public void cleanup() {
    Cursor c=Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR);
    network.getCanvas().setCursor(c);
}

```

Developers can provide more actions when the interaction mode was loaded or unloaded.

- Customize InputHandler

Here we show you how to customize InputHandler to highlight Element when move mouse cursor over it.



Highlighted on Move over

Highlighted on Move

over

Code:

```

InputHandler[] handlers = new InputHandler[] {
    new DefaultInputHandler(network),
    new ResizeInputHandler(network),
}

```

```


new SelectionInputHandler(network),
new EagerMoveInputHandler(network),
new InteractionInputHandler(network),
new PopupMenuInputHandler(network) ,
//Add the customized InputHandler.
//need put in the last position in this array.
new MyInputHandler(network)
};
\\
//set interaction mode
network.setInteractionMode(new InteractionMode(handlers));
//Define new InputHandler
public class MyInputHandler extends InputAdapter {
private TNetwork network;
private AbstractElement currentElement;
private Color highLightColor = new Color(100, 100, 255);
private int dy = 4;
private Color oldColor;
private Point oldLocation;
private boolean dragged = false;
public MyInputHandler(TNetwork network) {
    this.network = network;
}
//listen mouse click event
public void mouseClicked(MouseEvent e) {
    if (currentElement != null) {
        JOptionPane.showMessageDialog(network, "clicked at node '" +
currentElement.getID() + "'");
    }
}
//listen mouse drag event
public void mouseDragged(MouseEvent e) {
    dragged = true;
}
//listen mouse move event so when
//mouse move over an Element, highlight it.
public void mouseMoved(MouseEvent e) {
    Element element = network.getElementLogicalAt(e.getPoint());
    if (element != null && element instanceof AbstractElement) {
//set mouse cursor
Cursor c=Cursor.getPredefinedCursor(Cursor.HAND_CURSOR);
network.getCanvas().setCursor(c);
Color renderColor=((AbstractElement) element).getRenderColor();
if (element.equals(currentElement) && highLightColor.equals(renderColor)) {
    return;
}
//highlight element
set((AbstractElement) element);
return;
}
//reset element
reset();
    Cursor c=Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR);
network.getCanvas().setCursor(c);
}
//set highlighted display
private void set(AbstractElement element) {
    reset();
    currentElement = element;
    oldColor = element.getRenderColor();
    oldLocation = element.getLocation();
    element.putRenderColor(highLightColor);
    element.setLocation(oldLocation.x, oldLocation.y - dy);
    dragged = false;
}
//reset default display

```

```
private void reset() {  
    if (currentElement != null) {  
        currentElement.putRenderColor(oldColor);  
        if (!draged) {  
            currentElement.setLocation(oldLocation);  
        }  
    }  
    currentElement=null;  
}
```

Listening Keyboard & Mouse Event

As an extension of Swing JPanel, you can catch keyboard or mouse events by installing your listeners on the network canvas.


 network canvas is an internal Swing component contained in Network. All listeners should add on network canvas but not Network itself.

The following event listener can be added on network canvas:

- java.awt.event.KeyListener
- java.awt.event.MouseListener
- java.awt.event.MouseMotionLisener

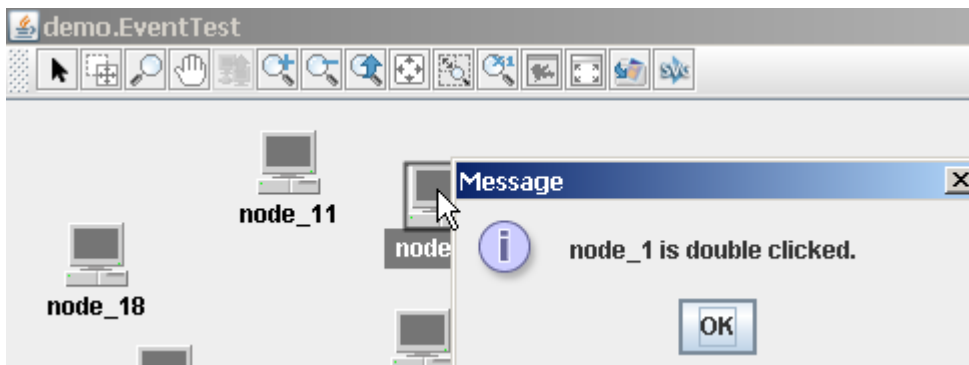
To determine the clicked element on the network canvas, you can use several methods like TNetwork.getElement*At.

- TNetwork.getElementLogicalAt(int,int/Point2D):get the element at specified location. "Logical" means the given location is a logical point, no matter the zoomer is changed.
- TNetwork.getElementPhysicalAt(int,int/Point2D):get the element at specified location. "Physical" means the given location is a physical point, that is, the given point is the absolute offset of origin of coordinates. In this case you have to consider the current zoomer of the network.

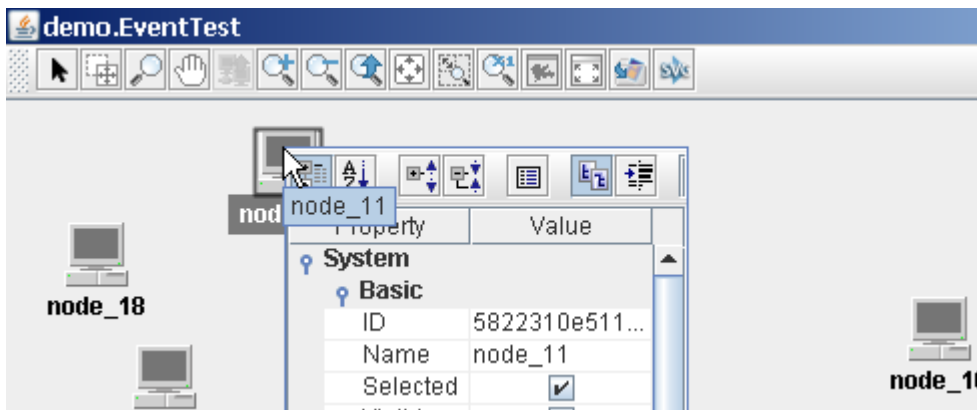
 if have more elements at the same location, then the top one will be returned.

The following procedure shows you how to determine the double clicked element of a network component.


```
network.getCanvas().addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        if (e.getClickCount() == 2) {
            Element element = network.getElementPhysicalAt(e.getPoint());
            if (element == null) {
                JOptionPane.showMessageDialog(network, "double click on nothing.");
            } else {
                JOptionPane.showMessageDialog(network,
                    element.getName() + " is double clicked.");
            }
        }
        //If mouse right click, popup Element property sheet
        else if (e.getClickCount() == 1 && e.getButton() == MouseEvent.BUTTON3) {
            Element element = network.getElementPhysicalAt(e.getPoint());
            if (element != null) {
                TPropertySheet sheet = new TPropertySheet(box);
                sheet.setEditable(true);
                JPanel pane = new TPropertySheetPane(sheet);
                pane.setPreferredSize(new Dimension(200, 300));
                box.getSelectionModel().setSelection(element);
                TWaverUtil.showPopupComponet(pane,
                    network.getCanvas(),
                    e.getPoint());
            }
        }
    }
});
```



Double click event



Right mouse click event

 You also can use following methods of network component to register a double click action listener. They are encapsulated methods base on the Swing event model, just more easy to use.

Method	Description
addElementdoubleClickedActionListener	Register an action listener when the element is double clicked.
addBackgroundDoubleClickedActionListener	Register an action listener when the network background is double clicked.

Using Copy & Paste Shortcut Key

Using default copy/paste shortcut key

By default, TNetwork does not enable the shortcut key for copy/paste actions. If you want enable it, use this method:

```
network.setEnableCopyPasteWithKeyboard(true);
```

The default shortcut key for copy is "Ctrl+C" and "Ctrl+V" for paste. The default action for copy is only copy the selected Elements. Please note that the default copy action does not copy any children Elements. If you want to do that, you need to customize a new action.

Customize Copy/Paste Action

First, we need to add a KeyListener, so when press "ctrl+c" we can copy the selected Elements (box.copySelection()). Then in the paste action, we can create new Elements according to the copied Elements.

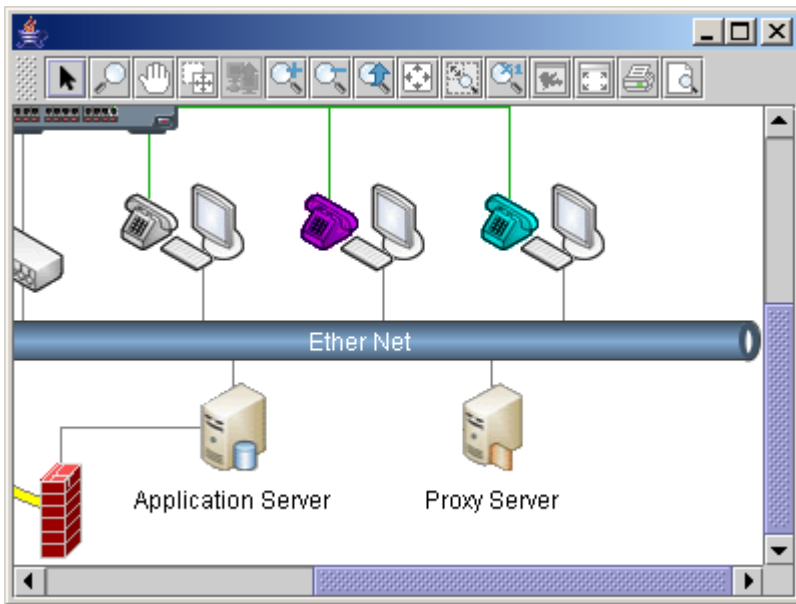
```
network.getCanvas().addKeyListener(new KeyAdapter() {
    public void keyPressed(KeyEvent e) {
        if (e.isControlDown() && e.getKeyCode() == KeyEvent.VK_C) {
            //copy elements.
            new CopyAction(network).actionPerformed(null);
            //or use "box.copySelection();"
        } else if (e.isControlDown() && e.getKeyCode() == KeyEvent.VK_V) {
            //paste action
            pasteElements(null, box);
        }
    }
});

public static List pasteElements(Element parentElement, TDataBox box) {
    List pastes = new ArrayList();
    try {
        List list = TWaverUtil.getCopyElements();
        if (list != null && list.size() > 0) {
            int offset = TWaverUtil.getPasteOffset() + 5;
            TWaverUtil.setPasteOffset(offset);
            for (int i = 0; i < list.size(); i++) {
                Element element = (Element) list.get(i);
                Element newElement = (Element) element.copy(box);
                //set my property
                //set display name
                newElement.setDisplayName("copy_" + i);
                //set element location
                if (element.getLocation() != null) {
                    int x = element.getLocation().x + offset;
                    int y = element.getLocation().y + offset;
                    newElement.setLocation(x, y);
                }
                //set parent node
                if (parentElement != null && newElement.getParent() == null) {
                    newElement.setParent(parentElement);
                }
                //add them into the DataBox
                if (!box.contains(newElement)) {
                    box.addElement(newElement);
                }
                pastes.add(newElement);
            }
        }
        box.getSelectionModel().setSelection(pastes);
    }
}
```

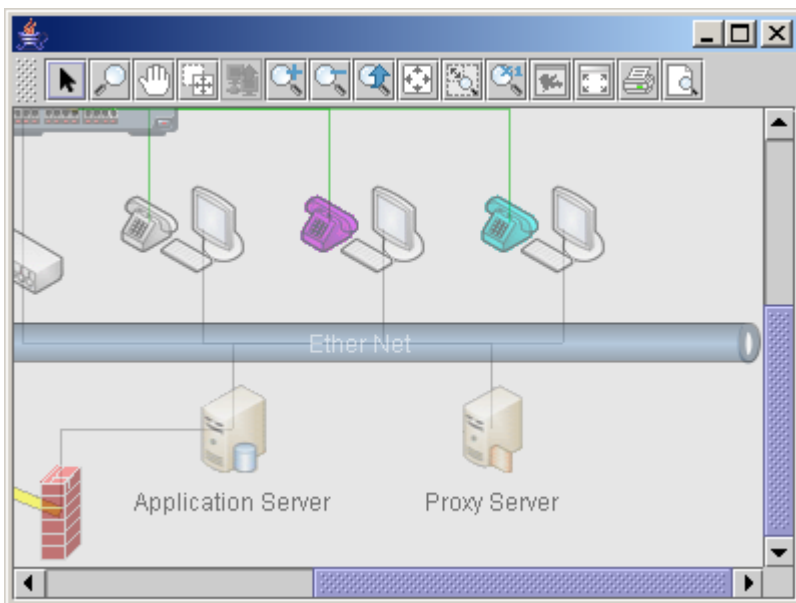
```
}  
}catch(Exception ex){  
    ex.printStackTrace();  
}  
return pastes;  
}
```


Using Alpha Transparent.

The network canvas can be alpha transparent. Use method 'setAlpha' of network to specify the alpha value.



No Alpha Transparency



Network with Alpha Transparent

Using Attachment

In section [Adding Decorated Icons](#) we know how to add attachment icons on an Element. In fact we can create more complex attachments to present network events. You can create attachment with bitmap icon, Java2D, or mixed.

Now we create an Attachment object with bitmap icon and Java2D drawing. We have a thermometer picture as below:



Then we extend more functions on this attachment:

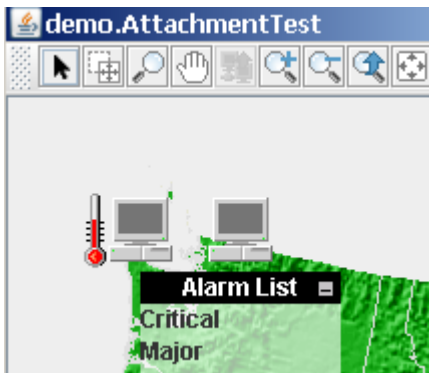
- Draw current temperature value on the picture.
- Double click the attachment show the value with a dialog.

```
public class ThermometerAttachment extends IconAttachment {
    public ThermometerAttachment(String name, ElementUI ui) {
        super(name, ui, "/demo/network/miscellaneous/instrument/thermometer.png");
    }
    public void paint(Graphics2D g2d) {
        //call super to paint attachment normally.
        super.paint(g2d);
        //then we draw the value line on the picture.
        g2d.setStroke(new BasicStroke(2));
        g2d.setColor(Color.red);
        int basePoint = 27;
        int topPoint = 5;
        //temperature value saved in 'temperature' client property.
        Object o = getElementUI().getElement().getClientProperty("temperature");
        if (o != null) {
            int value = ((Integer) o).intValue();
            g2d.drawLine(getLocation().x + 5, getLocation().y + basePoint, getLocation().x + 5, getLocation().y + 20 - value + topPoint);
        }
    }
    //add mouse left double click action.
    public void performAction(int gesture, MouseEvent e) {
        if(TWaverConst.MOUSE_LEFT_DOUBLE_CLICKED == gesture){
            Object value = getElementUI().getElement().getClientProperty("temperature");
            if (value != null) {
                //show the value message
                JOptionPane.showMessageDialog(this.getElementUI().getNetwork(), "temperature=" + ((Integer) value).intValue());
            }
        }
    }
}
```

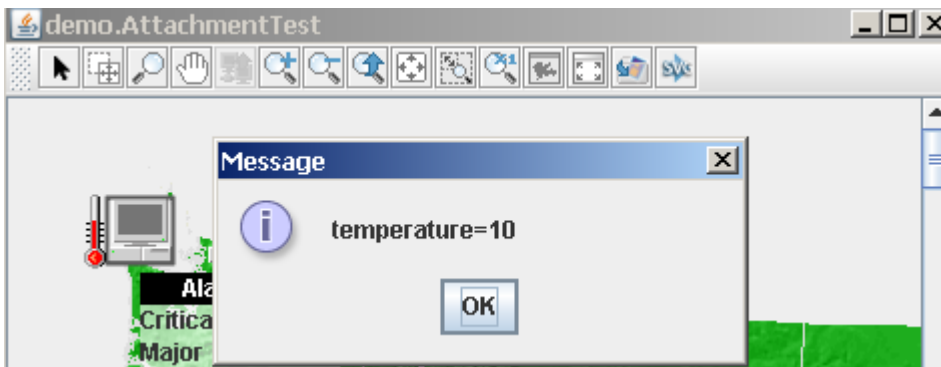
Now we can use this attachment:

```
Node node=new Node("one Node");
//register the attachment to system.
TUIManager.registerAttachment("Thermometer", ThermometerAttachment.class);
//Adds attachment with the specified attachment name
node.addAttachment("Thermometer");
Node node = new Node();
//display 'Thermometer' attachment
node.putClientProperty("StateIcon:Thermometer", Boolean.TRUE);
```

```
//set attachment value.
node.putClientProperty("temperature", new Integer(10));
//set attachment's position
node.putAttachmentPosition(TWaverConst.POSITION_LEFT);
node.setLocation(100,50100);
box.addElement(node);
```



Show temperature attachment

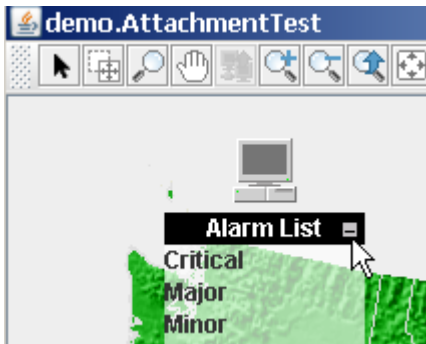


Double click attachment

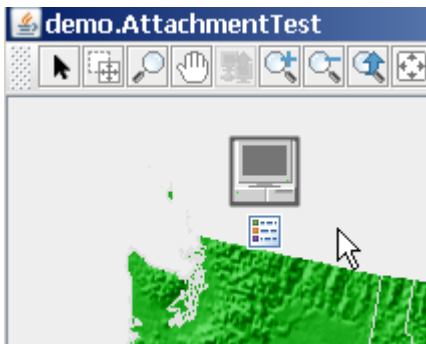
Use ComponentAttachment

```
public class ListAttachment extends ComponentAttachment{
    public ListAttachment(String name, ElementUI ui) {
        super(name, ui);
        JPanel panel = new JPanel();
        JLabel label = new JLabel("Alarm List", JLabel.CENTER);
        label.setBackground(Color.BLACK);
        label.setForeground(Color.WHITE);
        label.setOpaque(true);
        panel.setLayout(new GridLayout(6,1));
        panel.add(label);
        panel.add(new JLabel(AlarmSeverity.CRITICAL.getDisplayName()));
        panel.add(new JLabel(AlarmSeverity.MAJOR.getDisplayName()));
        panel.add(new JLabel(AlarmSeverity.MINOR.getDisplayName()));
        panel.add(new JLabel(AlarmSeverity.WARNING.getDisplayName()));
        panel.add(new JLabel(AlarmSeverity.INDETERMINATE.getDisplayName()));
        this.setMinimizable(true);
        this.setPosition(TWaverConst.POSITION_BOTTOM);
        this.setYOffset(5);
        panel.setOpaque(true);
        panel.setBackground(new Color(255, 255, 255, 160));
        this.setMinimizedIcon(TWaverUtil.getImageIcon("/demo/network/topo/attachment/alarm.png"));
        this.setComponent(panel);
    }
}
```

```
}
```

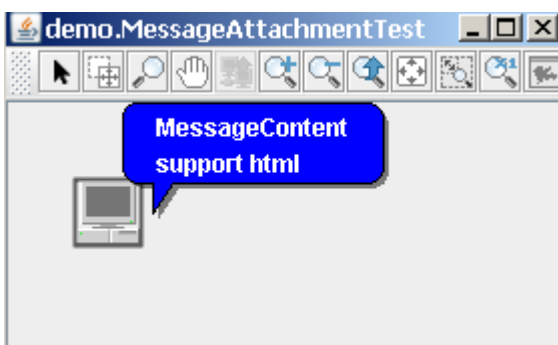


Click the minimize icon



Use MessageAttachment

```
Node node=new Node();
node.addAttachment(TWaverConst.ATTACHMENT_MESSAGE);
node.putMessageBackground(Color.blue);
node.putMessageForeground(Color.white);
node.putMessagePosition(TWaverConst.POSITION_RIGHT);
node.putMessageContent("<html>MessageContent<br>support html</html>");
box.addElement(node);
```



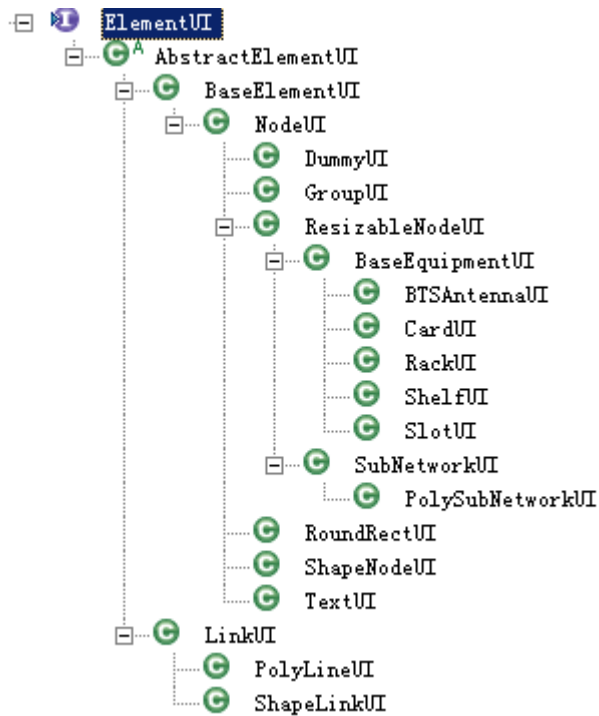
Using ElementUI

ElementUI defines an element UI delegate painter. By the UI class, the network element display style will be set. Class ElementUI



ElementUI is the dedicated renderer for the Elements which displayed on Network component. A type of Element needs a type of ElementUI to paint. Developers can create a customized ElementUI class to customize the appearance and interaction for an Element. Element, ElementUI and TNetwork all work as the Model-View-Controller model.

Class ElementUI



Customize ElementUI

Flowing example extends the ElementUI and paint a shadow for the Element. The example define a new Element and ElementUI (MyElement and MyElementUI). In class MyElementUI, we override methods paintBody/elementPropertyChange/getUIBounds/performAction and in MyElement override method getUIClassID return class name of MyElementUI:

```

public static class MyElementUI extends NodeUI {
    private Polygon shadow = null;

    public MyElementUI(TNetwork network, Node element) {
        super(network, element);
    }

    private Polygon createShadow(){
        Rectangle bounds = element.getBounds();
        int[] xs = new int[] {
            bounds.x + bounds.width / 4,
            bounds.x + bounds.width + bounds.width / 2, bounds.x + bounds.width,
            bounds.x
        };
        int[] ys = new int[] {
            bounds.y + bounds.height - bounds.height / 3,
            bounds.y + bounds.height - bounds.height / 3,
            bounds.y + bounds.height,
            bounds.y + bounds.height
        };
        int number = 4;
        shadow = new Polygon(xs, ys, number);
        return shadow;
    }

    public void paintBody(Graphics2D g2d) {
        if(shadow==null){
            createShadow();
        }
        Color shadowColor = new Color(128, 128, 128, 128);
        g2d.setColor(shadowColor);
    }
}

```

```

        g2d.fill(shadow);
        super.paintBody(g2d);
    }

    public void performAction(int gesture, MouseEvent e) {
        if (gesture == TWaverConst.MOUSE_LEFT_DOUBLE_CLICKED) {
            JOptionPane.showMessageDialog(getNetwork(), "double clicked");
        }
    }

    public void elementPropertyChange(PropertyChangeEvent evt) {
        super.elementPropertyChange(evt);
        createShadow();
    }

    //We add the shadow, so need to reconsider the element bounds.
    public Rectangle getUIBounds() {
        Rectangle result = super.getUIBounds();
        if (shadow != null) {
            result.add(shadow.getBounds());
        }
        return result;
    }
}

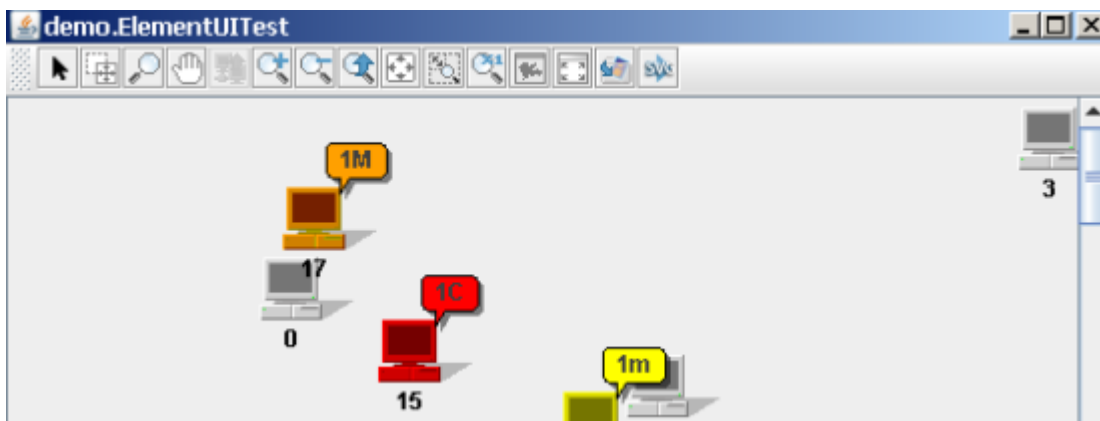
```

```

public static class MyElement extends Node {
    public MyElement() {
        super();
    }
    public String getUIClassID() {
        //return the class name of the related element ui
        return MyElementUI.class.getName();
    }
}

```

Run this example:



Using Filters

Filters used in TNetwork:


Name	Related Methods in TNetwork
PaintSelectionStateFilter Element selection state whether visible	PaintSelectionStateFilter getPaintSelectionStateFilter() setPaintSelectionStateFilter(PaintSelectionStateFilter paintSelectionStateFilter)
DoubleClickFilter Whether enable the default double-click action on Canvas (such as drill-down a SubNetwork or expand/close a Group)	DoubleClickFilter getDoubleClickFilter() setDoubleClickFilter(DoubleClickFilter doubleClickFilter)
VisibleFilter Element visible	addVisibleFilter(VisibleFilter filter) List getVisibleFilters() removeVisibleFilter(VisibleFilter filter)
MovableFilter Element movable	setMovableFilter(MovableFilter filter) addMovableFilter(MovableFilter filter) removeMovableFilter(MovableFilter filter) clearMovableFilters() List getMovableFilters()
ResizableFilter Element resizable	ResizableFilter getResizableFilter() setResizableFilter(ResizableFilter resizableFilter)
ElementBoundsInvalidatableFilter When Element property changed, whether invalidate Element bounds. If this property do not affect Element bounds or display, no need to invalidate bounds	ElementPropertyChangeRepaintFilter getElementPropertyChangeRepaintFilter() setElementBoundsInvalidatableFilter(ElementBoundsInvalidatableFilter filter)
ElementPropertyChangeRepaintFilter When Element property changed, whether need to repaint.	ElementPropertyChangeRepaintFilter getElementPropertyChangeRepaintFilter() setElementPropertyChangeRepaintFilter(ElementPropertyChangeRepaintFilter filter)
ElementLabelEditableFilter Whether Label is editable	EditableFilter getElementLabelEditableFilter() setElementLabelEditableFilter(EditableFilter elementLabelEditableFilter)
SendToTopFilter Whether need to send to top	SendToTopFilter getSendToTopFilter() setSendToTopFilter(SendToTopFilter sendToTopFilter)
SelectableFilter Whether Element is selectable	addSelectableFilter(SelectableFilter filter) removeSelectableFilter(SelectableFilter filter) List getSelectableFilters()

- [ElementBoundsInvalidatableFilter](#)
- [ElementLabelEditableFilter](#)
- [ElementPropertyChangeRepaintFilter](#)
- [PaintSelectionStateFilter](#)
- [SelectableFilter](#)
- [SendToTopFilter](#)
- [Using DoubleClickFilter](#)
- [Using MovableFilter](#)
- [Using ResizableFilter](#)

- [Using VisibleFilter](#)

ElementBoundsInvalidatableFilter

Function: whether invalidate the Element bound when Element property changed. Some properties of an Element may affect the Element bound and need to repaint immediately and some may not. If the property affect Element bound and need to recalculate, return true. If the property does not affect Element bound and just needs Element repaint, you need to use another filter [ElementPropertyChangeRepaintFilter](#). Please see following section for more information.

	<<Interface>>
	ElementBoundsInvalidatableFilter
	isElementBoundsInvalidatable(in element : Element,in event : PropertyChangeEvent) : boolean

Event: the event is Element property change event.



ElementLabelEditableFilter

Function: whether Element label text is editable. Return true means editable. If it is editable, you can double click Element label text to edit it. Default value is false.

	<<Interface>>
	Ed itable Filter
	isEditable(in element : Element) : boolean

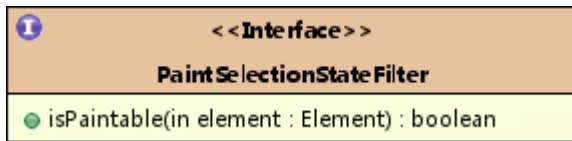
ElementPropertyChangeRepaintFilter

Function: when Element property value changed, whether need to repaint Element area.
 Like filter ElementBoundsInvalidatableFilter, they all repaint Element if return true. But filter
 ElementBoundsInvalidatableFilter will repaint Element and recalculate Element bound, but this filter just repaint
 Element.

	<<Interface>>
	ElementPropertyChangeRepaintFilter
	isInterested(in element : Element,in event : PropertyChangeEvent) : boolean

PaintSelectionStateFilter

Function: whether display selection border when Element is selected.



Please note the difference with [SelectableFilter](#) used to determine whether a Element is selectable.


SelectableFilter

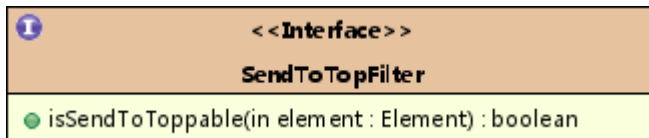
Function: control an Element whether is selectable.

I	<<Interface>>
	SelectableFilter
●	isSelected(in element : Element) : boolean

SendToTopFilter

Function: when an Element is selected, whether send it to top, by default is true. By default, an Element added into a DataBox or selected, it will automatically send to to level. You can control this default behaviour by this filter.

 when we see send an Element to top, it is only internal moved in its located layer. It will not move out of its layer. See more information here: [Using DataBox LayerModel](#)



Using DoubleClickableFilter

Some Elements in TWaver have default double click action. For example, double click a Group will reverse its expanded status; double click a Link will make it bundled; double click a SubNetwork will drill it down. But, if you dislike the default double click action, you can change it. You can make a double click filter to tell TWaver whether perform the default action for the double clicked Element. Let's make an example to close the default double click action for Link bundle, Group and SubNetwork.

```
//add SubNetwork, Group, Link to network.
Node from = new SubNetwork();
from.setImage("/demo/node.png");
from.setLocation(100, 100);
box.addElement(from);

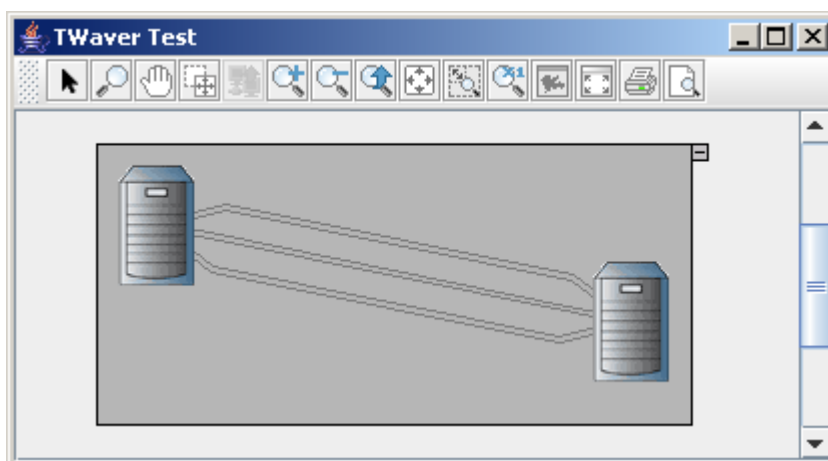
Node to = new SubNetwork();
to.setImage("/demo/node.png");
to.setLocation(300, 200);
box.addElement(to);


box.addElement(new Link(from, to));
box.addElement(new Link(from, to));
box.addElement(new Link(from, to));

Group group = new Group();
box.addElement(group);
from.setParent(group);
to.setParent(group);
group.setExpand(true);

//add a double click filter, ignore all default Elements action.
network.setDoubleClickableFilter(new DoubleClickableFilter() {
    public boolean isDoubleClickable(Element element) {
        return false;
    }
});
```

Run this code, you can see this network will not response any double click action.

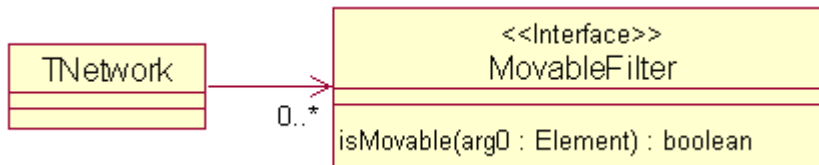


 network double click filter only used to disable Element default action. It will not affect any mouse listeners manually added on network canvas.


```
//This still work.  
network.getCanvas().addMouseListener(new MouseAdapter(){  
public void mouseClicked(MouseEvent e) {  
    System.out.println("This is still work");  
}  
});
```

Using MovableFilter

Network component control each element whether it can be moved by MovableFilter.



MovableFilter interface use method "isMovable" to determine whether given element is movable. More than one movable filters can added to one network. Element can be dragged only if all movable filters are allow it movable.

You can add and remove movable filters by methods "addMovableFilter" and "removeMovableFilter" of TNetwork.

In order to demonstrate this, we make a sample to show you how movable filter working:

First, we create lot of nods on network and give each node the order index as its name. Then we set the network popup menu generator to provide two menu items: one is used to fix all even index nodes, another used to reset all nodes can move freely.

Now define a new movable filter, which used to fix all even index nodes:

```

final MovableFilter filter = new MovableFilter() {
    public boolean isMovable(Element element) {
        if (element.getName() != null &&
            Integer.valueOf(element.getName()).intValue() % 2 == 0) {
            return false;
        }
        return true;
    }
};
  
```

Then we define a popup menu generator to create the two menu item:

```

network.setPopupMenuGenerator(new PopupMenuGenerator() {
    public JPopupMenu getPopupMenu(TView tview, MouseEvent mouseEvent) {
        TNetwork network=(TNetwork)tview;
        JPopupMenu menu = new JPopupMenu();

        Action action = new AbstractAction() {
            public void actionPerformed(ActionEvent e) {
                network.removeMovableFilter(filter);
            }
        };
        JMenuItem item = new JMenuItem("All Movable");
        item.addActionListener(action);
        menu.add(item);

        action = new AbstractAction() {
            public void actionPerformed(ActionEvent e) {
                network.addMovableFilter(filter);
            }
        };
        item = new JMenuItem("Fix Even Node");
        item.addActionListener(action);
        menu.add(item);
    }
});
  
```

```

        return menu;
    }
});

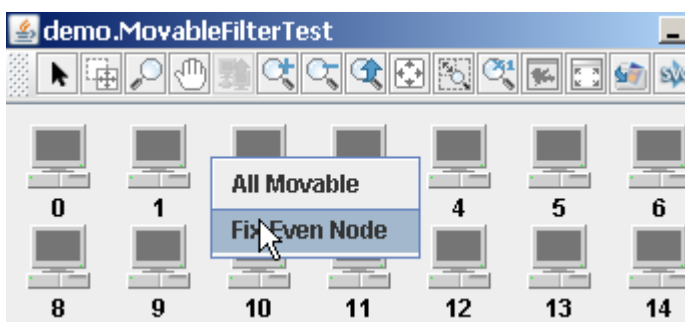
```

Finally create a lot of nodes and add them into network component. We use order index as the node label/name:

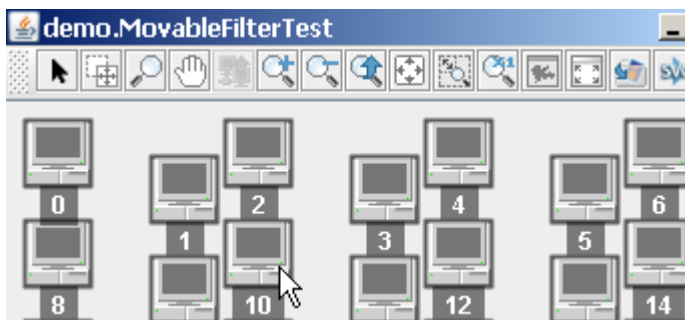
```

for(int i=0;i<40;i++){
    Node node=new Node();
    node.setName(""+i);
    node.setLocation(10+50*(i%8),10+50*(i/8));
    box.addElement(node);
}

```



Set movable filter



Move some nodes with movable filter

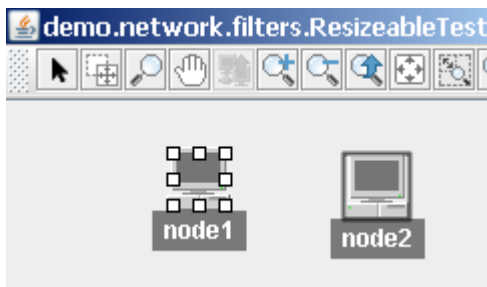
Using ResizableFilter

Function: whether a Element is resiazable. The default value is not resizable.

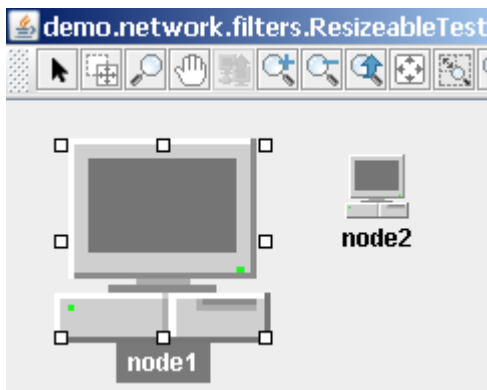


```

ResizableNode node1=new ResizableNode("node1");
ResizableNode node2=new ResizableNode("node2");
box.addElement(node1);
box.addElement(node2);
network.setResizableFilter(new ResizableFilter(){
    public boolean isResizable(Element element) {
        if(element.getID().equals("node1")){
            return true;
        }
        return false;
    }
});
    
```

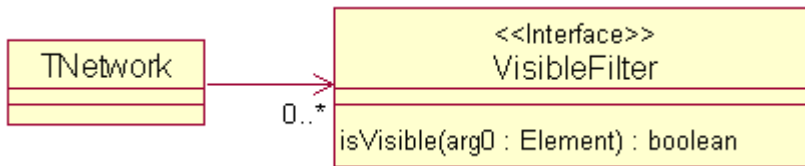


Node1 is resizable



Using VisibleFilter

Network component can control each element's visibility by VisibleFilter:



VisibleFilter interface define method "isVisible" to determine whether element is visible. One network component can set more than one VisibleFilters. Element will be displayed only if all visible filters allow this element be displayed.

You can add and remove visible filters by methods "addVisibleFilter" and "removeVisibleFilter" of TNetwork. When visible filters added or removed, whole network canvas will repainted to make sure all elements can displayed properly.

In order to demonstrate this, we make a sample to show you how visible filter working:

Firstly, we create lot of nodes on network and give each node the number as its name. Then we set the network popup menu generator to provide two menu items: one is used to hide all even index nodes, another used to reset to display all nodes.

Now define a new visible filter, which used to hide all even index nodes:

```

final VisibleFilter filter = new VisibleFilter() {
    public boolean isVisible(Element element) {
        if (element.getName() != null &&
            Integer.valueOf(element.getName()).intValue() % 2 == 0) {
            return false;
        }
        return true;
    }
};
  
```

Then we define a popup menu generator to create the two menu item:

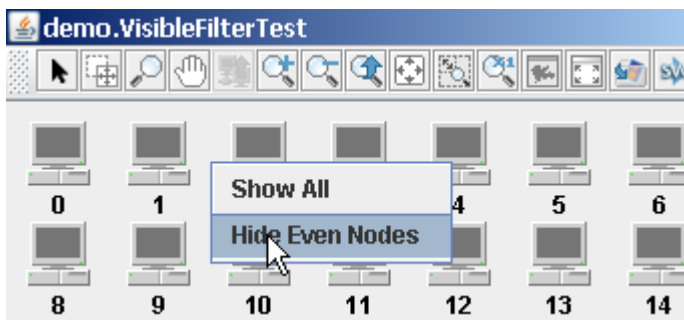
```

network.setPopupMenuGenerator(new PopupMenuGenerator() {
    public JPopupMenu generate(TView tview, MouseEvent mouseEvent) {
        final TNetwork network=(TNetwork)tview;
        JPopupMenu menu = new JPopupMenu();
        Action action = new AbstractAction() {
            public void actionPerformed(ActionEvent e) {
                network.removeVisibleFilter(filter);
            }
        };
        JMenuItem item = new JMenuItem("Show All");
        item.addActionListener(action);
        menu.add(item);
        action = new AbstractAction() {
            public void actionPerformed(ActionEvent e) {
                network.addVisibleFilter(filter);
            }
        };
        item = new JMenuItem("Hide Even Nodes");
        item.addActionListener(action);
        menu.add(item);
        return menu;
    }
});
  
```

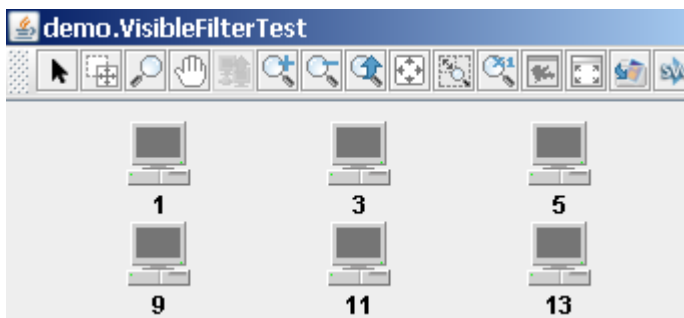
```
});
```

Finally create a lot of nodes and add them into network component. We use order index as the node label/name:


```
for(int i=0;i<40;i++){
    Node node=new Node();
    node.setName(""+i);
    node.setLocation(10+50*(i%8),10+50*(i/8));
    box.addElement(node);
}
```



Normal display



Click "Hide Even Node" menu, and all even nodes hided

 when node hidden, all related links will be hidden automatically without further endeavor.

Using Generator

Generator is an Interface, it only has one method and you can input a value and then it will return a value. TNetwork uses many Generators to generate some properties. Generator is the general control for these property values. For instance, the ElementLabelGenerator used to generate element label text. Following are some generators used in TNetwork:

ElementLabelGenerator	General control for all element label text
AlarmLabelGenerator	General control for all element alarm label text
ElementToolTipTextGenerator	General control for all element tooltip text
MessageContentGenerator	General control for all element message content text
ElementSelectColorGenerator	General control for all element selection border color
PopupMenuGenerator	General control for all popup menu
AlarmColorGenerator	General control for all alarm color
ElementOutlineGenerator	General control for all elements outline color
ElementBodyColorGenerator	General control for all elements body color

```

network.setElementLabelGenerator(new Generator(){
    public Object generate(Object object) {
        //the object here is a Element type
        Element element=(Element)object;
        if(element.isSelected()){
            return ((Element)object).getName();
        }
        return null;
    }
});

network.setPopupMenuGenerator(new PopupMenuGenerator(){
    public JPopupMenu generate(TView tview, MouseEvent mouseEvent) {
        //The tview here is a TNetwork
        TNetwork network=(TNetwork)tview;
        //get mouse pointed Element from mouseEvent
        Element element=network.getElementLogicalAt(mouseEvent.getPoint());
        //...
        JPopupMenu menu=new JPopupMenu();
        return menu;
    }
});

```

Using Link Layouter

By default, Link is a straight line on network component. But you can use a link layouter for a network to specify the path for each Link. Link layouter is an interface, network call the layouter's method 'getLinkPath' to get link path for each Link object.

Following link layouter use a curve line for all links:

```
network.setLinkLayout(new LinkLayouter() {
    public void layout() {}

    //return path for each Link object.
    public GeneralPath getLinkPath(LinkUI linkUI) {
        GeneralPath customPath = new GeneralPath();
        Point p1 = linkUI.getFromPoint();
        Point p2 = linkUI.getToPoint();

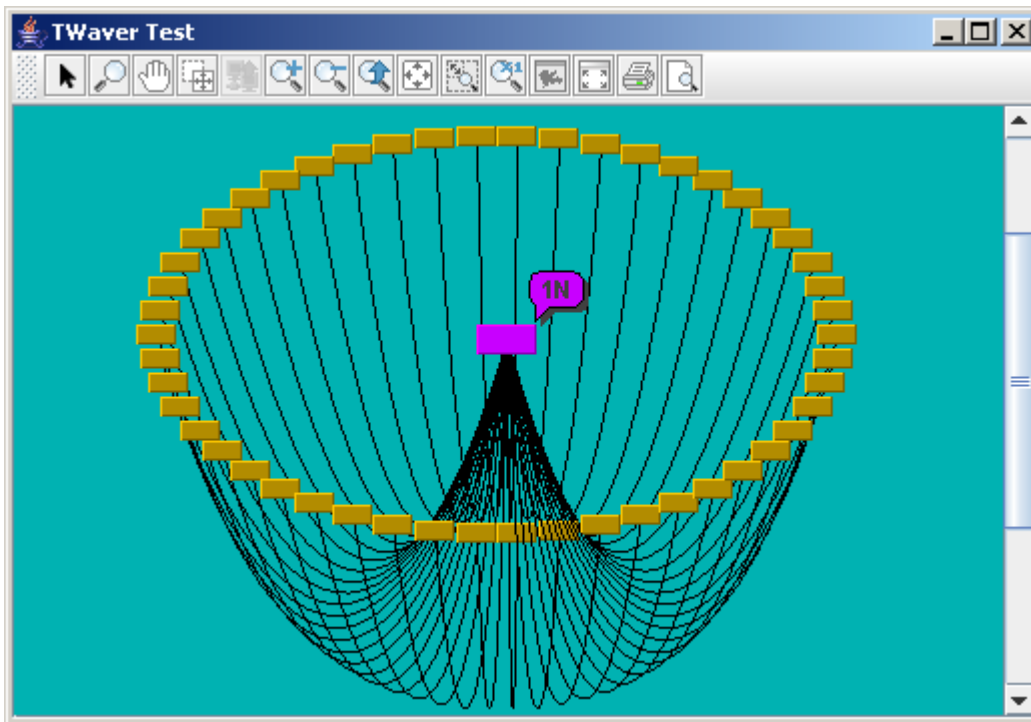
        customPath.moveTo(p1.x, p1.y);
        int yOffset = 300;
        //create a curve path.
        customPath.curveTo(p1.x, p1.y, (p1.x + p2.x) / 2, (p1.y + p2.y) / 2 + yOffset, p2.x, p2.y);
        return customPath;
    }
});
```

Then we put a lot of nodes and links on network component to see the layout result:

```
Background background= new ColorBackground(Color.cyan.darker());
network.setNetworkBackground(background);
Node center = new Node() {
    public int getWidth() { return 30; }
    public int getHeight() { return 15; }
};
center.setImage(TWaverConst.BLANK_IMAGE);
center.getAlarmState().setNewAlarmCount(AlarmSeverity.INDETERMINATE, 1);
center.setLocation(200, 200);
box.addElement(center);

int count = 50;
for (int i = 0; i < count; i++) {
    Node node = new Node() {
        public int getWidth() { return 20; }
        public int getHeight() { return 10; }
    };
    node.setImage(TWaverConst.BLANK_IMAGE);
    node.putRenderColor(Color.orange.darker());
    int x = center.getLocation().x + (int) (170 * Math.cos(Math.PI * 2 / count * i));
    int y = center.getLocation().y + (int) (100 * Math.sin(Math.PI * 2 / count * i));
    node.setLocation(x, y);
    box.addElement(node);

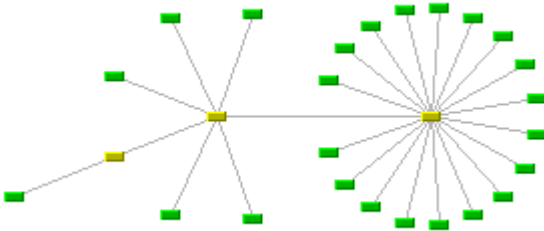
    Link link = new Link(node, center);
    link.putLinkWidth(1);
    link.putLinkColor(Color.black);
    link.putLinkOutlineWidth(0);
    box.addElement(link);
}
```

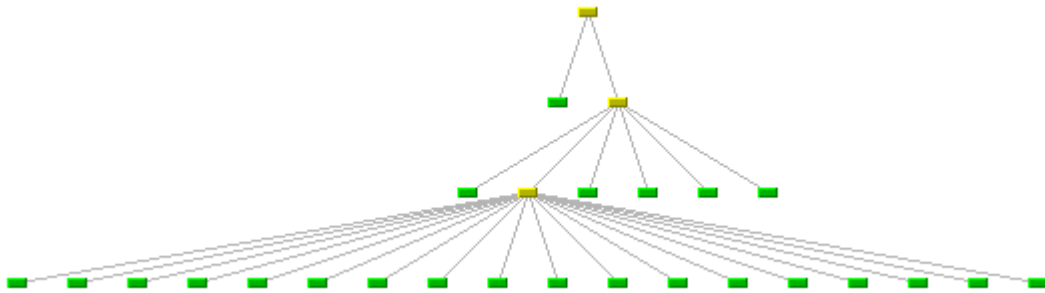
Using Network Layout

Some topology applications require automatic positioning of the network structure accordingly to general layout rules. Network component provides following layout mode:

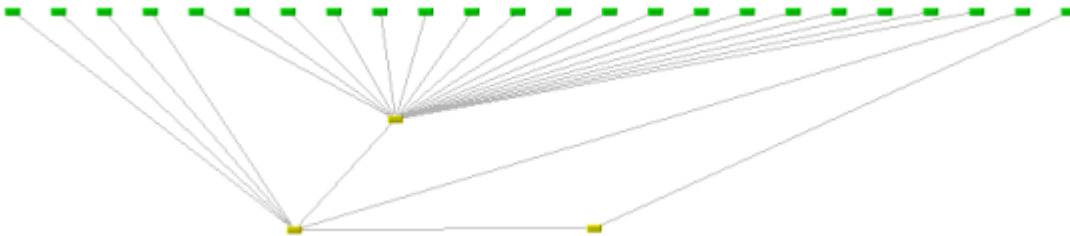
- TWaverConst.LAYOUT_CIRCULAR = 1



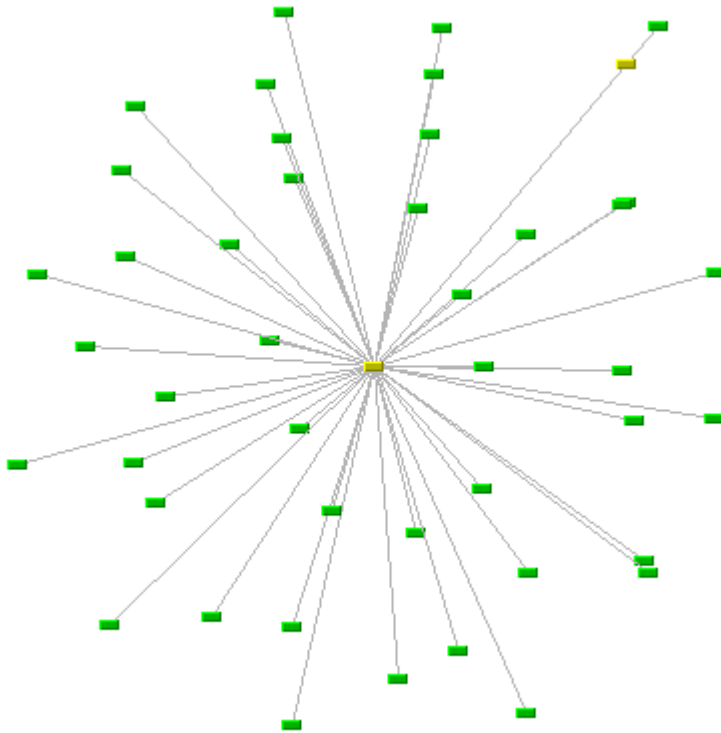
- TWaverConst.LAYOUT_TREE = 2



- TWaverConst.LAYOUT_HIERARCHIC = 3



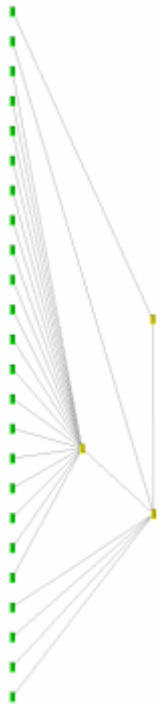
- TWaverConst.LAYOUT_SYMMETRIC = 4



- TWaverConst.LAYOUT_EAST = 5



- TWaverConst.LAYOUT_WEST = 6



Use method `TNetwork.doLayout(int type, boolean animate, Runnable runnable, Generator sizeGenerator)` to layout network:

Type: `TWaverConst.`

```
public final static int LAYOUT_CIRCULAR = 1;
public final static int LAYOUT_TREE = 2;
public final static int LAYOUT_HIERARCHIC = 3;
public final static int LAYOUT_SYMMETRIC = 4;
public final static int LAYOUT_EAST = 4;
public final static int LAYOUT_WEST = 4;
```

Animate: Whether layout the graph animated

Runnable: Will be invoked in Swing thread when layout finished

SizeGenerator: Return the size for Element


```
//Layout network with circular layout in animated mode.
network.doLayout(
    TWaverConst.LAYOUT_CIRCULAR,
    true,
    new Runnable() {
        public void run() {
            JOptionPane.showMessageDialog(network, "layout finished");
        }
    },
    new Generator(){
        public Object generate(Object object) {
            Element element = (Element) object;
            if (element instanceof Group) {
                if (((Group)element).isExpand()){
                    Rectangle bounds = network.getElementBounds(element);
                    bounds.grow(-bounds.width/4, -bounds.height/4);
                    return new Dimension(bounds.width, bounds.height);
                }
            }
            return new Dimension(element.getWidth() + 20, element.getHeight() + 20);
        }
    }
);
```

```
);
```

Operations for Network

- [Full Screen Network](#)
- [Network Overview](#)
- [Using Dragging Speed](#)
- [Using Magnifier](#)
- [Using Network Zoomer](#)

Full Screen Network

Click button  on network toolbar to full screen the network canvas, and back to normal display by clicking again. You can also press F11 key to do the same thing.
You can also control the full screen mode by APIs:

<code>network.showFullScreen();</code>	Enable full screen mode
<code>network.exitFullScreen();</code>	Exit full screen mode

Network Overview

Network Overview is a Swing component which displays the full range data of network. You can click the button on network toolbar to show/hide overview window. By default, Overview is float on the network canvas. As an instance of Swing JComponent, you can put Network Overview component to anywhere you want, such as a standalone window, or the right-bottom corner.


Using Dragging Speed

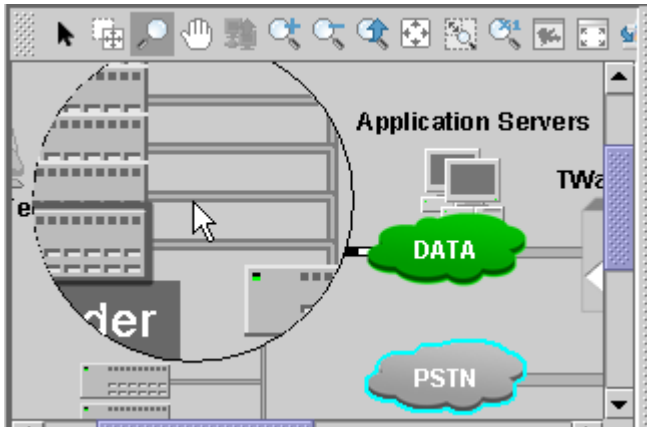
Elements on Network can be dragged & dropped. When the elements dragging out the network view port, network will scroll the view to ensure the in dragging elements visible in view. You can specify the network scroll speed when elements are being dragged out of the network's view. Call method `TNetwork.setDraggingSpeed` and pass into a `DraggingSpeed` instance which represents the dragging speed. `DraggingSpeed` also defines several static constants for use.

Constant	Value	Default	Description
<code>DraggingSpeed.SLOW</code>	5		Slow speed
<code>DraggingSpeed.NORMAL</code>	10	Y	Normal speed
<code>DraggingSpeed.HIGH</code>	20		High speed


TWaver use `DraggingSpeed.NORMAL` as the network default dragging speed.

Using Magnifier

You can view the network canvas with a magnifier. Click the button  on the network toolbar to take out the magnifier. With the magnifier mode, a magnifier will always come with the mouse cursor. See following image.



In the magnifier mode, you can click, select or do everything like normal. When any mouse button was clicked, the magnifier will be invisibled, unless the mouse was released again.

Press button  to back the default interaction mode.

Following codes show you how to enable the magnifier when press control key down:

```
network.getCanvas().addKeyListener(new KeyAdapter(){
    boolean isMagnifierShowing=false;
    public void keyPressed(KeyEvent e) {
        //if "ctrl" key pressed, enable magnifier mode.
        if(e.isControlDown()&&!isMagnifierShowing){
            InteractionMode m=InteractionModeFactory.getMagnifierMode(network);
            network.setInteractionMode(m);
            isMagnifierShowing=true;
        }
    }
    public void keyReleased(KeyEvent e) {
        //if release "ctrl" key, reset network
        if(!e.isControlDown()){
            InteractionMode m=InteractionModeFactory.getDefaultMode (network);
            network.setInteractionMode(m);
            isMagnifierShowing=false;
        }
    }
});
```

Using Network Zoomer

TWaver defines Zoomer interface used to zoom Network canvas.
Class `twaver.network.zoom.Zoomer`

 <<Interface>> Zoomer
<ul style="list-style-type: none"> ● <code>getMaxZoom() : double</code> ● <code>setMaxZoom(in maxZoom : double) : void</code> ● <code>getMinZoom() : double</code> ● <code>setMinZoom(in minZoom : double) : void</code> ● <code>getNetwork()</code> ● <code>getZoom() : double</code> ● <code>setZoom(in zoom : double) : void</code> ● <code>zoomIn() : void</code> ● <code>zoomOut() : void</code> ● <code>wheelZoomIn() : void</code> ● <code>wheelZoomOut() : void</code> ● <code>zoomReset() : void</code> ● <code>zoomBack() : void</code> ● <code>zoomToRect(in rect : Rectangle) : void</code> ● <code>addZoomListener() : void</code> ● <code>removeZoomListener() : void</code> ● <code>zoomToOverview() : void</code> ● <code>zoomToOverview(in gapX : int,in gapY : int) : void</code> ● <code>getWheelZoomIncrement() : double</code> ● <code>setWheelZoomIncrement(in wheelZoomIncrement : double) : void</code> ● <code>getZoomIncrement() : double</code> ● <code>setZoomIncrement(in zoomIncrement : double) : void</code>

TWaver `twaver.network.zoom.PhysicalZoomer` implements the interface `Zoomer`, it provides the ability for zoom.

TWaver Network toolbar provides several buttons for zoom operations:



Operate zoom by APIs

Following codes show you how to control the zoom of Network. It add a `MouseWheelListener` on network canvas, get `Zoomer` of Network and zoom it in or out:

```
network.getCanvas().addMouseWheelListener(new MouseWheelListener(){
public void mouseWheelMoved(MouseWheelEvent e) {
    if (e.getWheelRotation() > 0) {
        network.getZoomer().wheelZoomOut();
    } else {
        network.getZoomer().wheelZoomIn();
    }
}
```

```
}  
});
```

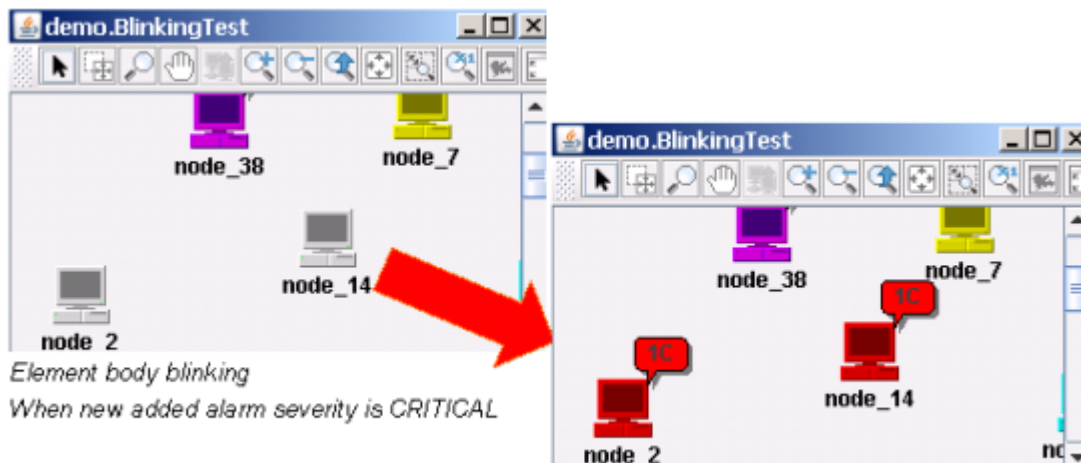
Using Blinking Rule

Class `BlinkingRule` defines the rule to control Element blinking. Element blinking includes Element body blinking and Element border blinking. Developers can extend new blinking rule to customize the blinking logic.

```
class Rule extends BlinkingRule{
    public boolean isBodyBlinking(Element element) {
        // blink when element's new alarm severity
        // is more urgent than warning level
        AlarmSeverity severity = element.getAlarmState().getHighestNewAlarmSeverity();
        if(severity == AlarmSeverity.CRITICAL){
            return true;
        }
        return false;
    }
    public boolean isOutlineBlinking(Element element) {
        return false;
    }
}
```

Set the blinking rule to a Network component:

```
Rule ruler = new Rule();
network.setBlinkingRule(ruler);
```



More Network Component Features

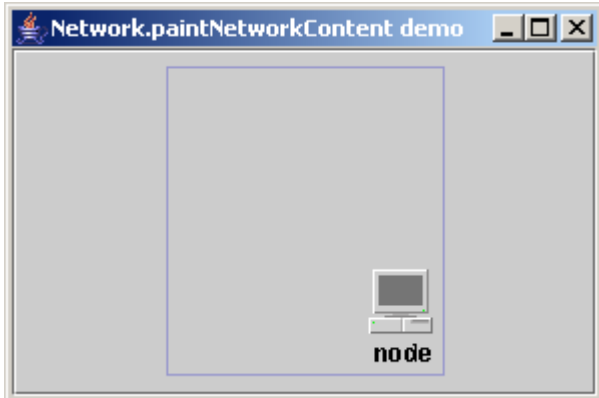
Use Method paintNetworkContent to Paint Network Canvas

`public void paintNetworkContent(Graphics g, double xScale, double yScale, boolean considerClipBounds)`

This method can be used to paint network canvas to any Swing component or an Image object. Below example will paint a network canvas on an image, then set this image to a Swing JButton.

```
TDataBox box=new TDataBox();
TNetwork network=new TNetwork(box);
Node node =new Node();
node.setName("node");
node.setLocation(100,100);
box.addElement(node);
BufferedImage bi = new BufferedImage(network.getCanvas().getWidth(),
network.getCanvas().getHeight(),BufferedImage.TYPE_4BYTE_ABGR);
network.paintNetworkContent(bi.getGraphics(), 1, 1, false);
JButton btnNetwork=new JButton(new ImageIcon(bi));
JFrame frame = new JFrame();
frame.setTitle("Network.paintNetworkContent demo");
frame.getContentPane().add(btnNetwork);
frame.setSize(300, 200);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
TWaverUtil.centerWindow(frame);
frame.setVisible(true);
```

Run this example:



Some Methods for Interaction

`enableDoubleClickToUp`

Enable the action that double click network background blank area to go upper SubNetwork. By default this is true.

`enableEnterEmptySubNetwork`

Whether can enter an empty SubNetwork by double clicking it. By default this is true.

`enableAutoScroll`

Whether the network canvas scroll bars are scrolled by element move. This is, when you drag some elements and move out of scroll view port, whether scroll the view port automatically. Default is true.



If set to false, then you can not drag and move elements out of view port.

`showLinkBundleHandler`

Determines whether the link bundle handler should be shown on network canvas.

enableAttachmentDefaultAction

Whether dispatch mouse event from attachment to the Element. By default this is false, that means attachment and element has their own events. If set to true, the attachment will dispatch its mouse events to element. In this case, the double clicks on attachment will be the same as double clicking element body.

dragCriticalValue

This is the threshold distance to trigger the drag action. That is, only drag elements longer than this distance, the drag action will be start. If not, do nothing.

isElementTransparentAreaSelectable

As you know all TWaver Node elements can have a picture. When you are using a picture with partial transparent area, you can use this method to tell TWaver whether should select the element when user click the transparent area. The default value is false.

Some Methods for Full-Screen

- showFullScreen - full screen mode
- exitFullScreen - exit full screen mode
- setFullScreenAttachment - The Swing component you want display on the top of screen when you in full-screen mode. By default this is null. In this case, TWaver will put network toolbar on the top of screen, if the network has toolbar. If you set a new component by this method, TWaver will display the component you set on the top of the screen when full-screen enabled.
- isFullScreen - check whether now is in full-screen mode
- setApplyBackgroundThroughSubNetwork - Whether use network background for SubNetwork, when the SubNetwork does not have own background.

For more information, please check TWaver JavaDoc contained in your TWaver Java Product CD.

Using Tree Component

TTree component is one of the major graphic components supplied with TWaver. It provides a hierarchical view of the elements contained in a DataBox. A tree has one root node, from which all the other nodes descend. Each node in the tree is associated with an element instance. It has a number of graphic properties like label, icon, tooltip etc. These properties are set according to the values of one or more attributes of the element.

- [Check-Box Tree Node](#)
- [Creating Tree Component](#)
- [Customizing Tree Node Handle Icon](#)
- [Customizing Tree Node Renderer](#)
- [Introducing the Tree Component](#)
- [Iterating Tree Node](#)
- [Lazy Loading](#)
- [Size of Tree Node](#)
- [Sorting the Tree Node](#)
- [Tooltip of Tree Node](#)
- [Tree Node Double Clicked Listening](#)
- [Tree Node Drag & Drop](#)
- [Tree Node Index Control](#)
- [Tree Popup Menu](#)
- [Tree Root Shift](#)
- [TWaver Tree Node](#)
- [Using Tree Generators](#)

Check-Box Tree Node

Use Selection Mode

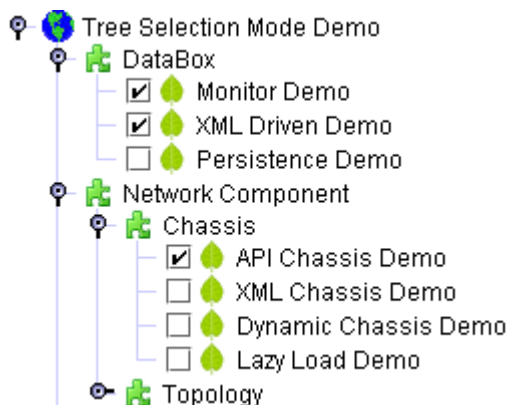
TWaver tree component allows you to choose check-box style for the selection. Use `TTree.setTTreeSelectionMode(int mode)` to switch the style.

- `TTree.DEFAULT_SELECTION`: normal selection style.
- `TTree.CHECK_SELECTION`: check-box selection style.
- `TTree.CHECK_CHILDREN_SELECTION`: check-box selection. The children node selection state will be changed with parent's selection.
- `TTree.CHECK_DESCENDANT_SELECTION`: check-box selection. The children and descendant node selection state will be changed with parent's selection.
- `TTree.CHECK_DESCENDANT_ANCESTOR_SELECTION`: check-box selection. The children and descendant node selection state will be changed with parent's selection. The selection state of the parent node also will be changed with its selection state.

Use CheckableFilter

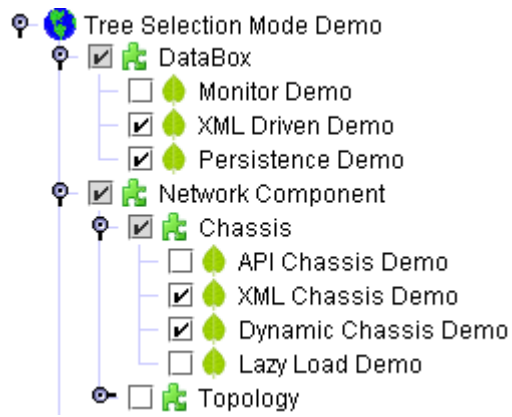
If you just want make several specified tree node checkable, you can use Checkable Filter.

```
//Create and use a checkable filter.
tree.setCheckableFilter(new CheckableFilter(){
    public boolean isCheckable(Element element) {
        //return true if it can be checked.
        if(...) {
            return true;
        } else {
            return false;
        }
    }
});
```



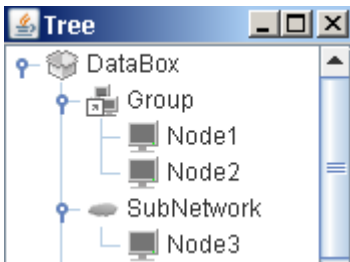
Use Tristate CheckBox

TWaver Tree node can be tristate. Use method `setEnabledTristateCheckBox` to enable tristate function. In the case of tristate enabled, the tree node looks like this:



Creating Tree Component

This section shows you how to create the following basic tree featuring a computer network.



To create this tree view, we need to go through these steps:

- Create TDataBox instance
- Create tree component
- Connect tree to the DataBox
- Create elements and add them into the DataBox
















```
//Create a new DataBox instance.
TDataBox box=new TDataBox();
//Create a new tree component with this box.
TTree tree=new TTree(box);
//Or you can set the DataBox later as follows:
//tree.setDataBox(box);

//Create group and subnetwork
Group group=new Group();
box.addElement(group);
SubNetwork subnetwork=new SubNetwork();
box.addElement(subnetwork);

//Create children nodes.
Node node1=new Node();
node1.setName("Node1");
node1.setParent(group);
box.addElement(node1);
Node node2=new Node();
node2.setName("Node2");
node2.setParent(group);
box.addElement(node2);
Node node3=new Node();
node3.setName("Node3");
node3.setParent(subnetwork);
box.addElement(node3);
```

Customizing Tree Node Handle Icon

Here is a picture of some tree nodes, as drawn by the Java, Windows, and Mac OS look and feel implementations:

 The Java Series  Books for Java I  Books for Java I  The Java Vir  The Java La	 The Java Series  Books for Java Pr  Books for Java Itr  The Java Virt  The Java Lan	 The Java Series  Books for Java Pr  Books for Java In  The Java Virt  The Java Lan
Java look and feel	Windows look and feel	Mac OS look and feel

As the preceding figures show, a tree conventionally displays an icon for the tree node handle and different Look and Feel will provides different handle icon. You can customize it in your Look and Feel, or you can use TWaver related methods to change it:

```
tree.setCollapsedIcon(TWaverUtil.getIcon("/demo/sheet/nested/collapse.png"));
tree.setExpandedIcon(TWaverUtil.getIcon("/demo/sheet/nested/expand.png"));
```



Customizing Tree Node Renderer

As a subclass of Swing JTree, you can set a new renderer for the tree component. The default implementation of TTree has created default renderer to draw each element on the tree component. The default renderer consider element name, alarm state, selection state, tooltip, icon and parent relationship.

To create a new tree node renderer and considering above aspects is not an easy task. So one better way is to create a renderer extends from the default tree node renderer implementation - twaver.tree.ElementNode class. It derived from Swing DefaultMutableTreeNode, so you can customize it to draw the tree node as your need.

By default tree node use Element.getIcon to get the image icon and display it on the tree node. Normally it is used to indicate the Element's type. You can also customize the type icon. Call Element.setCustomIcon method to specify a new type icon which will be displayed on tree node. If null was set, the default icon will be used.

If you want to use a java.swing.Icon object instead of the icon URL string to change the element icon on Tree component, use following code:

```
//Create an Icon object.
Icon icon = ....
//Change some element icon
Object key = TWaverConst.PROPERTYNAME_ELEMENT_TREE_ICON;
element.putClientProperty(key, icon);
or
element.putElementTreeIcon(icon);
```

Introducing the Tree Component

The tree component of TWaver extends Swing `JTree` object and implements `TView` interface of TWaver. It is a `JTree` extension and you can directly use it as a `JTree`. It connects to a `DataBox`, from which it obtains the all elements to be displayed. By default, the tree displays all the elements contained in the `DataBox`. However, it is also possible to restrict the contents displayed by selecting the root nodes, or by applying a visible filter. Elements that have a parent are displayed as the child node under the parent.

`TTree` provides the API for the most common uses of the tree component, such as:

- Setting or retrieving the associated `DataBox`: `getDataBox()` and `setDataBox()`
- Setting tree selection mode.
- Changing the root node of the tree.
- Filtering the tree nodes.
- Sorting the tree nodes by sort comparator.

Iterating Tree Node

Normally you can iterator all elements by `TDataBox.iterator` method. If you want iterate elements with the order of the tree displayed, use following methods:

- `TDataBox.breadthFirstEnumeration()`: iterate box by BreadthFirst rule
- `TDataBox.breadthFirstEnumeration(Element root)`: iterate box by BreadthFirst rule from the specified root element
- `TDataBox.depthFirstEnumeration()`: iterate box by DepthFirst rule
- `TDataBox.depthFirstEnumeration(Element root)`: iterate box by DepthFirst rule from the specified root element

Lazy Loading

The tree component allows you to load data lazily. This is important in the case of displaying huge network data. Just imagine creating a tree to display the file system, it is not good to load all files to the tree when it displayed, we hope files will be loaded only when the folder opened in the first time.

Use interface `twaver.tree.LazyLoader` to define lazy loader. You must implements the methods:

- `load`: Perform the data loading action.
- `isLoading`: Whether the data of this tree node has been loaded.
- `isLeaf`: Whether this tree node is a leaf. For leaf node, no load action performed.

Now we try to create a tree component with lazy loader to view the file system.

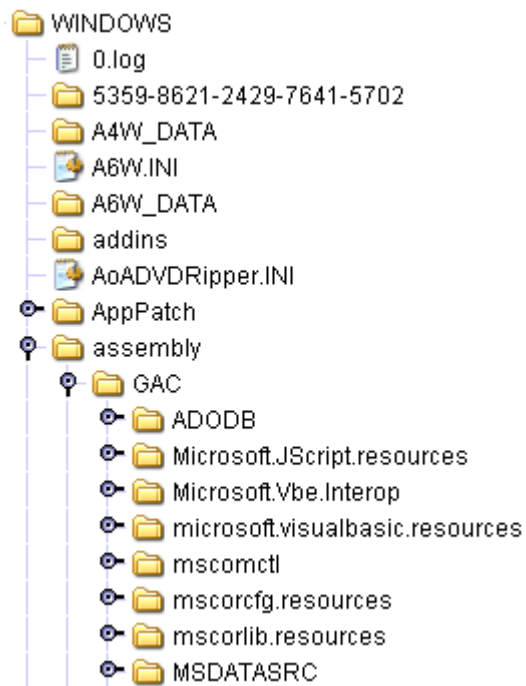
```
//set lazy loader
tree.setLazyLoader(new LazyLoader(){
    //load children
    public void load(Element element) {
        FileElement fileElement = (FileElement)element;
        File file = fileElement.getFile();
        if(file.isDirectory()){
            File[] files = FileElement.fileSystemView.getFiles(file, true);
            if (files != null) {
                for (int i = 0; i < files.length; ++i) {
                    FileElement child = new FileElement(files[i]);
                    fileElement.addChild(child);
                    box.addElement(child);
                }
            }
        }
        fileElement.setLoaded(true);
        label.setText("box size:" + box.size());
    }

    //check whether children is loaded.
    public boolean isLoading(Element element) {
        FileElement fileElement = (FileElement)element;
        return fileElement.isLoading();
    }

    //check whether the element has child
    public boolean isLeaf(Element element) {
        Boolean isLeaf = (Boolean)element.getClientProperty("isLeaf");
        if(isLeaf == null){
            FileElement fileElement = (FileElement)element;
            File file = fileElement.getFile();
            if(file.isDirectory()){
                File[] files = FileElement.fileSystemView.getFiles(file, false);
                if (files != null && files.length > 0) {
                    isLeaf = Boolean.FALSE;
                }else{
                    isLeaf = Boolean.TRUE;
                }
            }else{
                isLeaf = Boolean.TRUE;
            }
        }
        //cache result
        element.putClientProperty("isLeaf", isLeaf);
    }
    return isLeaf.booleanValue();
}
```



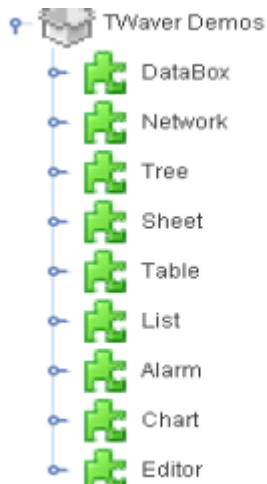
```
});
```



Size of Tree Node

The size of the TTree icon can be set as follows

```
tree.setIconWidth(30);
tree.setIconHeight(30);
```



Sorting the Tree Node

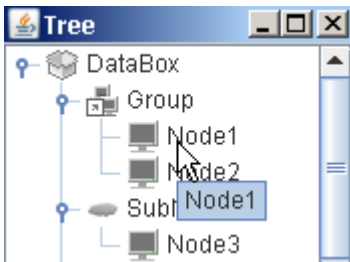
The tree component allows you to sort the displayed nodes. To do so, put an instance of `java.util.Comparator` to the tree component by using the method `setSortComparator`. The `Comparator.compare()` method will be called for pairs of `Element` instances in the `DataBox` to establish a sort order. Objects are ordered locally.

All the objects are sorted again when a new comparator is set. If there is no comparator (which is the default case), the nodes are displayed in random order. Please note that the same set of objects is always displayed in the same order, even if no sort order is defined.

To retrieve the current sort order, use the method `getsortComparator`.

Tooltip of Tree Node

By default TTree node will display its related element tooltip text.



Disable tooltip text by calling `TTree.setElementToolTipDisplayable (false)` method.

Tree Node Double Clicked Listening

Use method `TTree.addElementNodeDoubleClickedActionListener` to listen mouse double click event of the tree node.

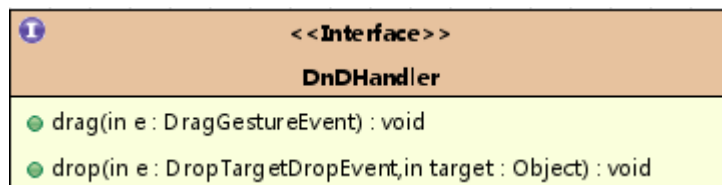
Tree Node Drag & Drop

TWaver Tree component provides support for node drag/drop. When you are doing this, you are changing the node's parent relationships.

Class DnDHandler has two methods:

```
public void drag(TTree tree, DragGestureEvent e);
public void drop(TTree tree, DropTargetDropEvent e, Object target);
```

The "target" is the selected Element.



When tree node dragging start, the drag method will be called, where you can set the drag gesture (TWaver default gesture is MOVE and drag data type).

When tree node receive dropped node, method drop will be called to execute your actions.

Class twaver.tree.DefaultDnDHandler implements interface DnDHandler. This default implementation will change dragged Element's parent when drag/drop finished.

Please note that in default TWaver Tree node is not draggable. You need to call this to enable the drag/drop:

```
setElementDraggable(boolean isElementDraggable)
```

Here is an example to customize drag-and-drop to implements a drag-to-copy function.

```

public class TreeDNDHandlerTest extends JPanel {
    TDataBox box = new TDataBox();
    TTree tree = new TTree(box);
    public TreeDNDHandlerTest() {
        init();
        //enable drag-and-drop for tree node
        tree.setElementDraggable(true);
        //customize dnd handle
        tree.setDnDHandler(new DefaultDnDHandler() {
            private boolean isCopy = false;
            public void drag(final TTree tree, DragGestureEvent e) {
                e.startDrag(null, TRANSFERABLE, null);
            }
            public void drop(TTree tree, DropTargetDropEvent e, Object target) {
                //drag enter. Check whether is copy mode now.
                if (e.getDropAction() == DnDConstants.ACTION_COPY) {
                    System.out.println("copy mode");
                    box.copySelection();
                    //set to copy mode.
                    isCopy = true;
                }
                //If not copy mode, then use default action of twaver.
                super.drop(tree, e, target);
            }
        });
    }
    protected void dropProcess(TTree tree, Element targetElement) {
        if (isCopy) {
  
```

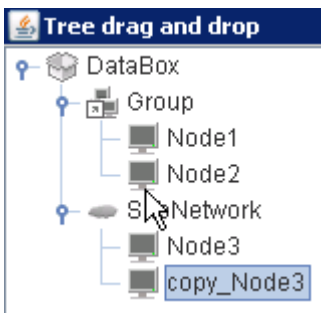
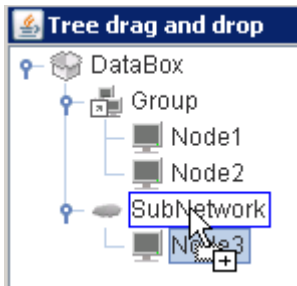
```

Collection selectedElements = tree.getDataBox()
    .getSelectionModel().getAllSelectedElement();
if (selectedElements == null || selectedElements.isEmpty()) {
    return;
}
//target can't included in sources.
if (selectedElements.contains(targetElement)) {
    return;
}
//paste dragged element to target element.
pasteElements(targetElement,tree.getDataBox());
} else {
    super.dropProcess(tree, targetElement);
}
//reset the copy mode to false.
isCopy = false;
}
});
this.setLayout(new BorderLayout());
this.add(new JScrollPane(tree), BorderLayout.CENTER);
tree.expandAll();
}
public static List pasteElements(Element parentElement, TDataBox box) {
    List pastes = new ArrayList();
    List list = TWaverUtil.getCopyElements();
    if (list != null && list.size() > 0) {
        int offset = TWaverUtil.getPasteOffset() + 5;
        for (int i = 0; i < list.size(); i++) {
            Element element = (Element) list.get(i);
            Element newElement = (Element) element.copy(box);
            newElement.setName("copy_" + (element.getName() == null ? "" : element.getName()));
            System.out.println(newElement.getName());
            Point p = element.getLocation();
            if (p == null) {
                p = new Point(0, 0);
            }
            newElement.setLocation(p.x + offset, p.y + offset);
            if (parentElement != null) {
                newElement.setParent(parentElement);
            }
            box.addElement(newElement);
            pastes.add(newElement);
        }
        box.getSelectionModel().setSelection(pastes);
    }
    return pastes;
}
private void init() {
    Group group = new Group();
    box.addElement(group);
    SubNetwork subnetwork = new SubNetwork();
    box.addElement(subnetwork);
    Node node1 = new Node();
    node1.setName("Node1");
    node1.setParent(group);
    box.addElement(node1);
    Node node2 = new Node();
    node2.setName("Node2");
    node2.setParent(group);
    box.addElement(node2);
    Node node3 = new Node();
    node3.setName("Node3");
    node3.setParent(subnetwork);
    box.addElement(node3);
}
public static void main(String[] args) {

```

```
JFrame frame = new JFrame();
frame.setTitle("Tree drag and drop");
frame.getContentPane().add(new TreeDNDHandlerTest(), BorderLayout.CENTER);
frame.setSize(900, 700);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
TWaverUtil.centerWindow(frame);
frame.setVisible(true);
}
```

Press ctrl key, drag a tree node, drop to make a copy:



Tree Node Index Control

You can control the tree node index by invoking `TDataBox.moveTo**` methods. Please note that these methods only effect the position of TWaver Tree node or TWaver List item. It does not affect the painting index of element on network component.

These methods are:

- `TDataBox.moveToUp`: move tree node to the upper position of its parent
- `TDataBox.moveToDown`: move tree node to the lower position of its parent
- `TDataBox.moveToTop`: move tree node to the first position of its parent
- `TDataBox.moveToBottom`: move tree node to the last position of its parent

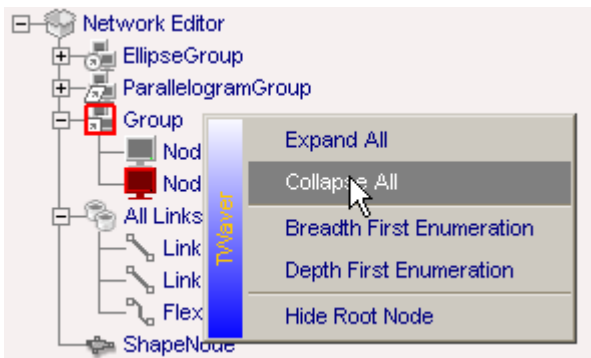
Use `twaver.DataBoxSequenceListener` to monitoring the tree node index change event. Method `hierarchyChanged` will be called back when the tree node index is changed.

Tree Popup Menu

Tree component provides the ability to show a customized popup menu by using popup menu generator. The menu generator can be used in Network component, so it's possible to share one popup menu generator instance between Tree component and Network component.

```
PopupMenuGenerator generator=new MyPopupMenuGenerator();
//share the same popup menu generator instance for network and tree
network.setPopupMenuGenerator(generator);
tree.setPopupMenuGenerator(generator);
```

TWaver Tree component creates a default popup menu generator as follows:



Set it to null to disable popup any menus.

Tree Root Shift

Once a Tree component connects to DataBox, it will display the whole hierarchical data of the DataBox. Tree will display a default root tree node represent the DataBox itself, all top level elements will display as its children. However, you can shift the tree root to any other element which contains in the DataBox. You can specify the root element when create Tree component, or by the method `TTree.setRootElement` in the runtime.

TWaver Tree Node

There are two kinds of tree node in TWaver: `twaver.tree.DataBoxNode` and `twaver.tree.ElementNode`. `DataBoxNode` connects to a `DataBox`, `ElementNode` connects to a `Element`. Some useful methods about tree node:

- Show/hide tree root node, `tree.setRootVisible(boolean rootVisible);`
- Get tree root node, `tree.getRootNode();`
- Set `DataBox` node icon, `tree.setDataBoxIconURL("/root.png")`
- Set `DataBox` node label text, `tree.getDataBox().setName("name");`
- Set tree root node, `tree.setRootElement(rootNode);`
- Set tree node label text, By default use `element.getName()` and change by method `setName(String name)`
- Set tree node tooltip text, By default use `element.getToolTipText()`, and changed by method `setToolTipText(String tip)`
- `twaver.tree.DataBoxNode` , `twaver.tree.ElementNode` By default use `((AbstractElement)element).getElementTreeIcon()`, and changed by `((AbstractElement)element).putElementTreeIcon(Icon elementTreeIcon)`

Using Tree Generators

TWaver Tree defines three kinds of generators: tooltip text generator (elementToolTipTextGenerator), element label text generator (elementLabelGenerator) and element icon generator (elementIconGenerator). They are used to control tree node tooltip text, tree node label text and tree node icon.

We can define new generator to customize the default behaviour of tree node. Following example customizes ElementIconGenerator and return different icon:

Define NumberNode

```
public static class NumberNode extends Node {
    private int value;
    private Color color;
    public NumberNode() {
        this.value = TWaverUtil.getRandomInt(TreeTest.MAX) + 1;
        this.setName(value + "");
    }
    public int getValue() {
        return this.value;
    }
    public Color getColor() {
        return color;
    }
    public void setColor(Color color) {
        this.color = color;
    }
}
```

Add Element

```
SubNetwork subnetwork = new SubNetwork();
Node node3 = new Node();
node3.setName("Node3");
node3.setParent(subnetwork);
box.addElement(node3);
NumberNode node4=new NumberNode();
Node node5=new Node();
node4.addChild(node5);
box.addElement(node4);
box.addElement(node5);
box.addElement(new NumberNode());
box.addElement(new NumberNode());
box.addElement(new NumberNode());
```

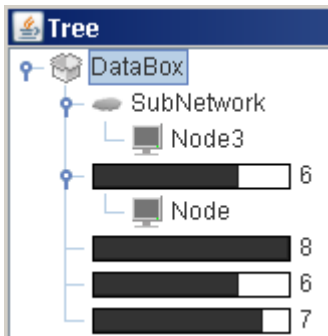
Set IconGenerator

```
tree.setElementIconGenerator(new Generator() {
    public Object generate(Object object) {
        if (!(object instanceof NumberNode)) {
            return new TTreeCellIcon((Element) object, tree);
        }
        final NumberNode node = (NumberNode) object;
        Icon icon = new Icon() {
            public int getIconHeight() {
                return 16;
            }
        }
    }
}
```

```

public int getIconWidth() {
    return 100;
}
//paint tree node icon
public void paintIcon(Component c, Graphics g, int x, int y) {
    //paint selected state
    if (node.isSelected()) {
        g.setColor(UIManager.getColor("Tree.selectionBackground"));
        g.fill3DRect(x, y, getIconWidth(), getIconHeight(), true);
        g.setColor(UIManager.getColor("Tree.selectionBorderColor"));
        g.drawRect(x, y, getIconWidth(), getIconHeight());
    }
    g.setColor(node.getColor());
    g.fill3DRect(x + 1, y + 2, node.getValue() * getIconWidth() / MAX - 2, getIconHeight() - 4, true);
    ((Graphics2D) g).setStroke(TWaverConst.BASIC_STROKE);
    g.setColor(Color.BLACK);
    g.drawRect(x + 1, y + 2, getIconWidth() - 2, getIconHeight() - 4);
}
};
return icon;
}
});

```

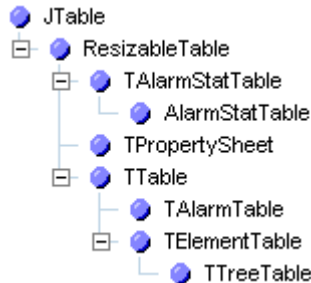


Using ElementIconGenerator

Using Table Component

TWaver table component (twaver.table.TTable) is derived from Swing JTable. TWaver table enhance the JTable with sorting, paging and filtering capabilities, and more very usable features for OSS developers.

TWaver Table class structure



ResizableTable

This is a table can be resized.

TTable

This is a table support sort, paging, data filter. It is the parent class of other TWaver tables (TAlarmTable, TElementTable and TTreeTable).

We strongly suggest you use class TElementTable not TTable unless you know how to use TTable very well, because TElementTable is a DataBox connected table in TWaver, as a DataBox enabled Swing components, TElementTable it is more easy to use and understand as you can control the data from the connected DataBox.

TAlarmTable

A table used to display alarm objects.

AlarmID	AlarmSeverity	Acked	ElementID	ProbableCa
1	Minor		NODE_1	
2	Minor	✓	NODE_2	
5	Major		NODE_3	
6	Major		NODE_3	

TElementTable

The most important and the most often used TWaver Table component. It works with a DataBox and displays each Element in a row. It support XML configuration and more easy to use. TElementTable is the component we suggest you to use in most of cases.

TTreeTable

This is a very special Table component. It is a combination of a TWaver Tree and a TWaver Table. It provides the benefits for tree and table.

	Name	Age
DataBox		
Sam	Sam	89
Tom	Tom	12
Lisa	Lisa	20
Pete	Pete	30

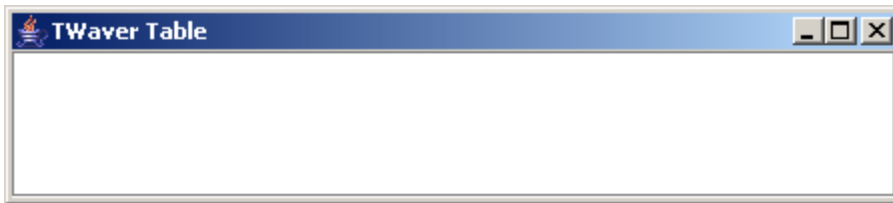
- [Creating Table Component](#)
- [Handling Data with TableModel](#)
- [Hiding Table Columns](#)
- [Locking Table](#)
- [More Tips for Table](#)
- [Resizing Table Column & Row](#)
- [Table Cell Editor](#)
- [Table Cell Renderer](#)
- [Table Column Sorting](#)
- [Table Paging](#)

- [Table Popup Menu](#)
- [Using Internal Columns](#)
- [Using Table Columns](#)
- [Using Table Filter](#)
- [Using Table Listener](#)
- [Using Table Navigator](#)

Creating Table Component

As an extension of Swing `JTable`, it is very simple to create a table component instance. Following snippet creates an empty table component and displays it in a `JFrame`:

```
JFrame frame = new JFrame("TWaver Table");
TTable table = new TTable();
frame.getContentPane().add(new JScrollPane(table), BorderLayout.CENTER);
frame.setSize(400, 100);
TWaverUtil.centerWindow(frame);
frame.show();
```



Constructors of TWaver table component:

Method	Description
<code>TTable()</code>	Default constructor used to create an empty table.
<code>TTable(String tableName)</code>	Create a table with a configuration name which defined in <code>TWaver.XML</code> .

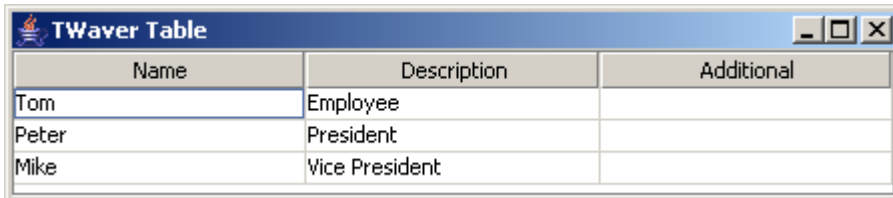
Handling Data with TableModel

TWaver extends class TTableModel from Swing TableModel. Normally you need to retrieve the table model to handle table data.

```
TTable table = new TTable();
//retrieve the internal table model instance.
TTableModel model = table.getTableModel();
```

Use following code to add/insert rows:

```
//add first row
Vector row = new Vector();
row.addElement("Peter");
row.addElement("President");
model.addRow(row);
//add second row
row = new Vector();
row.addElement("Mike");
row.addElement("Vice President");
model.addRow(row);
//insert row
row = new Vector();
row.addElement("Tom");
row.addElement("Employee");
model.insertRow(0,row);
```



Name	Description	Additional
Tom	Employee	
Peter	President	
Mike	Vice President	

Use TTableModel.removeRow(int index) to remove row from table model.
 Use TTableModel.clearPublishedData () to clear all published data.
 Use TTableModel.clearRawData () to clear all row data.
 Use TTableModel.getPublishedData () to get all published data, include invisible columns
 Use TTableModel.getCurrentPageData () get current page published data, include invisible columns

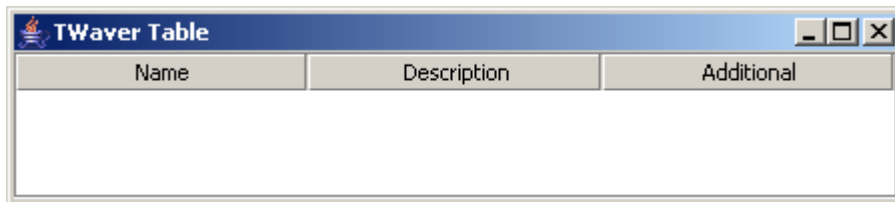


Please see section [Table Paging](#) for more details about published data and raw data.

Hiding Table Columns

You can change the visibility to show or hide table columns.

```
TTable table = new TTable();
table.addColumn(new TTableColumn("Name"));
table.addColumn(new TTableColumn("Description"));
table.addColumn(new TTableColumn("Additional"));
```

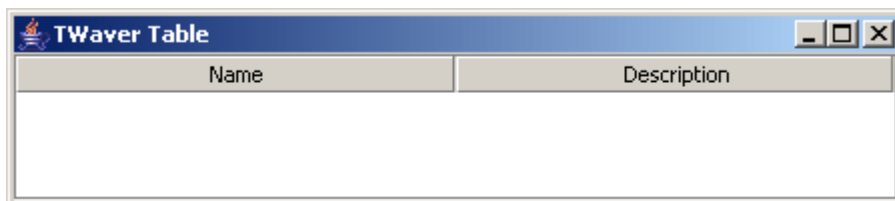


Hide "Additional" column:

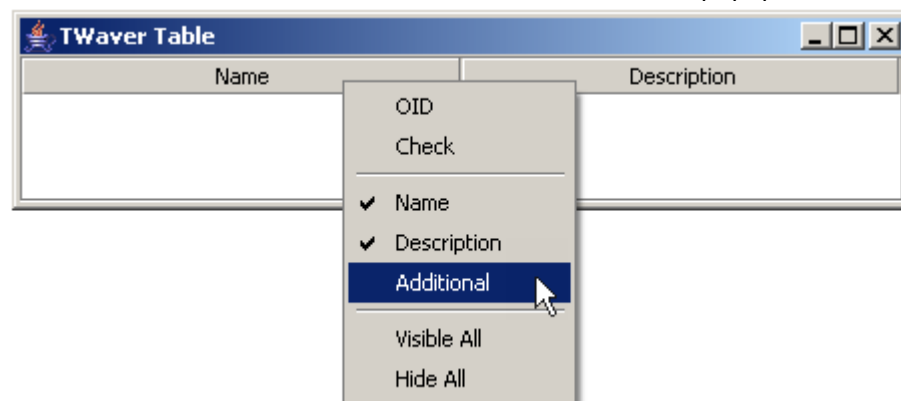
```
table.getColumnByName("Additional").setVisible(false);
```

Also you can use column visible filter:

```
table.setColumnVisibleFilter(new ColumnVisibleFilter(){
    public boolean isVisible(TTableColumn column) {
        return !"Additional".equals(column.getName());
    }
});
```



You can also show or hide columns from the column header popup menu:

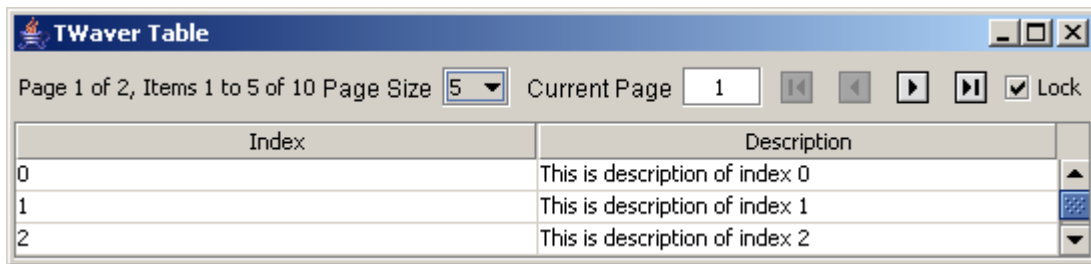


TWaver Table		
Name	Description	Additional

Locking Table

When large numbers of data insert into table in a short time, the table will be updated and scrolled rapidly. That means you maybe have no enough time to see check the details. For example, an OSS application may raise thousands alarms and insert alarms into the table in short order. In this case, you need to "freeze" the table to check each alarm to know what happened.

Fortunately, TWaver table provides the capability to lock table. You can lock table by clicking the "lock" check box on the right side of table navigator panel.



Or lock/unload table by APIs:

```
TTableModel model = table.getTableModel();
//lock table
model.lock();
//unlock table
model.unlock();
```

During the locked time, the published data will be kept, even new rows are inserted. If unlocked, the raw data will be republished instantly.

More Tips for Table

Select Whole Row on Rigt-Click

Use following method to enable right-click-select-row function:

`TTable.setSelectableOnRightClick(true)`

- Using Alternating Row Color

```
TElementTable table = new TElementTable(box){
    public Component prepareRenderer(TableCellRenderer renderer, int row,int column) {
        Component result=super.prepareRenderer(renderer, row, column);
        if(row%2==0){
            result.setBackground(Color.white);
        }else{
            result.setBackground(new Color(238, 238, 238));
        }
        return result;
    }
};
```

- Hiding Grid Line

```
//Hiding vertical grid line
table.setShowVerticalLines(false);
//Hiding horizontal grid line
table.setShowHorizontalLines(false);
```

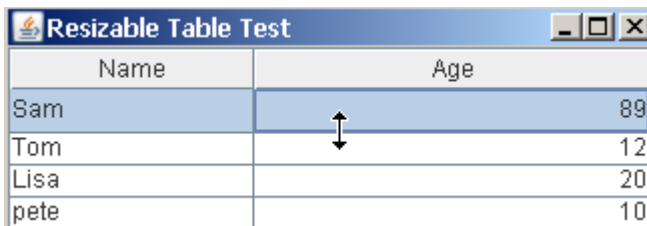
Resizing Table Column & Row

TWaver Table has the ability to resize each column and row size. See JavaDoc of class `twaver.table.ResizableTable` for more details.

Adjust Row Height

By default row height is not adjustable. You can call method `setRowResizable` to change:

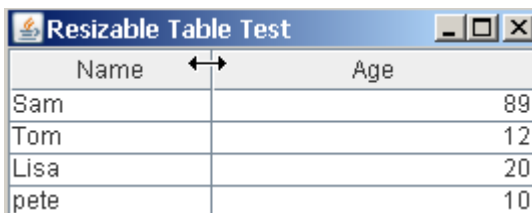
```
table.setRowResizable(true);
```



Name	Age
Sam	89
Tom	12
Lisa	20
pete	10

Adjust Column Width

You can drag/drop the gap of table header to adjust the table column width. This is enabled by default.

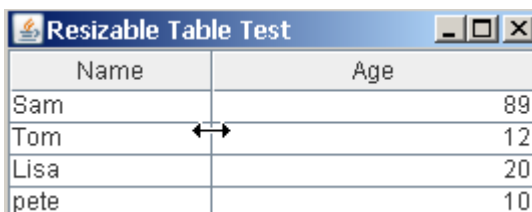


Name	Age
Sam	89
Tom	12
Lisa	20
pete	10

Adjusting Table Column

You can also drag/drop the gap in data area to adjust column width, if you use following method enable it (by default it disabled):

```
table.setColumnResizable (true);
```



Name	Age
Sam	89
Tom	12
Lisa	20
pete	10

Now you can drag/drop column gap in the data area to adjust column width. Please note by default this is disabled.

Column Pack

Pack a column means adjust the column width to a preferred size by considering all cell width and this column header's width. Follow will pack all table columns:

```
table.packAllColumns(true); // "true" means consider this column header's width
```

AlarmID	AlarmSeverity			P...	Owner
ALM_10	Indeterminate			S...	Peter
ALM_11	Major	✓		R...	Mary
ALM_12	Critical			T	Peter

Before Column Pack

AlarmID	AlarmSeverity	Acked	ElementID	ProbableCause	Owner
ALM_10	Indeterminate		Router-1	Software program abnormality terminated	Peter
ALM_11	Major	✓	Router-2	Receive failure	Mary
ALM_12	Critical		Router-1	Tranceiver problem	Peter

After Column Pack

To pack a specified column:

```
table.packColumn(table.getTableModel().getColumnByName("age"), true);
```


Table Cell Editor

Just like Swing JTable, each table column can has a cell editor to edit the value of each cells of this column. TWaver provides a lot of predefined table cell editors in package `twaver.table.editor` for your use. Please check JDK document for more details about table and editor.

```
TTable table = new TTable();
table.addColumn(new TTableColumn("Index",
    "Index",
    new twaver.table.renderer.NumberRenderer()));
table.addColumn(new TTableColumn("color",
    "Color",
    new twaver.table.renderer.ColorRenderer()));
table.addColumn(new TTableColumn("date",
    "Date",
    new twaver.table.renderer.DateRenderer()));
TTableModel model = table.getTableModel();
//set color column editable.
model.getPublishColumn(1).setWrite(true);

for (int i = 0; i < 10; i++) {
    Vector row = new Vector();
    row.addElement(new Integer(i));
    row.addElement(Color.red);
    row.addElement(new Date());
    model.addRow(row);
}
```

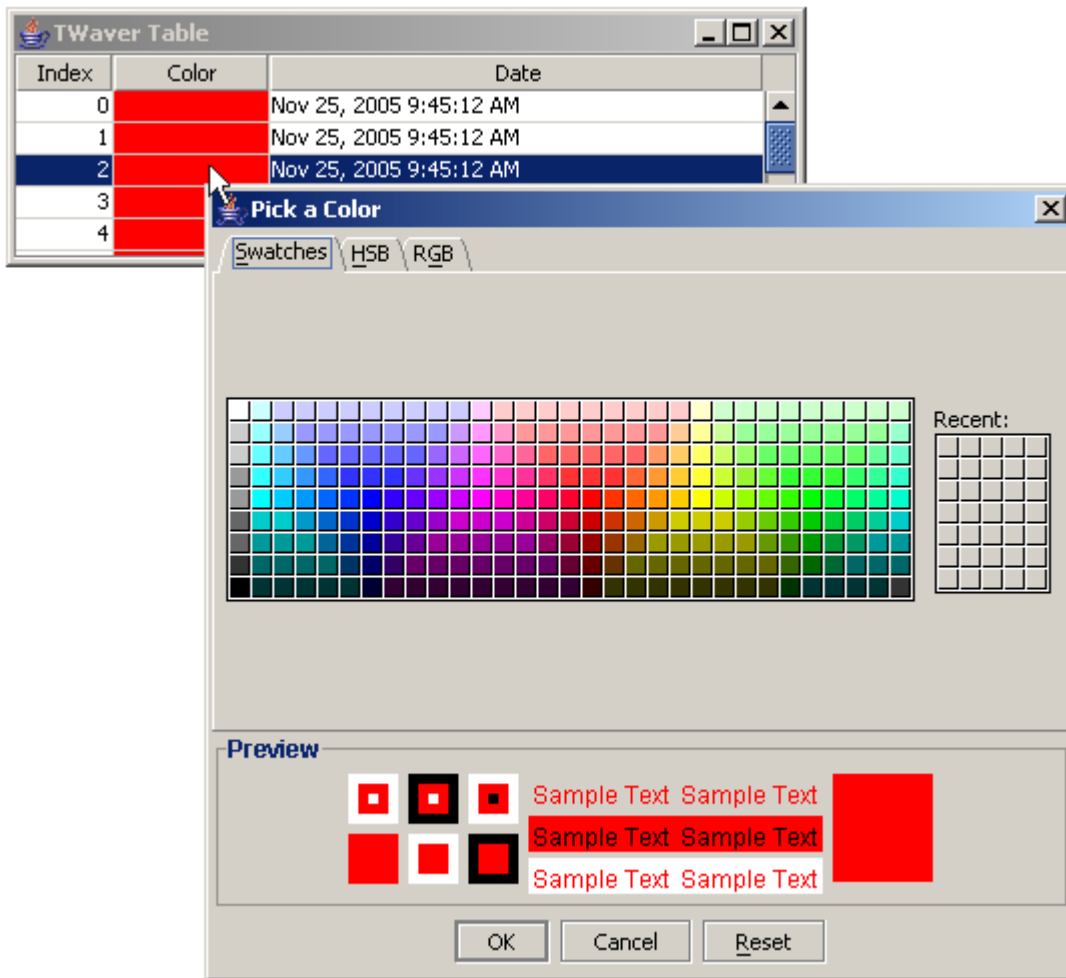
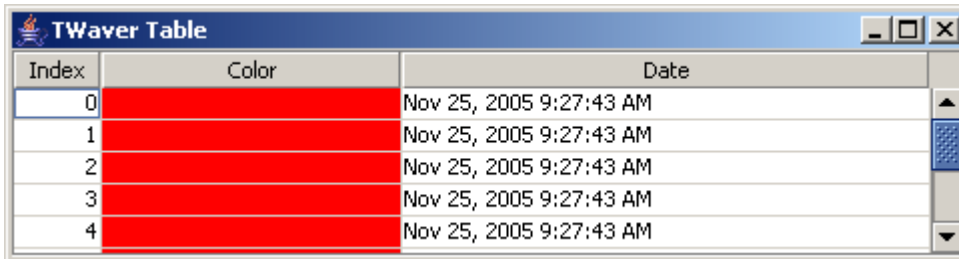


Table Cell Renderer

Just like Swing `JTable`, each table column can have a cell renderer to paint each cell of this column. TWaver provides a lot of predefined table cell renderers in package `twaver.table.renderer` for use. Please check JDK document for more details about table and renderer.

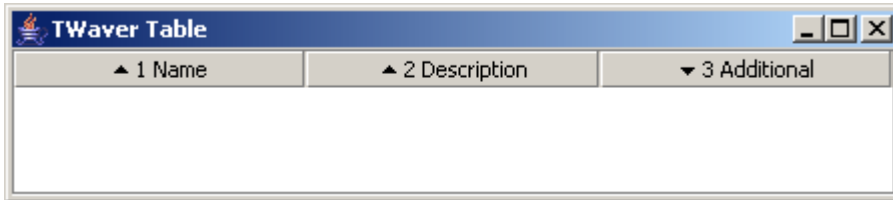
```
TTable table = new TTable();
table.addColumn(new TTableColumn("Index",
    "Index",
    new twaver.table.renderer.NumberRenderer()));
table.addColumn(new TTableColumn("color",
    "Color",
    new twaver.table.renderer.ColorRenderer()));
table.addColumn(new TTableColumn("date",
    "Date",
    new twaver.table.renderer.DateRenderer()));
TTableModel model = table.getTableModel();
model.getPublishedColumn(1).setEditable(true);
for (int i = 0; i < 10; i++) {
    Vector row = new Vector();
    row.addElement(new Integer(i));
    row.addElement(Color.red);
    row.addElement(new Date());
    model.addRow(row);
}
```



Index	Color	Date
0		Nov 25, 2005 9:27:43 AM
1		Nov 25, 2005 9:27:43 AM
2		Nov 25, 2005 9:27:43 AM
3		Nov 25, 2005 9:27:43 AM
4		Nov 25, 2005 9:27:43 AM

Table Column Sorting

TWaver table data can be sorted by column. The sort mode can be ascend, descend or none. Click table column header to change the sorting mode. Press Ctrl key and click table column header to sort table with multiple columns.

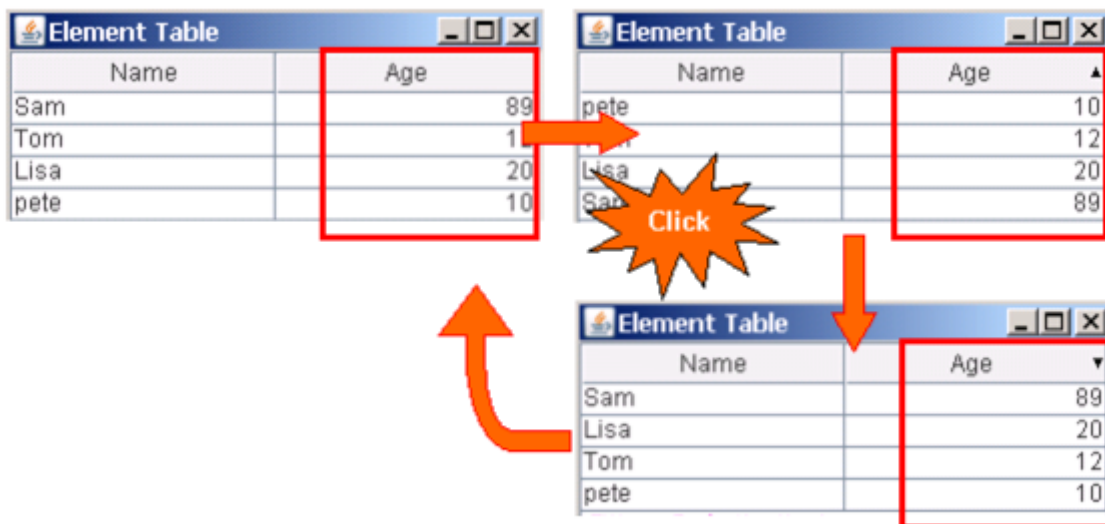


Use `table.setSortable(false)` to disable table sorting.

Use `TTableColumn.setSortComparator(java.util.Comparator)` to customize the sorting.

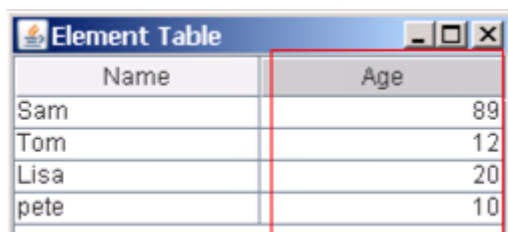
Using Sort

TWaver Table support column sort and enabled by default. Click Table column header to sort column:



You can set a column not sortable:

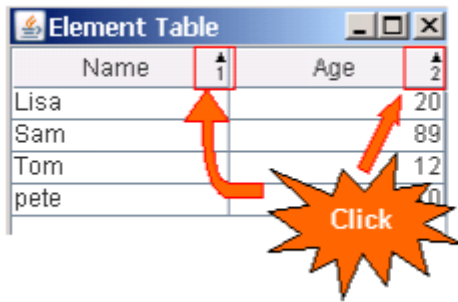
```
table.getTableModel().getColumnByName("age").setSortable(false);
```



You can see above picture, the sort disabled column header is in a more darker color.

Multi-Column Sort

TElementTable support multi-column sort. You can press key "ctrl" and click column header to see multi-column sort:



Name	Age
Lisa	20
Sam	89
Tom	12
pete	0

Press "ctrl" key and click

Sort by APIs

Single column sort:

```
table.getTableModel().sortColumn(0, false);
```

Multi-column sort:

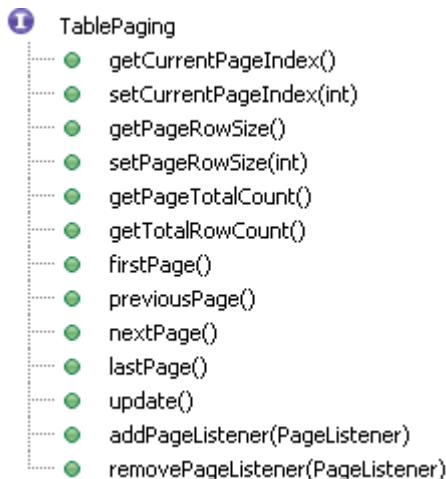
```
table.getTableModel().sortColumn(0, false); //here also can be "true"
table.getTableModel().sortColumn(1, true); //use "true" equals to pressing "ctrl" key
```

Table Paging

TWaver table model has the ability for pagination. When paging enabled, all original data of table model will be saved in a collection which called raw data. Similarly, table model will figure out and save the data of current page into another collection which called published data. Only published data will be displayed on TWaver table component. This process called "data publishing".

You can visit raw data and published data APIs of table model.

The class `twaver.table.TTableModel` implements the interface `twaver.table.TablePaging`. You can get the paging information with `twaver.table.TablePaging`. interface. The paging information includes page size, current page index, total page count, etc.



Following snippet shows how to control the pagination.

```

TTable table = new TTable();
table.addColumn(new TTableColumn("Index"));
table.addColumn(new TTableColumn("Description"));
TTableModel model = table.getTableModel();
//insert 10 rows to the table.
for(int i=0;i<10;i++){
    Vector row = new Vector();
    row.addElement(""+i);
    row.addElement("Description for index "+i);
    model.addRow(row);
}
//set page size to 3.
model.setPageRowSize(3);
//goto the second page.
model.setCurrentPageIndex (2);
    
```

TWaver Table	
Index	Description
3	Description for index 3
4	Description for index 4
5	Description for index 5

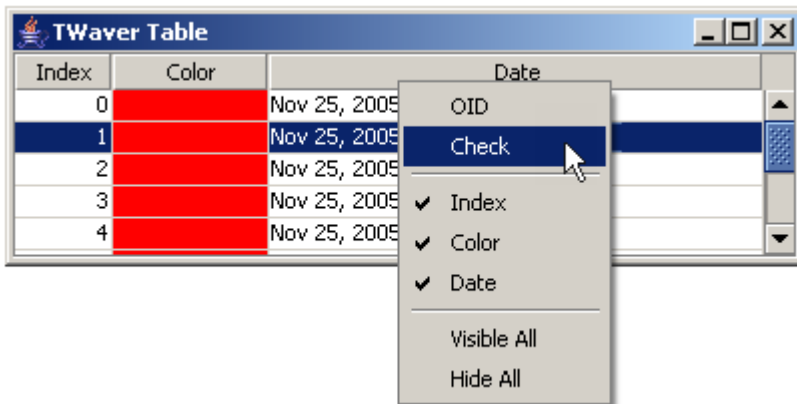


For method `TTableModel.setCurrentPageIndex(int pageIndex)`, please note that the first page would be 1, not 0.
More detailed information about table paging, please see java doc of class `TTableModel`.

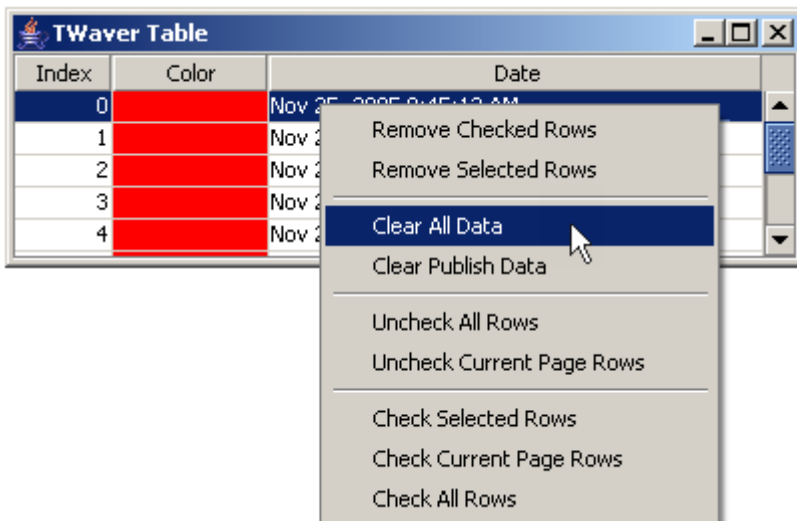
You can navigate the table pages by table navigator panel. See section [Using Table Navigator](#) for the details.

Table Popup Menu

There are two kinds of popup menus on TWaver table: body popup menu and column header popup menu. The default popup menu looks as follows:



The default popup menu on column header



The default popup menu on table body

To customize the popup menu, you need to create a `TTablePopupMenuFactory` and set it to the table.

```

TTable table = new TTable();
//disable popup menu on column header.
table.setTableHeaderPopupMenuFactory (null);
//disable popup menu on table body.
table.setTableBodyPopupMenuFactory(null);
//change popup menu on column header.
table.setTableHeaderPopupMenuFactory (new TTablePopupMenuFactory() {
public JPopupMenu getPopupMenu(TTable table, MouseEvent e) {
    JPopupMenu menu = new JPopupMenu();
    menu.add(new JMenuItem("menu one"));
    menu.add(new JMenuItem("menu two"));
    return menu;
}
});
//change popup menu on table body.
table.setTableBodyPopupMenuFactory(new TTablePopupMenuFactory() {
public JPopupMenu getPopupMenu(TTable table, MouseEvent e) {
    JPopupMenu menu = new JPopupMenu();

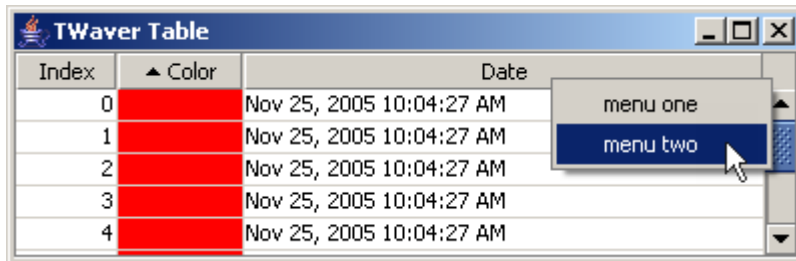
```



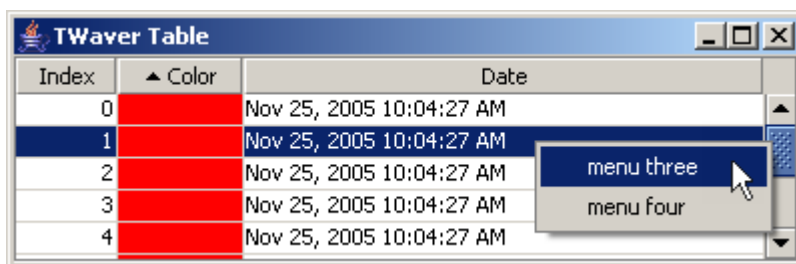
```

        menu.add(new JMenuItem("menu three"));
        menu.add(new JMenuItem("menu four"));
        return menu;
    }
});

```



Customized popup menu on column header



Customized popup menu on table body

Using Internal Columns

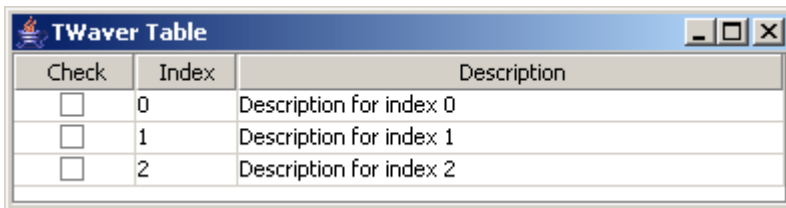
TWaver table has two invisible columns for internal use: OID column and Check column.

- [Check Column](#)
- [OID Column](#)

Check Column

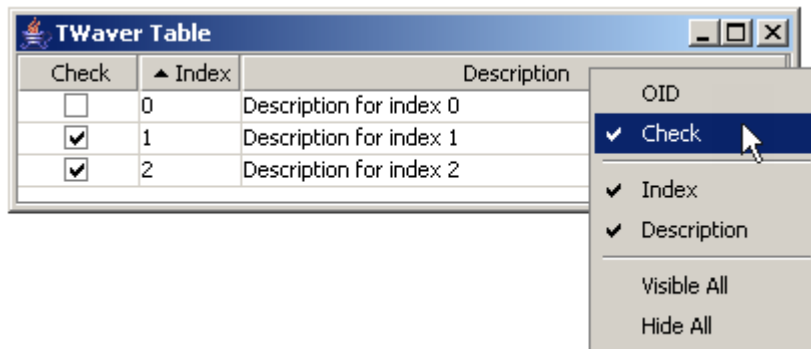
Check status is an automatically generated value for each table row. In fact, they are instances of `java.lang.Boolean`. Check status present whether or not the current row is checked by user. By default check column is invisible, and you can make it visible as follows:

```
TTable table = new TTable();
//set check status column visible.
table.getCheckColumn.setVisible(true);
```



Check	Index	Description
<input type="checkbox"/>	0	Description for index 0
<input type="checkbox"/>	1	Description for index 1
<input type="checkbox"/>	2	Description for index 2

You can click column header popup menu to show or hide check column:



Check	▲ Index	Description
<input type="checkbox"/>	0	Description for index 0
<input checked="" type="checkbox"/>	1	Description for index 1
<input checked="" type="checkbox"/>	2	Description for index 2

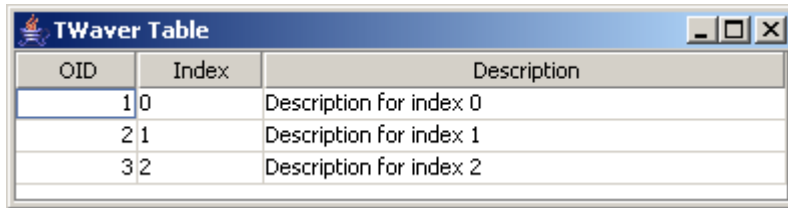
OID
☒ Check
☒ Index
☒ Description
Visible All
Hide All

You can get the list of elements whose check column is selected in `TElementTable` by the method `"getAllCheckedElements"`

OID Column

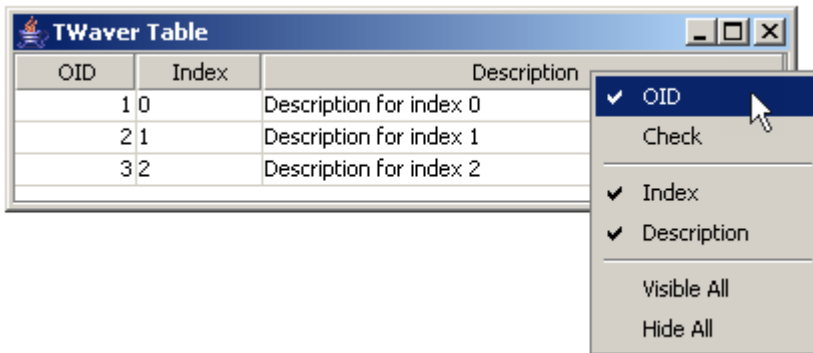
OID is an automatically generated unique identifier for each table row. In fact, they are instances of `java.lang.Integer`. OID is designed for internal use and invisible by default. You can make it visible as follows:

```
TTable table = new TTable();
//set OID column visible.
table.getOIDColumn.setVisible(true);
```



OID	Index	Description
1 0		Description for index 0
2 1		Description for index 1
3 2		Description for index 2

You can click column header popup menu to show or hide OID column:



OID	Index	Description
1 0		Description for index 0
2 1		Description for index 1
3 2		Description for index 2

- ✓ OID
- Check
- ✓ Index
- ✓ Description
- Visible All
- Hide All

Using Table Columns

Normally an empty table can do nothing. So let's add some columns into the table now. TWaver provides class `twaver.table.TTableColumn` to encapsulate the table column informations. It is extended from `javax.swing.table.TableColumn` and gives you more options for sorting, visible and others.

Adds a column:

```
twaver.TTable
public void addColumn(TableColumn aColumn)
public void addColumns(List columnList)
twaver.table.TTableModel
public void addColumn(TableColumn aColumn)
public void addColumns(List columnList)
```

Deletes a column:

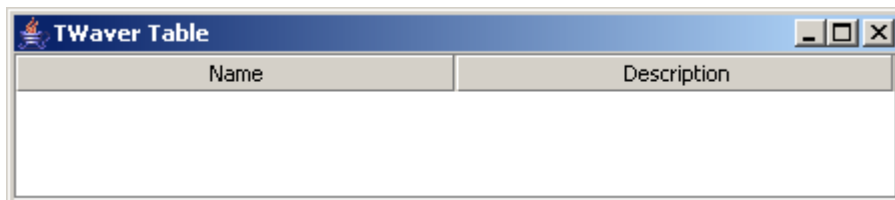
```
twaver.TTable
public void removeColumn(TableColumn column)
public void removeColumnByName(String columnName)
twaver.table.TTableModel
public void removeColumn(TableColumn column)
public void removeColumnByName(String columnName)
```

Clears all columns:

```
twaver.table.TTableModel
public void clearAllColumns()
```

Here we create two columns for table:

```
TTable table = new TTable();
table.addColumn(new TTableColumn("Name"));
table.addColumn(new TTableColumn("Description"));
```



Following properties are defined in TWaver table column:

Property	Type	Description
name	String	The column name. It will be used as the key of I18N
display name	String	The column display name. If display name is not null, then it will be displayed as the column header text, otherwise, TWaver will get the internationalized string from i18n resource files. The key will be "table.column."+name.
sortComparator	java.util.Comparator	Comparator for column sorting.

editable	boolean	Whether or not the column is editable.
visible	boolean	Whether or not the column is visible.
sort mode	int	<ul style="list-style-type: none"> • TTableColumn.SORT_NON: no sorting • TTableColumn.SORT_ASCEND: ascend sortinig • TTableColumn.SORT_DESCEND: descend sorting
preferredWidth	int	Preferred column width.
javaClass	Class	The data class type of this column. TWaver will use it to get the registered renderer and editor from TUIManager.
editor	TableCellEditor	Editor for column.
renderer	TableCellRenderer	Renderer for column.

See javadoc of class `twaver.table.TTableColumn` for details.

Using Table Filter

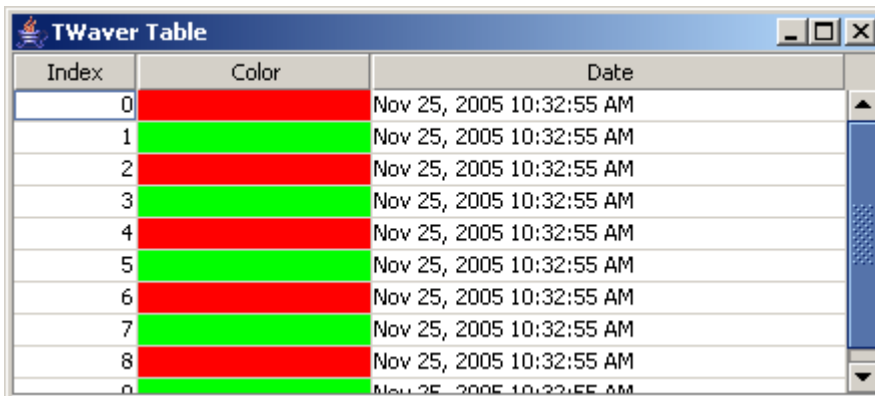
TWaver table has the capability to filter data. You can set arbitrary number filters for one table.

Using TTableRowFilter

Following snippet demonstrate how to use table data filter.

```
TTable table = new TTable();
table.addColumn(new TTableColumn("Index"));
table.addColumn(new TTableColumn("Color"));
table.addColumn(new TTableColumn("Date"));
TTableModel model = table.getTableModel();

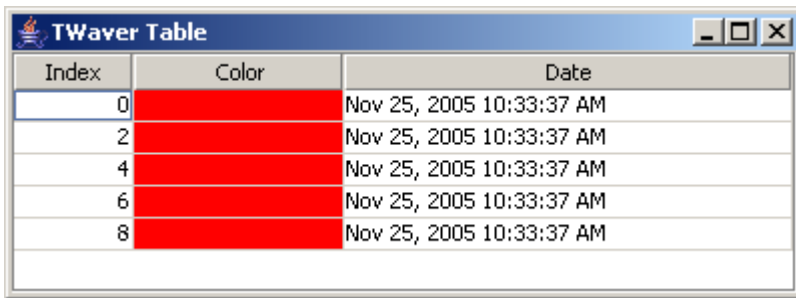
for (int i = 0; i < 10; i++) {
    Vector row = new Vector();
    row.addElement(new Integer(i));
    if(i%2==0){
        row.addElement(Color.red);
    }else{
        row.addElement(Color.green);
    }
    row.addElement(new Date());
    model.addRow(row);
}
```



Index	Color	Date
0	Red	Nov 25, 2005 10:32:55 AM
1	Green	Nov 25, 2005 10:32:55 AM
2	Red	Nov 25, 2005 10:32:55 AM
3	Green	Nov 25, 2005 10:32:55 AM
4	Red	Nov 25, 2005 10:32:55 AM
5	Green	Nov 25, 2005 10:32:55 AM
6	Red	Nov 25, 2005 10:32:55 AM
7	Green	Nov 25, 2005 10:32:55 AM
8	Red	Nov 25, 2005 10:32:55 AM
9	Green	Nov 25, 2005 10:32:55 AM

Then we create a data filter, which will filter out the non-red color rows:

```
//add a table row filter, filter out rows with non-red color.
model.addRowFilter(new TTableRowFilter(){
    public boolean isVisible(TTable table, Vector rowData) {
        //Note: TTable has two internal predefined columns,
        //so the column index is start from 2.
        Color color=(Color) rowData.get(3);
        return color.equals(Color.red);
    }
});
```



Index	Color	Date
0		Nov 25, 2005 10:33:37 AM
2		Nov 25, 2005 10:33:37 AM
4		Nov 25, 2005 10:33:37 AM
6		Nov 25, 2005 10:33:37 AM
8		Nov 25, 2005 10:33:37 AM

With this capability, you can make more complex table data filters for your application.

Using Table Listener

TableModel can add table listeners to monitor these cases:

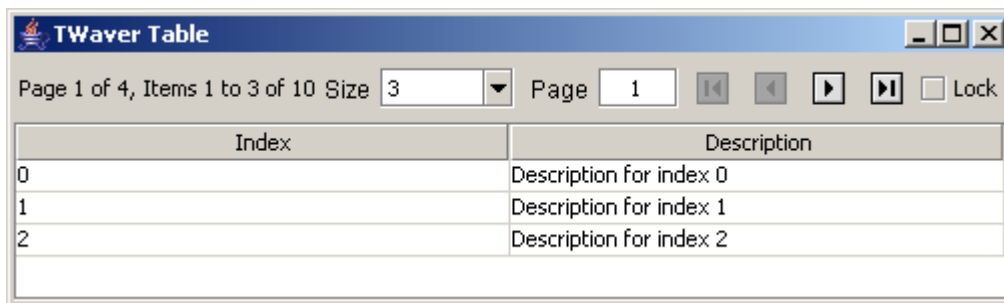
- rowClicked
- rowSelectionChanged
- beforeCellValueChanged
- tableUpdated
- lockedChanged
- tableDataChanged

See class `twaver.table.TTableListener` for more information.

Using Table Navigator

Use APIs to control the table paging is not always convenient for all cases. TWaver provides a predefined navigate panel for table component. It extends Swing JPanel, and works with table. Use following code to create a table navigate panel.

```
//create table with paging options.
TTable table = new TTable();
...
frame.getContentPane().add(new JScrollPane(table), BorderLayout.CENTER);
frame.getContentPane().add(new TTableNavigator(table.getTableMode()),
BorderLayout.NORTH);
```



Click "first", "next", "previous" and "last" buttons to navigate pages. You can also change the page size and current page index on navigator panel.

The page size options can be changed as follows:

```
TTableModel model=alarmTable.getTableModel();
int[] options=new int[] {20, 30, 50, 0};
TTableNavigator navigator = new TTableNavigator(model, options);
```

Using Element Table Component

Element table component (`twaver.table.TElementTable`) is an extension of TWaver table. Just like alarm table, element table is used to display TWaver element objects. Element table connects to a DataBox and gets all elements from the DataBox, and lists them on the table with predefined columns.

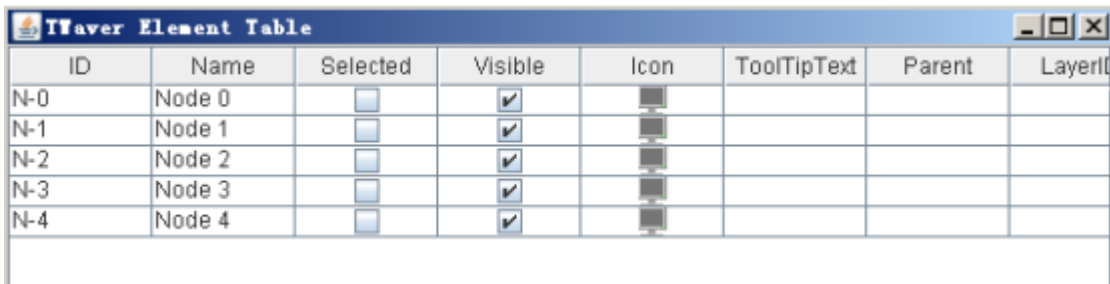
Element table derived from TWaver table, so it inherit all features of the basic table like customized columns, sorting, paging control, locking, popup menu and filters. Please see more details in section [Using Table Component](#).






- [Creating Element Table Component](#)
- [Customizing Element Table Component](#)
- [Getting Elements from Table](#)
- [Loading Data](#)
- [Using SendToTop Filter and SendToBottom Filter](#)
- [Using Visible Filter](#)

Creating Element Table Component

Normally we need a DataBox to create an element table. In this case the table will connect to a DataBox, display and monitor the data of the DataBox. Following snippet creates an element table component and displays all elements of the DataBox:

```
JFrame frame = new JFrame("TWaver Element Table");
TDataBox box = new TDataBox();
TElementTable table = new TElementTable(box);
table.setElementClass(Element.class);
frame.setContentPane(new JScrollPane(table));
frame.setSize(400, 100);
TWaverUtil.centerWindow(frame);
//create several elements.
for (int i = 0; i < 5; i++) {
    Node node = new Node("N-" + i);
    node.setName("Node " + i);
    node.getAlarmState().addNewAlarm(AlarmSeverity.getNonClearedRandomSeverity());
    box.addElement(node);
}
frame.show();
```



ID	Name	Selected	Visible	Icon	ToolTipText	Parent	LayerID
N-0	Node 0	<input type="checkbox"/>	<input checked="" type="checkbox"/>				
N-1	Node 1	<input type="checkbox"/>	<input checked="" type="checkbox"/>				
N-2	Node 2	<input type="checkbox"/>	<input checked="" type="checkbox"/>				
N-3	Node 3	<input type="checkbox"/>	<input checked="" type="checkbox"/>				
N-4	Node 4	<input type="checkbox"/>	<input checked="" type="checkbox"/>				

Customizing Element Table Component

Define Table via [Define BeanInfo by XML](#)

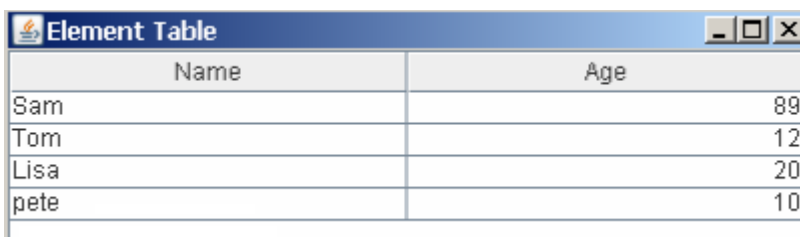
```
TDataBox box = new TDataBox();
TElementTable table = new TElementTable(box);
//register BeanInfo by XML
table.registerElementClassXML(Person.class, "table.xml");
//set element class type
table.setElementClass(Person.class);
box.addElement(new Person("Sam", 89));
box.addElement(new Person("Tom", 12));
box.addElement(new Person("Lisa", 20));
box.addElement(new Person("pete", 10));
```

Person.java

```
public static class Person extends Node {
    public Person(String name, int age) {
        this.setName(name);
        this.putClientProperty("age", age);
    }
    public int getAge(){
        String value= this.getClientProperty("age").toString();
        return Integer.valueOf(value).intValue();
    }
}
```

Config file: table.xml

```
<beaninfo>
<attribute
  name="name"
  displayName="Name"/>
<attribute
  clientPropertyKey="age"
  displayName="Age"/>
</beaninfo>
```



Name	Age
Sam	89
Tom	12
Lisa	20
pete	10

See [Using Alarm Table Component](#) and [Using Table Component](#).

Getting Elements from Table

In TElementTable, each row connects to an Element object. Cells in Table normally display a value of an Element. So we can get Element object from row information and vice versa.

- Get Element by Row Index

```
TElementTable.getElementByRowIndex(int row)
```

- Get Element by Row Data

```
TElementTable.getElementByRowData(Vector rowData)
```

- Get Published Element

```
public List getPublishedElements()
```

- Get Elements for Current Page

```
public List getCurrentPageElements()
```

- Get Elements for Selected Rows

```
table.getDataBox().getSelectionModel().getAllSelectedElement();
```

An Example

```
TDataBox box=new TDataBox();
final TElementTable table = new TElementTable(box);
table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
table.setElementClass(Person.class);

List attributes = new ArrayList();
ElementAttribute nameAttribute = new ElementAttribute();
nameAttribute.setName("name");
nameAttribute.setDisplayName("Name");
nameAttribute.setEditable(false);
attributes.add(nameAttribute);
ElementAttribute ageAttribute = new ElementAttribute();
ageAttribute.setEditable(true);
ageAttribute.setJavaClass(Integer.class);
ageAttribute.setClientPropertyKey("age");
ageAttribute.setDisplayName("Age");
ageAttribute.setSortable(false);
ageAttribute.setDescription("this is the description.");
attributes.add(ageAttribute);
ElementAttribute descriptionAttribute = new ElementAttribute();
descriptionAttribute.setClientPropertyKey("description");
```

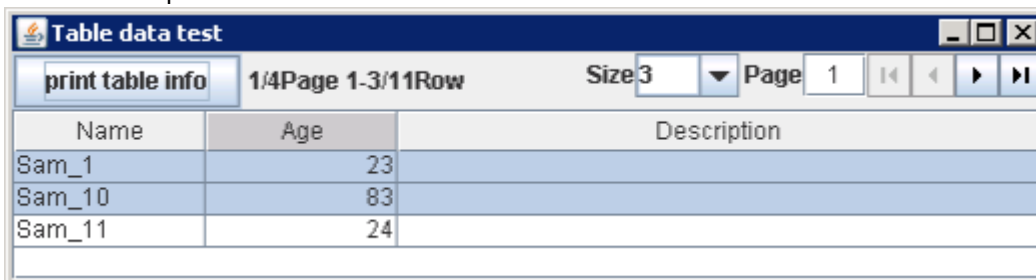
```

descriptionAttribute.setDisplayName("Description");
descriptionAttribute.setWidth(300);
descriptionAttribute.setFontStyle(TWaverConst.FONT_STYLE_BOLD);
descriptionAttribute.setEditable(false);
attributes.add(descriptionAttribute);
table.registerElementClassAttributes(Person.class, attributes);

for(int i=0;i<100;i++){
    box.addElement(new Person("Sam_"+i, TWaverUtil.getRandomInt(100)));
}
VisibleFilter visibleFilter=new VisibleFilter(){
    public boolean isVisible(Element element) {
        if(element.getName().startsWith("Sam_1")){
            return true;
        }
        return false;
    }
};
table.addVisibleFilter(visibleFilter);
table.packAllColumns(false);
table.getTableModel().setPageRowSize(3);
JPanel pane=new JPanel(new BorderLayout());
TTableNavigator navigator = new TTableNavigator(table.getTableModel(), new int[] {20, 30, 50, 0}, false);
JButton btn=new JButton("print table info");
btn.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) {
        //Get published elements
        System.out.println("all published elements size is: "+table.getPublishedElements().size());
        //get elements on current page
        System.out.println("all current page elements size is: "+table.getCurrentPageElements().size());
        //get columns displaying
        System.out.println("all visible columns size is: "+table.getTableModel().getPublishedColumn().size());
        //Get selected rows
        System.out.println("all selected rows size is: "+table.getTableModel().getSelectedRows().size());
    }
});
JPanel northPane=new JPanel(new BorderLayout());
northPane.add(btn,BorderLayout.WEST);
northPane.add(navigator,BorderLayout.CENTER);
pane.add(northPane,BorderLayout.NORTH);
pane.add(new JScrollPane(table),BorderLayout.CENTER);
JFrame frame = new JFrame();
frame.setTitle("Table data test");
frame.getContentPane().add(pane,BorderLayout.CENTER);
frame.setSize(900, 700);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
TWaverUtil.centerWindow(frame);
frame.setVisible(true);

```

Run this example:

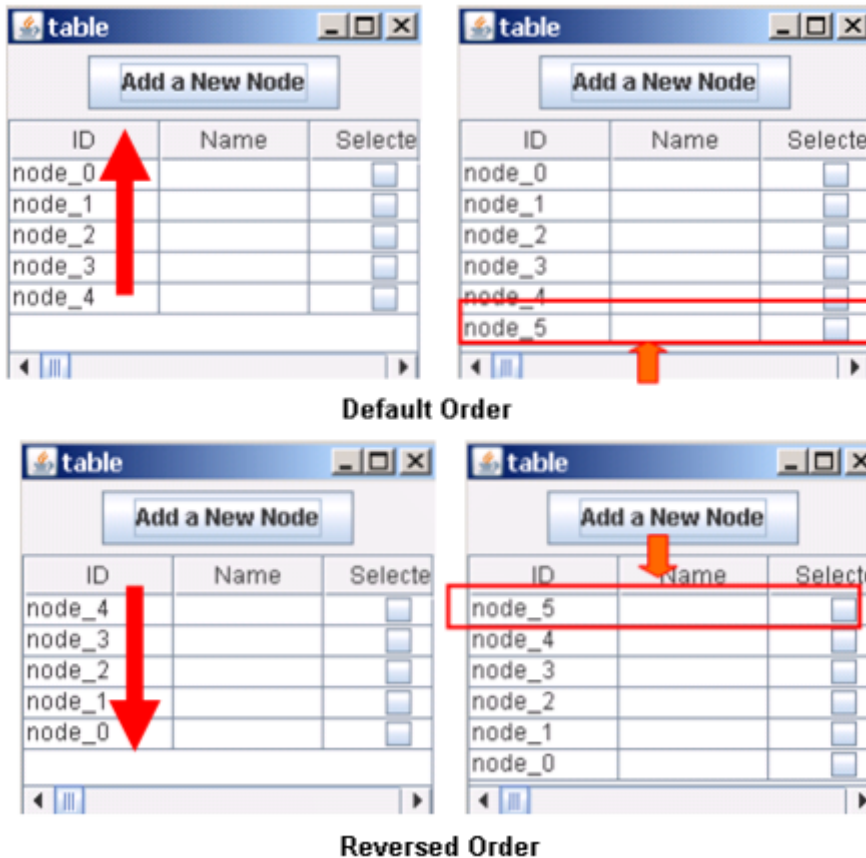


Result:

all published elements size is: 11
all current page elements size is: 3
all visible columns size is: 3
all selected rows size is: 2

Loading Data

Table has the same Element order with its connected DataBox. By default, the later added Element listed in the table bottom. But you may want see the later added Element show on the top. For example, the new occurred Alarm normally is wanted to be listed on the top, the method 'table.setConverseIncreaseOrder(true)' can be invoked to do this.



The initial sequence of the elements in table is the same with databox's twaver.TDataBox

```
public void sendToTop(final Element element)
public void sendToBottom(Element element)
public void sendToBottom(Element element, Element referenceElement)
```


 After the initialization of table ,the order of form data will be no change when databox order changed

Table has another order model - Hierarchy Order, can be set through the following methods

```
TElementTable
public void setIteratorByHierarchy(boolean iteratorByHierarchy)
```

The hierarchy order of table and the depth traversal of databox are using the same way, can change the order of form data through the following methods

```
twaver.TDataBox
public void addElement(int index, final Element element)
public void moveTo(int index, Element element)
public void moveToUp(Element element)
public void moveToDown(Element element)
public void moveToTop(Element element)
```



By Hiberarchy Index model, `table.setConverseIncreaseOrder(true)` will be no longer useful

Using SendToTop Filter and SendToBottom Filter

You can use `sendToTopFilter` or `sendToBottomFilter` to set the row always in the top of the table or int the bottom of the table.

```
public void setSendToTopFilter(SendToTopFilter sendToTopFilter)
public void setSendToBottomFilter(SendToBottomFilter sendToBottomFilter)
```

ID	Legend Name ▲	Legend Icon	ID	Legend Name ▼	Legend Icon
[0]	TerminalPoint		[0]	TerminalPoint	
[1]	Text		[1]	Text	
[2]	Card		[2]	ShapeSubNetwork	
[3]	FlexionLink		[3]	ShapeLink	
[4]	Node		[4]	PolySubNetwork	
[5]	OrthogonalLink		[5]	OrthogonalLink	
[6]	PolySubNetwork		[6]	Node	
[7]	ShapeLink		[7]	FlexionLink	
[8]	ShapeSubNetwork		[8]	Card	
[9]	BaseElement		[9]	BaseElement	

Using Visible Filter

You can use VisibleFilter to control the row visibility for TElementTable.

List getVisibleFilters()

void addVisibleFilter(VisibleFilter visibleFilter)

void removeVisibleFilter(VisibleFilter visibleFilter)

```
table.addVisibleFilter(new VisibleFilter(){
    public boolean isVisible(Element element) {
        Person person = (Person)element;
        return person.getAge() >= 18;
    }
});
```

Visible filter table	
Name	Age
Sam	89
Lisa	20

Using TreeTable Component

A TreeTable is a combination of a Tree and a Table - a component capable of both expanding and contracting rows, as well as showing multiple columns of data. TWaver contain a TTreeTable component, and very easy to use.

- [Creating Tree Table Component](#)
- [Using Tree Table Component](#)

Creating Tree Table Component

You can use TreeTable like a Table component, and get the Tree component to customize it. Following snippet creates a simple TreeTable.

TreeTable and TElementTable use the same config file:

```
<beaninfo>
<attribute
name="name"
displayName="Name"
editable="true"/>









<attribute
name="demoClass"
displayName="DemoClass"
editable="false"
width="200"/>
</beaninfo>
```

```
TWaverUtil.registerBeanInfoWithoutDefault(DemoNode.class);
...
//create tree table.
TDataBox box= new TDataBox();
TTreeTable table=new TTreeTable(box);
TTree tree=table.getTree();
//set attribute to the tree table.
table.setRowHeight(18);
table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
table.setElementClass(DemoNode.class);
//Set tree column display name
table.setTreeColumnDisplayName("Demos");
//set DataBox icon
tree.setDataBoxIconURL ("/demo/resource/share/box.png");
//add some data into the DataBox.
...
```

Demos	Name	DemoClass
All Demos		
DataBox		
Monitor Property Demo	Monitor Property Demo	class demo.databox.monitorprop...
Monitor Element Demo	Monitor Element Demo	class demo.databox.monitorele...
XML Driven Demo	XML Driven Demo	class demo.databox.xmldriven.X...
Conversion Demo	Conversion Demo	class demo.databox.conversion....
Network		
Tree		

Hiding Root Node

```
TTree tree = table.getTree(); //hide root node
tree.setRootVisible(false);
```

Demos	Name	DemoClass
 DataBox		
 Monitor Property Demo	Monitor Property Demo	class demo.databox.monitorproperty.Monitor...
 Monitor Element Demo	Monitor Element Demo	class demo.databox.monitorelement.Monitor...
 XML Driven Demo	XML Driven Demo	class demo.databox.xmldriven.XMLDrivenDe...
 Conversion Demo	Conversion Demo	class demo.databox.conversion.Conversion...
 Network		
 Tree		
 Sheet		

Using Tree Table Component

Cause TreeTable is a combination of a TWaver Tree and a TWaver Table, so it provides the features of TWaver Tree and TWaver Table.

Please see section [Using Table Component](#) and [Using Tree Component](#) for more details.

Using Alarm Table Component

Alarm table component (`twaver.table.TAlarmTable`) is an extension of TWaver table. Alarm table used to display alarm objects. Alarm table connects to a DataBox and gets all alarm objects from alarm model of the DataBox, then lists them on the table with predefined columns.

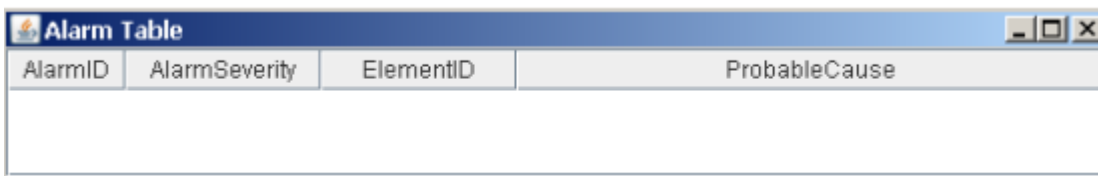
Alarm table is derived from TWaver table, so it inherit all features of the basic table like customized columns, sorting, paging control, locking, popup menu and filters. Please see more details in section [Using Table Component](#).

- [Creating Alarm Table Component](#)
- [Customizing Alarm Table Component](#)
- [Operations for TAlarmTable](#)
- [Using AlarmVisibleFilter](#)
- [Working with DataBox and AlarmModel](#)

Creating Alarm Table Component

Normally we need a DataBox to create an alarm table. In this case the table will connect to the DataBox, display and monitor the alarm model of the DataBox. Following snippet creates an empty alarm table component and displays it in a JFrame:

```
JFrame frame = new JFrame("Alarm Table");
TDataBox box = new TDataBox();
TAlarmTable table = new TAlarmTable(box);
frame.setContentPane(new JScrollPane(table));
frame.setSize(400, 100);
TWaverUtil.centerWindow(frame);
frame.show();
```



Constructors of alarm table component:

Method	Description
TAlarmTable(TDataBox box)	Create alarm table with default columns, and connect to specified DataBox.
TAlarmTable(TDataBox box , String tableName)	Create alarm table with a configuration name which should be defined in TWaver.XML. The table connects to specific DataBox.
TAlarmTable(TDataBox box , TTableColumn[] columns)	Create alarm table with given columns. The table will connect to the specified DataBox.

There are many default columns. These are visible acquiescently:

Alarm.PROPERTY_ALARMID
Alarm.PROPERTY_ALARMSEVERITY
Alarm.PROPERTY_ACKED
Alarm.PROPERTY_ELEMENTID
Alarm.PROPERTY_PROBABLECAUSE

```
private static TTableColumn[] createDefaultColumns() {
    return new TTableColumn[] {
        createColumn(Alarm.PROPERTY_ALARMID, 60, true),
        createColumn(Alarm.PROPERTY_ALARMSEVERITY, 100, true),
        createColumn(Alarm.PROPERTY_ACKED, 50, true).setRenderer(new AcknowledgedRenderer()),
        createColumn(Alarm.PROPERTY_ELEMENTID, 100, true).setRenderer(new ElementIDRenderer()),
        createColumn(Alarm.PROPERTY_PROBABLECAUSE, 300, true),
        createColumn(Alarm.PROPERTY_ACKUSERID, 80, false),
        createColumn(Alarm.PROPERTY_CLEARUSERID, 60, false),
        createColumn(Alarm.PROPERTY_CLEARED, 40, false),
        createColumn(Alarm.PROPERTY_ALARMSTYPE, 60, false),
        createColumn(Alarm.PROPERTY_TRENDINDICATION, 40, false),
        createColumn(Alarm.PROPERTY_RAISEDTIME, 120, false),
        createColumn(Alarm.PROPERTY_ACKTIME, 120, false),
```

```
createColumn(Alarm.PROPERTY_CLEAREDTIME, 120, false),
createColumn(Alarm.PROPERTY_LASTCHANGEDTIME, 120, false),
createColumn(Alarm.PROPERTY_CORRELATEDALARMS, 60, false),
createColumn(Alarm.PROPERTY_SPECIFICPROBLEM, 200, false),
createColumn(Alarm.PROPERTY_ADDITIONALTEXT, 200, false),
createColumn(Alarm.PROPERTY_COMMENTS, 200, false),
createColumn(Alarm.PROPERTY_PROPOSEDREPAIRACTION, 200, false),
};
}

private static TTableColumn createColumn(String name, int width, boolean visible) {
    return new TTableColumn(name, TWaverUtil.getString("alarm.property." + name), width).setVisible(visible);
}
```

Customizing Alarm Table Component

The default alarm table has been well designed to present alarms, include default visible columns, column preferred column width, cell renderers and editors, popup menus and so on. But you can always change the default setting of alarm table, or customize it to need your specific needs. All features mentioned in section [Using Table Component](#) can be customize here, please see it for details.

Herein we will give you a sample how to define a new alarm property and display it in alarm table as a customized column.

- Add "customIcon" property to alarm object
- Display "customIcon" column in alarm table
- Render "customIcon" property as a user icon
- Property "customIcon" can be changed in line with a predefined user list

Define "CustomIcon" Property

First we define CustomIcon class to encapsulate "customIcon" information.

```
class CustomIcon {
    private String name = null;
    private Icon icon = null;
    public CustomIcon(String name, Icon icon) {
        this.name = name;
        this.icon = icon;
    }
    String getName() {
        return name;
    }
    void setName(String name) {
        this.name = name;
    }
    Icon getIcon() {
        return icon;
    }
    void setIcon(Icon icon) {
        this.icon = icon;
    }
}
```

Then we need to attach the "customIcon" information to alarm object. You can define the new property as standard java bean property like "Alarm.setCustomIcon" and "Alarm.getCustomIcon", or as a client property, a more easier and preferred way in TWaver:

```
for (int i = 1; i < 5; i++) {
    AlarmSeverity severity = TWaverUtil.getRandomNonClearedSeverity();
    Alarm alarm = new Alarm(new Integer(50 + i * 10), node5.getID(), severity);
    alarm.putClientProperty("CustomIcon", new CustomIcon("Icon_" + i, TWaverUtil.getIcon("/demo/resource/images/attachment" + i + ".png")));
    box.getAlarmModel().addAlarm(alarm);
}
```

Add "CustomIcon" Column

To display the CustomIcon property on table, we need to create a new table column and add it into table model. That's very easy:

```
TTableColumn ownerColumn = new TTableColumn("CustomIcon", "CustomIcon");
ownerColumn.setPreferredWidth(50);
```

```
alarmTable.getTableModel().addColumn(ownerColumn);
```

The new column has name "customIcon" and display name "CustomIcon". TWaver will try to get "customIcon" property from Alarm object with java beans introspection. If no found, then try to get it from client properties.

Alarm Table					
AlarmID	AlarmSeverity	Acked	ElementID	Pro...	CustomIcon
60	Warning				demo.swing.TableTest\$CustomIc...
70	Indeterminate				demo.swing.TableTest\$CustomIc...
80	Warning				demo.swing.TableTest\$CustomIc...
90	Minor				demo.swing.TableTest\$CustomIc...

CustomIcon property added into alarm table

Set Renderer & Editor for "CustomIcon" Column

So far the property has been introspected and displayed on alarm table. In order to render and edit the "customIcon" property with a more friendly way, we now create and set renderer/editor for this new column:

```
//create and set a renderer.
ownerColumn.setRenderer(new DefaultTableCellRenderer() {
    public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean
hasFocus, int row, int column) {
        JLabel label = (JLabel) super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);
        if (value instanceof CustomIcon) {
            CustomIcon owner = (CustomIcon) value;
            label.setIcon(owner.getIcon());
            label.setText(owner.getName());
        }
        return label;
    }
});
```

Alarm Table					
AlarmID	AlarmSeverity	Acked	ElementID	ProbableCause	CustomIcon
60	Minor				Icon_1
70	Critical				Icon_2
80	Warning				Icon_3
90	Indeterminate				Icon_4

```
//create and set an editor.
//1. Make column editable.
ownerColumn.setEditable(true);
//2. Create a JComboBox as the editor options.
JComboBox options = new JComboBox();
for (int i = 1; i < 5; i++) {
    options.addItem(new CustomIcon("Icon_" + i, TWaverUtil.getIcon("/demo/resource/images/attachment" + i
+ ".png")));
}
//3. Make a renderer for the JComboBox.
options.setRenderer(new DefaultListCellRenderer() {
    public Component getListCellRendererComponent(JList list, Object value, int index, boolean isSelected, boolean
cellHasFocus) {
        super.getListCellRendererComponent(list, value, index, isSelected, cellHasFocus);
        if (value instanceof CustomIcon) {
            CustomIcon icon = (CustomIcon) value;
            this.setIcon(icon.getIcon());
            this.setText(icon.getName());
        }
    }
});
```

```

return this;
}
});
//4. Create editor and set it to column.
ownerColumn.setEditor(new DefaultCellEditor(options));

```

Alarm Table					
AlarmID	AlarmSeverity	Acked	ElementID	ProbableCause	CustomIcon
60	Minor				Icon_1
70	Critical				Icon_2
80	Warning				Icon_3
90	Indeterminate				Icon_4

Operations for TAlarmTable

TAlarmTable extends from TTable and provides more methods to handle Alarm data.

Vector twaver.table.TAlarmTable.getRowDataByAlarmID(Object alarmID)

Get table row data via specified alarm id.

Alarm twaver.table.TAlarmTable.getAlarmByOID(Object oid)

Get alarm object by table row OID.

Alarm twaver.table.TAlarmTable.getAlarmByRowData(Vector rowData)

Get alarm object by table row data.

List twaver.table.TAlarmTable.getAllCheckedAlarms()

Gets all checked alarm objects.

List twaver.table.TAlarmTable.getAllSelectedAlarms()

Gets all selected alarm objects.

List twaver.table.TAlarmTable.getCurrentPageAlarms()

Gets all alarms of the current page.

List twaver.table.TAlarmTable.getPublishedAlarms()

Gets all published alarms.

Alarm twaver.table.TAlarmTable.getAlarmByRowIndex(int rowIndex)

Gets alarm by table row index.

A example for handle alarms by using popup menu:

```
//init data
TDataBox box = new TDataBox();
Node node1 = new Node();
box.addElement(node1);
Node node2 = new Node();
box.addElement(node2);
final AlarmModel model = box.getAlarmModel();
for (int i = 0; i < 10; i++) {
    model.addAlarm(new Alarm("alarm1_" + i, node1.getID(), AlarmSeverity.getNonClearedRandomSeverity()));
}
for (int i = 0; i < 10; i++) {
    model.addAlarm(new Alarm("alarm2_" + i, node2.getID(), AlarmSeverity.getNonClearedRandomSeverity()));
}
final TAlarmTable table = new TAlarmTable(box);

//add double click action
table.addAlarmDoubleClickedActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) {
        Alarm alarm = (Alarm)e.getSource();
        AlarmUI.createFrameAlarmUI(alarm).setVisible(true);
    }
});

//add popup menu
AbstractAction clearAction=new AbstractAction(){
    public void actionPerformed(ActionEvent e) {
        List alarms=table.getAllSelectedAlarms();
        for(int i=0;i<alarms.size();i++){
            ((Alarm)alarms.get(i)).setCleared(true);
        }
    }
};
clearAction.putValue(Action.NAME, "Clear alarms");

AbstractAction ackedAction=new AbstractAction(){
    public void actionPerformed(ActionEvent e) {
        List alarms=table.getAllSelectedAlarms();
        for(int i=0;i<alarms.size();i++){
            if(!((Alarm)alarms.get(i)).isAked()){
                ((Alarm)alarms.get(i)).setAked(true);
            }
        }
    }
};
```

```

    }
    }
    };
    ackedAction.putValue(Action.NAME, "Acked alarms");
    final JPopupMenu popMenu = new JPopupMenu();
    popMenu.add(new JMenuItem(ackedAction));
    popMenu.add(new JMenuItem(clearAction));
    table.setTableBodyPopupMenuFactory(new TTablePopupMenuFactory() {
        public JPopupMenu getPopupMenu(TTable table, MouseEvent e) {
            return popMenu;
        }
    });
    //enable remove alarms when alarm cleared. The default value is false.
    model.setAutoRemoveAlarmWhenCleared(true);
    JFrame frame = new JFrame();
    frame.setTitle("Alarm table demo");
    frame.getContentPane().add(new JScrollPane(table), BorderLayout.CENTER);
    frame.setSize(600, 400);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    TWaverUtil.centerWindow(frame);
    frame.setVisible(true);

```

Run the example:

AlarmID	AlarmSeverity	Acked	ElementID	ProbableCause
alarm1_0	Minor	<input type="checkbox"/>		
alarm1_1	Critical	<input type="checkbox"/>		
alarm1_2	Warning	<input type="checkbox"/>		
alarm1_3	Major	<input type="checkbox"/>		
alarm1_4	Major	<input type="checkbox"/>		
alarm1_5	Warning	<input type="checkbox"/>		
alarm1_6	Critical	<input type="checkbox"/>		
alarm1_7	Major	<input type="checkbox"/>		
alarm1_8	Minor	<input type="checkbox"/>		
alarm1_9	Indeterminate	<input type="checkbox"/>		

Using AlarmVisibleFilter

Alarm visible filter can be used in Alarm Table to filter out some alarm data. If you only want see critical alarms, you can do it like this way:

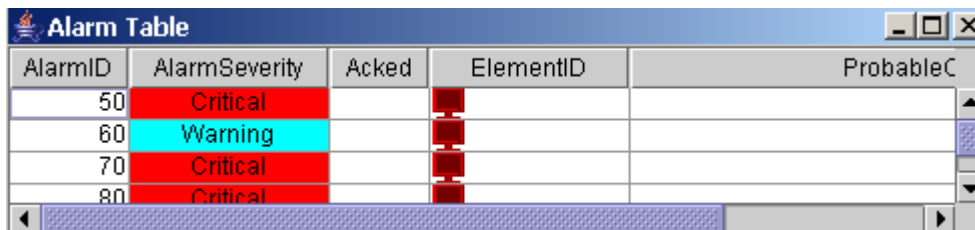
```
TAlarmTable table = new TAlarmTable(box);
table.addVisibleFilter(new AlarmVisibleFilter(){
    public boolean isVisible(Alarm alarm) {
        return alarm.getAlarmSeverity() == AlarmSeverity.CRITICAL;
    }
});
```

Working with DataBox and AlarmModel

Once the alarm table is created and connects to some DataBox, it will always display the alarm informations of the DataBox. In section [Using Alarm Model](#) we know that all alarm objects are managed by alarm model of DataBox. So we can handle alarm model to change the data of alarm table. That is, alarm table is a component driven by DataBox and alarm model.

Each alarm in alarm model will be present as one table row in alarm table. Now we create several alarms and put them into alarm model and see what happen on alarm table.

```
Node node5 = new Node();
box.addElement(node5);
node5.getAlarmState().clear();
for (int i = 0; i < 4; i++) {
    AlarmSeverity severity = TWaverUtil.getRandomNonClearedSeverity();
    Alarm alarm = new Alarm(new Integer(50 + i * 10), node5.getID(), severity);
    box.getAlarmModel().addAlarm(alarm);
}
```



AlarmID	AlarmSeverity	Acked	ElementID	ProbableC
50	Critical			
60	Warning			
70	Critical			
80	Critical			

Setting Databox by this method:

```
public void setDataBox(TDataBox box)
```

Setting Databox and columns by this method:

```
public void setDataBox(TDataBox box, TTableColumn[] columns)
```

Using AlarmOverview Component

The alarm overview component (`twaver.table.TalarmOverview`) is designed to statistic and display alarm information of a DataBox. It is based on Swing `JTable` and `JPanel`, and displays alarm statistic information in a two-dimensional table format or chart.

- [Alarm Statistic Table](#)
- [Creating AlarmOverview Component](#)

Alarm Statistic Table

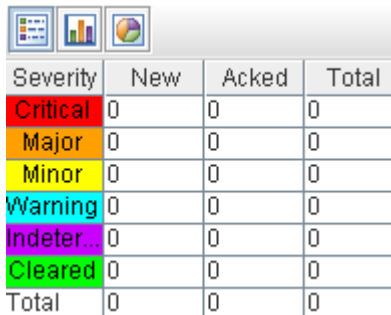
The alarm statistic table in AlarmOverview component used to display the alarm summary information of all elements contained in the connected DataBox. It also provides a bunch of methods to retrieve the total amount of the alarms.

Creating AlarmOverview Component

The AlarmOverview component can be created in a similar way. You can set a TDataBox for it, also you can set the alarm severity for it.

Creating an AlarmOverview

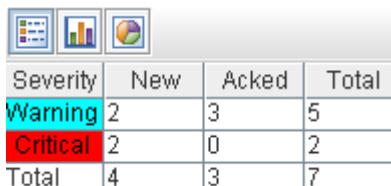
```
//Create AlarmOverview component by specified DataBox.
TDataBox box=new TDataBox();
TAlarmOverview alarmOverview=new TAlarmOverview(box);
```



Severity	New	Acked	Total
Critical	0	0	0
Major	0	0	0
Minor	0	0	0
Warning	0	0	0
Indeter...	0	0	0
Cleared	0	0	0
Total	0	0	0

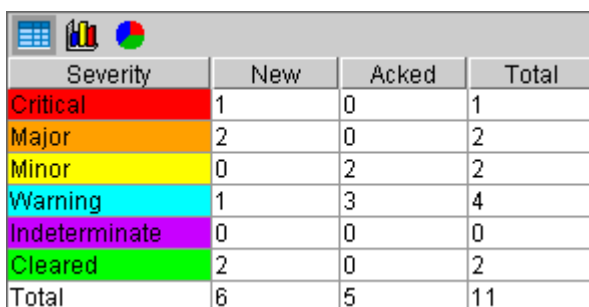
Creating an AlarmOverview with an alarm severity list

```
TDataBox box=new TDataBox();
TAlarmOverview alarmOverview=new TAlarmOverview(box);
List alarmSeverities=new ArrayList();
alarmSeverities.add(AlarmSeverity.WARNING);
alarmSeverities.add(AlarmSeverity.CRITICAL);
TAlarmOverview alarmOverView=new TAlarmOverview(box,alarmSeverities);
```



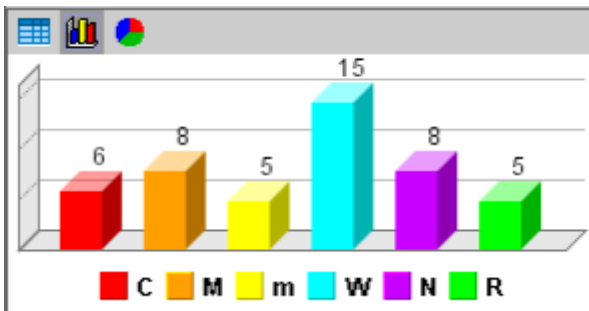
Severity	New	Acked	Total
Warning	2	3	5
Critical	2	0	2
Total	4	3	7

Once alarm overview component connects to a DataBox, it will monitor the alarm state of each element of the databox, and display the summary information with a table and chart. You can change the displaying style by clicking icon buttons on the top.

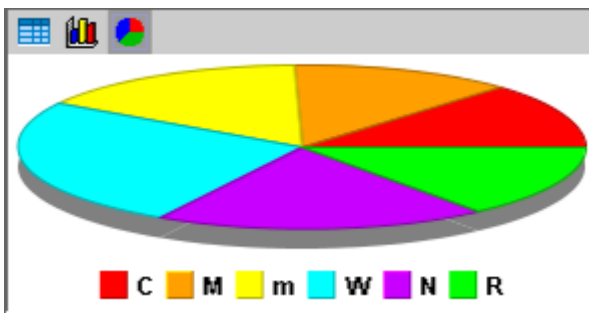


Severity	New	Acked	Total
Critical	1	0	1
Major	2	0	2
Minor	0	2	2
Warning	1	3	4
Indeterminate	0	0	0
Cleared	2	0	2
Total	6	5	11

Display Alarm Counts with Table



Display Alarm Counts with Bar Chart



Display Alarm Counts with Pie Chart

Using List Component

TWaver List component is one predefined graphical component of TWaver. TWaver List component is extended from Swing JList and add more features. Also, as an instance of TView, List component works with DataBox.

- [Creating List Component](#)
- [Using Checkable Filter](#)
- [Using List Visible Filter](#)
- [Using Selection Mode](#)
- [Using Sort Comparator](#)

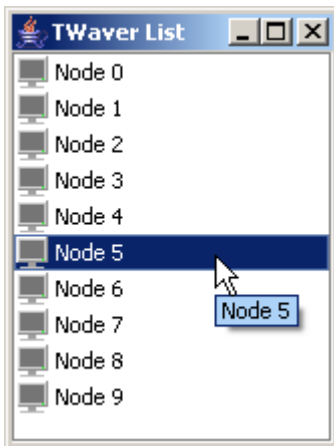
Creating List Component

TWaver List works with DataBox. Following code creates a List with a DataBox, and then displays it in a JFrame.

```
TDataBox box=new TDataBox();
TList list=new TList(box);

for(int i=0;i<10;i++){
    Element node=new Node();
    node.setName("Node "+i);
    box.addElement(node);
}

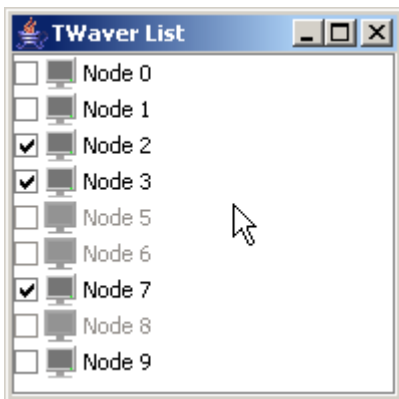
JFrame frame=new JFrame();
frame.setContentPane(new JScrollPane(list));
frame.setTitle("TWaver List");
TWaverUtil.centerWindow(frame);
frame.setVisible(true);
```



Using Checkable Filter

Use `CheckableFilter` to determine which items are checkable under check selection mode. Following code shows you how to make some List items not checkable:

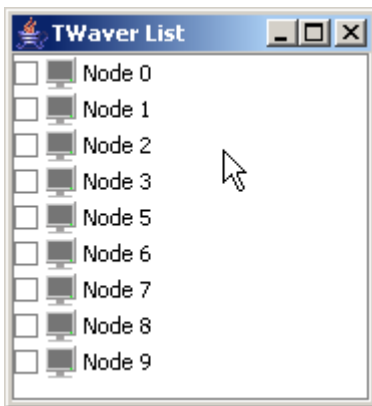
```
list.setCheckableFilter(new CheckableFilter(){  
    public boolean isCheckable(Element element) {  
        return !element.getName().equals("Node 5") &&  
            !element.getName().equals("Node 6") &&  
            !element.getName().equals("Node 8");  
    }  
});
```



Using List Visible Filter

Just like other TWaver components, List also support visible filter. Following code used to hide "Node 4":

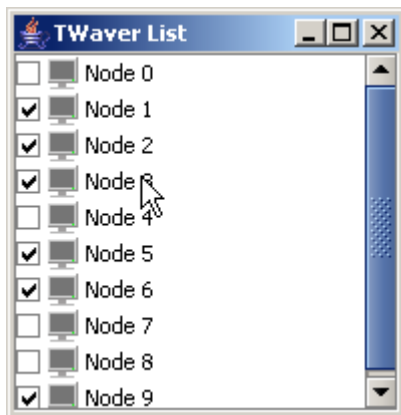
```
list.addVisibleFilter(new VisibleFilter(){  
    public boolean isVisible(Element element) {  
        return !element.getName().equals("Node 4");  
    }  
});
```




Using Selection Mode

Like TWaver Table component, List component support default selection mode and check selection mode. In check selection mode, a check box will used for each item to check.

```
list.setTListSelectionMode(TList.CHECK_SELECTION);
```

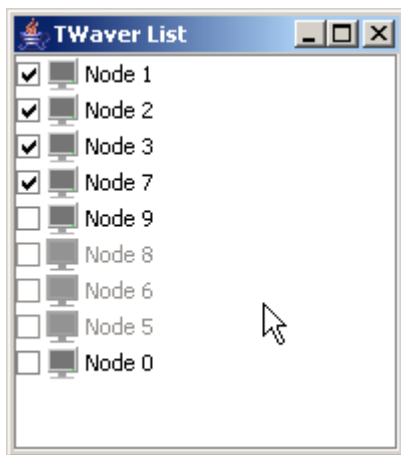


 please do not confuse method `setTListSelectionMode` (TWaver defined) and method `setSelectionMode` (Swing defined).

Using Sort Comparator

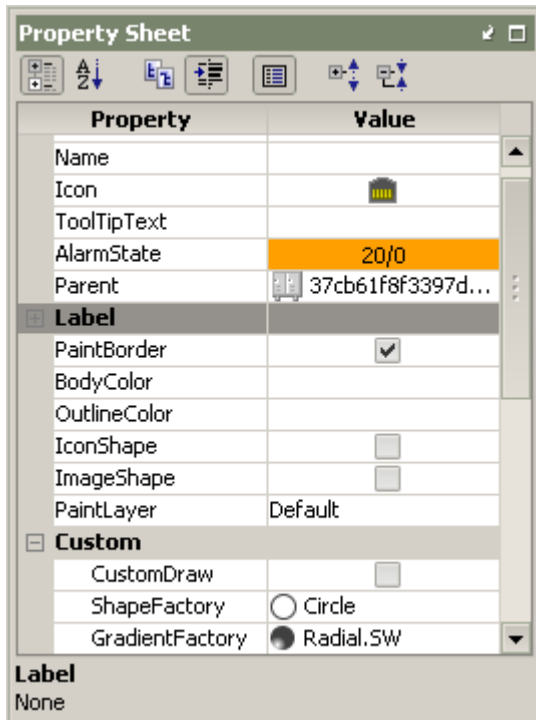
You can use a comparator to sort the list item order. Following codes sort the List items in a customized rule: put checked items on top and unchecked items to bottom:

```
list.setSortComparator(new Comparator(){  
    public int compare(Object o1, Object o2) {  
        if(((Element)o1).isSelected()){  
            return -1;  
        }else{  
            return 1;  
        }  
    }  
    public boolean equals(Object obj) {  
        return false;  
    }  
});
```



Using PropertySheet Component

TPropertySheet component is a table view used to display all properties of an element instance. PropertySheet component display the element properties in the table with two columns, left is the property name, right is the property value, as following:



TPropertySheet support property categories. A set of properties can be put grouped in a category. The property category also can nested.

TPropertySheet can receive an element instance, iterate display all its properties. Each row of the sheet is representing one property. Each property has its renderer and editor. Renderer responsible for render this property value and the editor responsible for edit the property.

The properties of an Element are defined as the JavaBeans specification. You can find more information about JavaBeans technology at:

<http://Java.sun.com/docs/books/tutorial/Javabeans/>.

- [BeanInfo and BeanInfoLoader](#)
- [BeanInfo Format](#)
- [Customizing PropertySheet Style](#)
- [Define BeanInfo by XML](#)
- [Extending BeanInfo](#)
- [Properties Grouping](#)
- [PropertySheet Events](#)
- [TPropertySheet Cell Editor](#)
- [TPropertySheet Cell Renderer](#)
- [TWaver and JavaBeans](#)
- [Using Property Sheet Pane](#)

BeanInfo and BeanInfoLoader

In JavaBeans specification, all bean features like method, properties, event etc are defined by BeanInfo. Thus, Beans tools can get/set the beans property value.

In TWaver, all Element class is described by a XML file which has the same name with the Element. TWaver use an information loader (BeanInfoLoader) loads all bean informations from this XML file.

When BeanInfoLoader loading the bean information, it will try to find this XML file such as Node.XML from disk, in the same time, it will also load all it's super class/interface to get all parent's character. All these bean informations are cached by TWaver once it was loaded.

BeanInfo Format

BeanInfo file are the simple XML format. It has a node <beaninfo> as the root node and a serial of <attribute> nodes as the child node. Each <attribute> node represents one property.

As per this specifaction, you can define all informations of an Element very easily. Such as the Location property: the location property is peculiar property of BaseElement (Link no location information), so we define it in BaseElement.XML file. It's not a client property, and you can read/write its value, the related method is getLocation/setLocation. So we can compose it as follows:

```
<attribute
  name="location"
  read="true"
  readMethod="getLocation"
  editable="true"
  writeMethod="setLocation"/>
```

For read only properties, TPropertySheet will not allow to edit/change its value. A typically example is the ID of Element.

For client properties (which visit via putClientProperty/getClientProperty method), you should set the 'clientProperty' to true, and set the JavaClass and clientPropertyKey value. Here is an example:

```
<attribute
  name="link.texture"
  clientProperty="true"
  JavaClass="Java.lang.Boolean"
  clientPropertyKey="link.texture"/>
```

Customizing PropertySheet Style

Editable

setEditable(boolean isEditable)

- Using Renderer

setPropertyRenderer(TableCellRenderer propertyNameRenderer)

- Disable Category

//Show properties in categories

sheet.setMode(TPropertySheet.VIEW_AS_CATEGORIES);

//show properties in a flat list

sheet.setMode(TPropertySheet.VIEW_AS_FLAT_LIST);

- Sort

//Use a new category sort comparator

sheet.setCategorySortingComparator(Comparator categorySortingComparator);

//Use a new property sort comparator

sheet.setPropertySortingComparator(Comparator propertySortingComparator);

//whether sort categories

sheet.setSortingCategories(true);

//whether sort properties

sheet.setSortingProperties(true);

- Using Indent

//Display extra indent for property name

sheet.setPropertyExtraIndent(boolean isPropertyExtraIndent);

- Other Settings

//Set property distinct level.

sheet.setPropertyDistinctLevel(TPropertySheet.DISTINCT_LEVEL_ELEMENT)

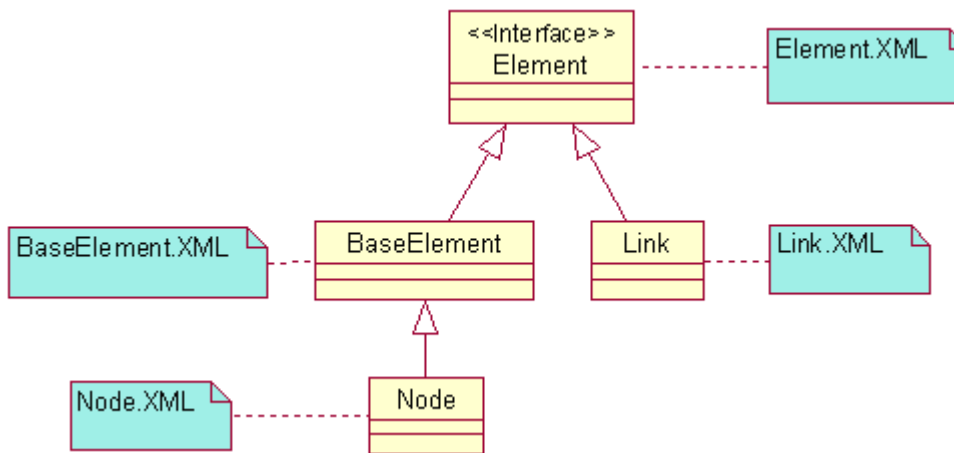
sheet.setPropertyDistinctLevel(TPropertySheet.DISTINCT_LEVEL_CLASS)

//Set a property visible filter

sheet.setElementAttributeVisibleFilter(ElementAttributeVisibleFilter filter)

Define BeanInfo by XML

In TWaver, each Element implementation can be defined by a BeanInfo XML file:



Each XML describe the related Element properties. XML files are also can extends from its parent XML file. For example, if Node class defines "image" property, then all subclass of Node will automatically inherit. Thus, each BeanInfo file can only describe its own properties. Here is the example for Element and BaseElement BeanInfo describe file:

Element.xml:

```

<beaninfo>

<category name="system">
<category name="system.basic">
<attribute
name="ID"
editable="false"/>
...
</category>
</category>

<category name="system">
<category name="system.border">
<attribute
clientPropertyKey="border.insets"
javaClass="java.lang.Integer"/>
...
</category>
</category>

<category name="system">
<category name="system.alarm">
<attribute
name="alarmState"
enableBatch="false"/>
...
</category>
</category>

<category name="system">
<category name="system.attachment">
<attribute
clientPropertyKey="attachment.position"
editor="twaver.table.editor.EnumTypeEditor@twaver.position|false"
renderer="twaver.table.renderer.EnumTypeRenderer@twaver.position"/>

```

```
...
</category>
</category>

<category name="system">
  <category name="system.label">
    <attribute
      name="displayName"
      editor="twaver.table.editor.HTMLTextEditor"/>
  ...
  </category>
</category>

<category name="system">
  <category name="system.message">
    <attribute
      clientPropertyKey="StateIcon:attachment.message"
      displayName="i18nKey:twaver.Element.message"
      javaClass="java.lang.Boolean"/>
  ...
  </category>
</category>

</beaninfo>
```

BaseElement.XML:

```
<beaninfo>

<category name="system">
  <category name="system.basic">
    <attribute
      name="location"
      editable="true"/>
    <attribute
      name="height"
      editable="false"/>
    <attribute
      name="width"
      editable="false"/>
    <attribute
      javaClass="java.awt.Color"
      clientPropertyKey="body.color"/>
    <attribute
      javaClass="java.lang.Boolean"
      clientPropertyKey="body.fill"/>
    <attribute
      javaClass="java.lang.Boolean"
      clientPropertyKey="body.raised"/>
  </category>
</category>

<category name="system">
  <category name="system.custom">
    <attribute
      javaClass="java.lang.Boolean"
      clientPropertyKey="custom.draw"/>
    <attribute
      clientPropertyKey="custom.draw.shape.factory"
      editor="twaver.table.editor.EnumTypeEditor@twaver.shape|false"
      renderer="twaver.table.renderer.EnumTypeRenderer@twaver.shape"/>
    <attribute
      javaClass="java.lang.Boolean"
```

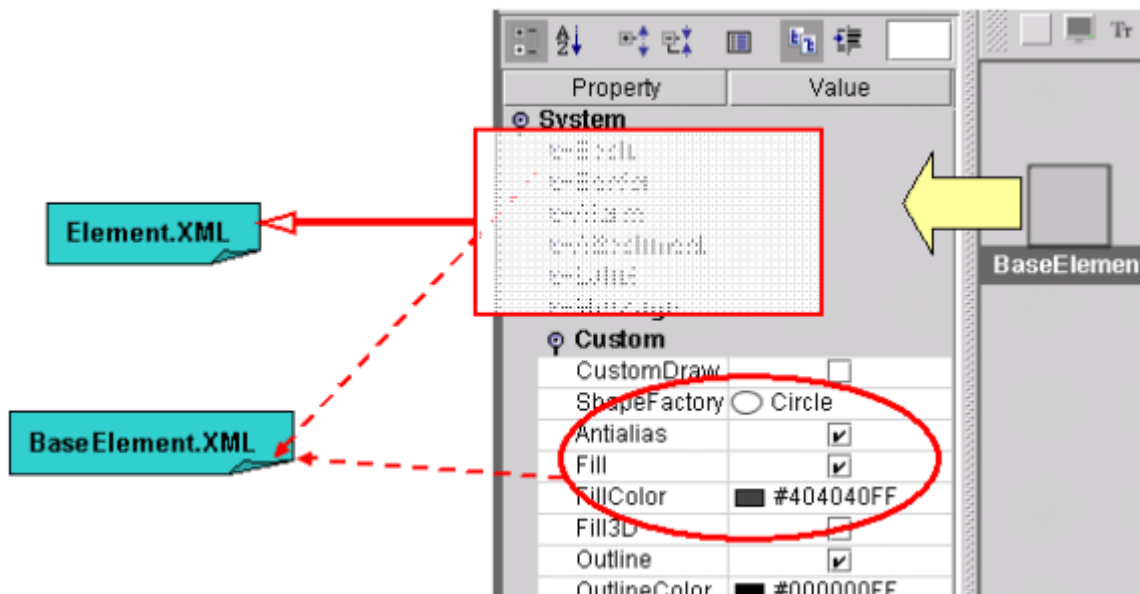
```

        clientPropertyKey="custom.draw.antialias"/>
    <attribute
        javaClass="java.lang.Boolean"
        clientPropertyKey="custom.draw.fill"/>
    <attribute
        javaClass="java.awt.Color"
        clientPropertyKey="custom.draw.fill.color"/>
    <attribute
        javaClass="java.lang.Boolean"
        clientPropertyKey="custom.draw.fill.3d"/>
    <attribute
        javaClass="java.lang.Boolean"
        clientPropertyKey="custom.draw.outline"/>
    <attribute
        javaClass="java.awt.Color"
        clientPropertyKey="custom.draw.outline.color"/>
    <attribute
        clientPropertyKey="custom.draw.outline.stroke"
        editor="twaver.table.editor.StrokeEditor"
        renderer="twaver.table.renderer.StrokeRenderer"/>
    <attribute
        javaClass="java.lang.Boolean"
        clientPropertyKey="custom.draw.outline.3d"/>
    <attribute
        javaClass="java.lang.Boolean"
        clientPropertyKey="custom.draw.gradient"/>
    <attribute
        javaClass="java.awt.Color"
        clientPropertyKey="custom.draw.gradient.color"/>
    <attribute
        clientPropertyKey="custom.draw.gradient.factory"
        editor="twaver.table.editor.EnumTypeEditor@twaver.gradient|false"
        renderer="twaver.table.renderer.EnumTypeRenderer@twaver.gradient"/>
    <attribute
        javaClass="java.lang.Boolean"
        clientPropertyKey="custom.draw.default.border"/>
</category>
</category>

</beaninfo>

```

With these definitions, when you add a BaseElement object in TWaver Editor, it will display like this:



Extending BeanInfo

To extend new Managed Object and new property, you need to:

- Extends new Business Class
- Compose new XML file

In order to demonstrate this function, now we define a new Element called LightBulb. It extends from BaseElement, and adds a new property: turn on/turn off. When this property changed, we change LightBulb image to update its new status. Now first we define the LightBulb class:

```
public class LightBulb extends Node {
    private boolean shining = false;
    public LightBulb() {
        super();
        init();
    }
    public LightBulb(boolean shining) {
        this();
        this.setShining(shining);
    }
    public LightBulb(Object id) {
        super(id);
        init();
    }
    private void init() {
        this.setImage("/off.png");
    }
    public boolean isShining() {
        return shining;
    }
    public void setShining(boolean shining) {
        boolean oldValue = this.shining;
        this.shining = shining;
        //fire the property change event.
        if (oldValue != shining) {
            this.firePropertyChange("shining", oldValue, shining);
        }
        if (shining) {
            this.setImage("/on.png");
        } else {
            this.setImage("/off.png");
        }
    }
}
```

Then we define the LightBulb BeanInfo XML file:

LightBulb.xml:

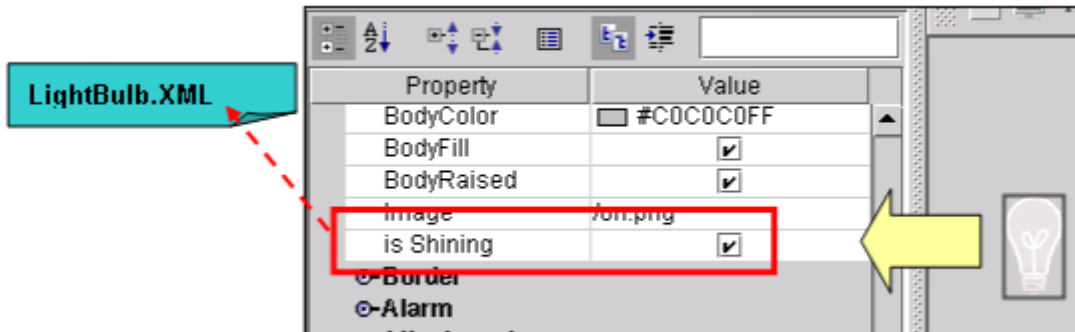
```
<beaninfo>
<category name="system">
<category name="system.basic">
<attribute
name="shining"
readMethod="isShining"
write="true"
displayName="is Shining"
writeMethod="setShining"/>
```

```
</category>
</category>
</beaninfo>
```

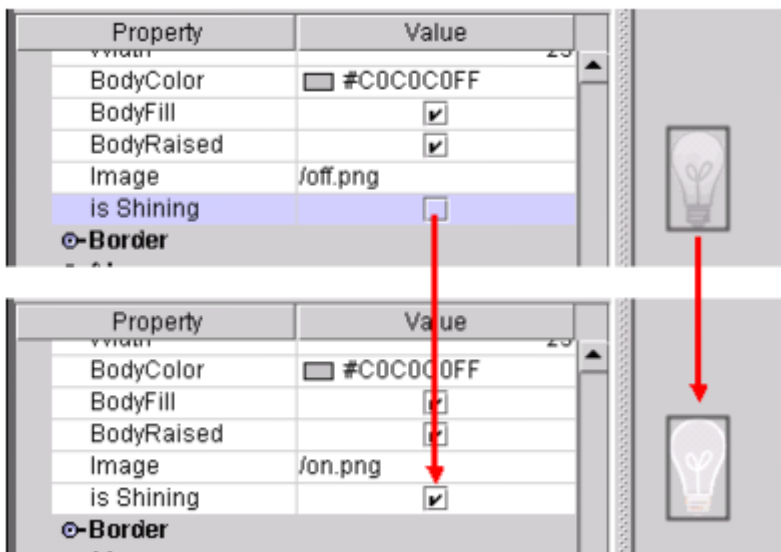
Register Config File

```
TWaverUtil.registerBeanInfo(LightBulb.class, "/LightBulb.xml");
```

Now you can add LightBulb class into TWaver Editor. It'll look like:



When you click 'shining' checkbox to change the value, TWaver will automatically update the related property value, and change object's image at once.



Properties Grouping

To making a fine and clear look PropertySheet component, TWaver provides the ability to group object properties into more categories. It use symbol plus and minus for category node. The categories can be defined by API or XML.

- [Grouping by API](#)
- [Grouping by XML](#)

Grouping by API

You can create property categories with APIs. See TWaver JavaDoc for more details.

Grouping by XML

Define new categories in TWaver.xml file:

```
<propertysheet>
<category name="system" isExpend="false"/>
<category name="mine"
displayName="My Property"
isExpend="true"/>
</propertysheet>
```

For category element:

name

The name of the category. TWaver will try to find it by a resource key, which is "sheet.category"+name. For instance, if you define a new category with name "system", then you should define a resource item in property file like "sheet.category.system=System".

displayName

The display name of the category. If display name was defined, TWaver will use it as the final display string directly.

isExpend

Category expended status. The default value is "true".

Change the Element bean XML file, and assign properties into the defined categories.

For instance, if you defined a new object MyNode, the MyNode.XML should be written like this:

```
<attribute clientPropertyKey="MyProp1"
categoryName="mine"/>
<attribute clientPropertyKey="MyProp2"
categoryName="mine"/>
```

Define MyNode class:

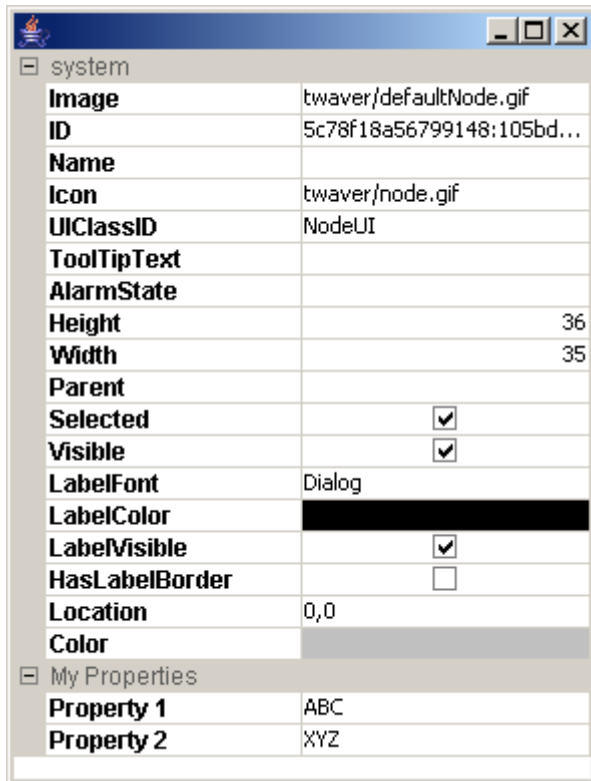
```
public class MyNode extends twaver.Node {
    public MyNode() {
        super();
        init();
    };
    public MyNode(Object id) {
        super(id);
        init();
    }

    private void init() {
        this.putClientProperty("MyProp1", "ABC");
        this.putClientProperty("MyProp2", "XYZ");
    }
}
```

Redirect TWaver resource agent:

```
public static void main(String[] args) {
```

```
...
TWaverUtil.setResourceAgent(MyNode.class);
...
}
```



PropertySheet Events

Use PropertySheetListener to monitoring property change for PropertySheet. Including category expand and collapse, single/double click, property sheet model and property values change.

Event types defined in twaver.table. PropertySheetEvent:

- PropertySheetEvent.VALUE_CHANGED = 1; // Property value changed
- PropertySheetEvent.MODEL_PUBLISHED = 2; // Data published
- PropertySheetEvent.CATEGORY_EXPANDED = 3; // Category expanded
- PropertySheetEvent.CATEGORY_COLLAPSED = 4; // Category collapsed
- PropertySheetEvent.ROW_CLICKED = 5; // Single click
- PropertySheetEvent.ROW_DOUBLE_CLICKED = 6; // Double click


```
TDataBox box=new TDataBox();
TPropertySheet sheet=new TPropertySheet(box);

PropertySheetListener l=new PropertySheetListener(){
    public void propertySheetChanged(PropertySheetEvent event) {
        if(event.getRowIndex() >= 0){
            if(PropertySheetEvent.CATEGORY_COLLAPSED==event.getType()){
                System.out.println("Row:"+event.getRowIndex()+"_CATEGORY_COLLAPSED");
            }else if(PropertySheetEvent.CATEGORY_EXPANDED==event.getType()){
                System.out.println("Row:"+event.getRowIndex()+"_CATEGORY_EXPANDED");
            }else if(PropertySheetEvent.MODEL_PUBLISHED==event.getType()){
                System.out.println("Row:"+event.getRowIndex()+"_MODEL_PUBLISHED");
            }else if(PropertySheetEvent.ROW_CLICKED==event.getType()){
                System.out.println("Row:"+event.getRowIndex()+"_ROW_CLICKED");
            }else if(PropertySheetEvent.ROW_DOUBLE_CLICKED==event.getType()){
                System.out.println("Row:"+event.getRowIndex()+"_ROW_DOUBLE_CLICKED");
            }else if(PropertySheetEvent.VALUE_CHANGED==event.getType()){
                System.out.println("Row:"+event.getRowIndex()+"_VALUE_CHANGED");
            }else{
                System.out.println("Row:"+event.getRowIndex()+event.getType());
            }
        }
    }
};
sheet.addPropertySheetListener(l);

Element element=new Node();
box.addElement(element);
element.setSelected(true);

JFrame frame = new JFrame();
frame.setTitle("Property sheet event test");
frame.getContentPane().add(new JScrollPane(sheet),BorderLayout.CENTER);
frame.setSize(300, 500);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
TWaverUtil.centerWindow(frame);
frame.setVisible(true);
```

Run the program:

Property sheet event test	
Property	Value
System	
Basic	
ID	ad64376b7405eff1:-35868c1f:11dd...
Name	
Selected	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Icon	
ToolTipText	

Click a row, the result is:

Row:2_ROW_CLICKED

TPropertySheet Cell Editor

TPropertySheet can display Element properties; it can also edit property value. Similar to Renderer, it defined as cell Editor.

The basic Editor principle is: when TPropertySheet edit some property, it will try to get the corresponding Editor of the property class type firstly. Then the Editor will responsible for the cell editing. For more detail about JTable and CellEditor, please reference Java Swing JTable package.

In TPropertySheet component, each property is different type. Each different type can be edit by different Editor. For example, you maybe want edit Boolean type with a check box component; edit a Color with a color picking dialog, etc. So, we can define many editors to render most class type.

- [Customizing Editor](#)
- [Managing Editor](#)
- [Default Editor](#)

Customizing Editor

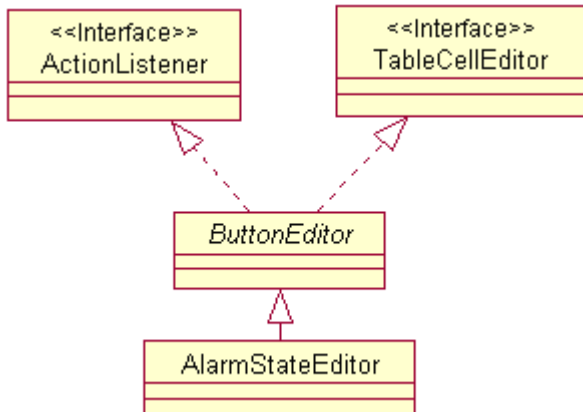
Customizing Editor

Developers can extend Editor to edit the specified type value. They need:

- Coding the new Editor class
- Register the Editor class into TPropertySheet

Now we use AlarmStateEditor as the example to show you how to create a new Editor.

You can implements `javax.swing.table.TableCellEditor` interface, or extends `TWaver ButtonEditor` directly:



`ButtonEditor` is an Editor which works like a `JButton`. That is, when you click the cell to try to edit value, it will press down and release up. When it is clicked, the edit action will perform.

When we design the Editor, for a simple type like numeric or text values, we can edit it in the cell with a `JTextField`, `JComboBox`, and `JCheckBox` etc. For a complicated type such as `Color`, `AlarmState`, we normally need a dialog to display the option values. This dialog works as a delegater, it display all options, receive user inputting, check all input values and return the inputting values to the Editor.

In this example, we display a dialog to receive user inputting:

```

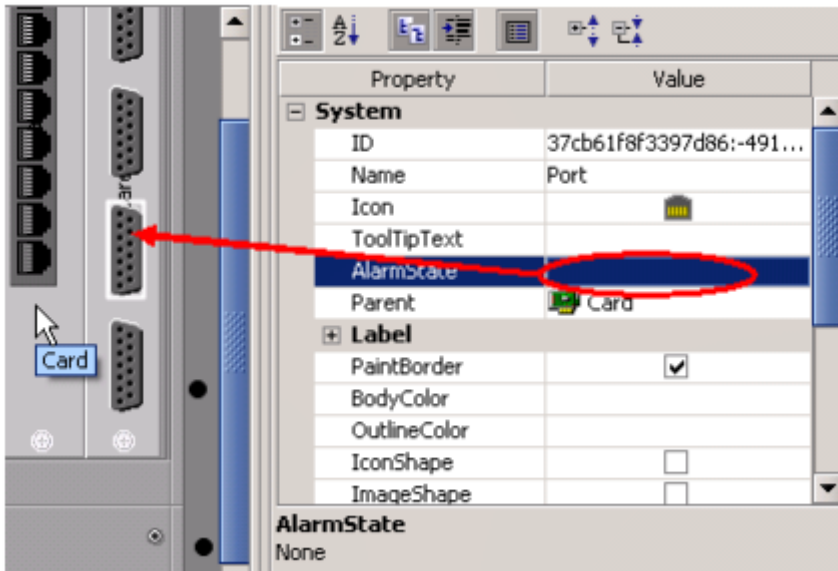
public class AlarmStateEditor extends ButtonEditor {
    JDialog dialog;
    \
    public void editPerformed() {
        Window window = (Window)SwingUtils.getWindowForComponent(table);
        if (window instanceof Dialog) {
            dialog = new AlarmStateEditorUI( (Dialog) window, box, (AlarmState) value, this);
        } else if(window instanceof Frame){
            dialog = new AlarmStateEditorUI( (Frame) window, box, (AlarmState) value, this);
        }else{
            dialog = new AlarmStateEditorUI( (Frame) null, box, (AlarmState) value, this);
        }
        dialog.setVisible(true);
    }
    \
    public void cancelPerformed() {
        dialog.dispose();
    }
    \
    public void okPerformed() {
        dialog.dispose();
    }
}
    
```

The `AlarmStateEditorUI` is a `JDialog`, it display all alarm informations with a `JTable`. User can change all values in table. Here omit the code.

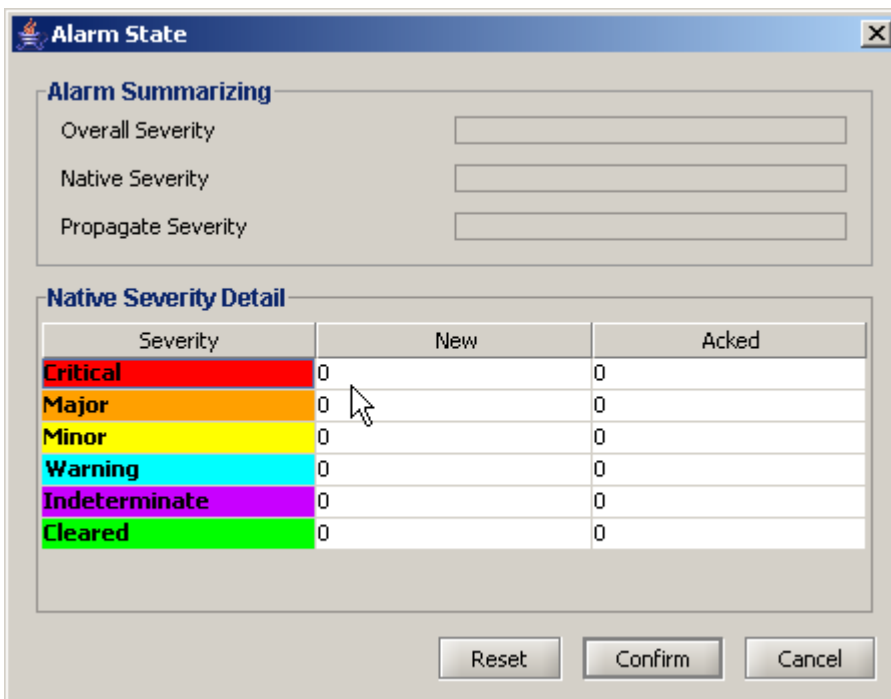
Then we register it with following code:

```
TPropertySheet sheet=new TPropertySheet();
sheet.getCellEditorManager().registerEditor(AlarmState.class, AlarmStateEditor.class);
```

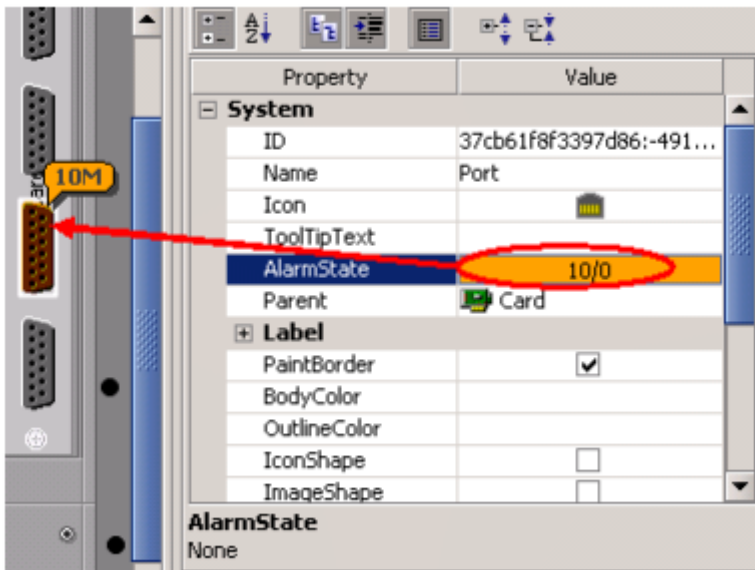
It will looks like this when you run it:



No Alarm



Show the alarm dialog to edit



Element with new alarm

Now, the new alarm value has been updated into the Element instance.

Managing Editor

Managing Editor

All TPropertySheet editors are managed by a CellEditorManager instance. CellEditorManager is the editor manager and container. You can register an editor into the manager, or look up the registered editor. You can use following methods defined in class TUIManager to register default editors:

```
public static void registerTableCellEditor(Class valueClass, String editorClass){...}
public static void registerTableCellEditor (Class valueClass, Class editorClass) {...}
```

The following codes show you how to register editor in TWaver:

```
registerTableCellEditor(Byte.class, ByteEditor.class);
registerTableCellEditor(byte.class, ByteEditor.class);
registerTableCellEditor(Short.class, ShortEditor.class);
registerTableCellEditor(short.class, ShortEditor.class);
registerTableCellEditor(Integer.class, IntegerEditor.class);
registerTableCellEditor(int.class, IntegerEditor.class);
registerTableCellEditor(Long.class, LongEditor.class);
registerTableCellEditor(long.class, LongEditor.class);
registerTableCellEditor(Float.class, FloatEditor.class);
registerTableCellEditor(float.class, FloatEditor.class);
registerTableCellEditor(Double.class, DoubleEditor.class);
registerTableCellEditor(double.class, DoubleEditor.class);
registerTableCellEditor(BigInteger.class, BigIntegerEditor.class);
registerTableCellEditor(BigDecimal.class, BigDecimalEditor.class);
registerTableCellEditor(Boolean.class, BooleanEditor.class);
registerTableCellEditor(boolean.class, BooleanEditor.class);
registerTableCellEditor(Color.class, ColorEditor.class);
registerTableCellEditor(String.class, StringEditor.class);
registerTableCellEditor(AlarmState.class, AlarmStateEditor.class);
registerTableCellEditor(Point.class, PointEditor.class);
registerTableCellEditor(Font.class, FontEditor.class);
registerTableCellEditor(java.util.Date.class, SpinnerDateEditor.class);
registerTableCellEditor(java.sql.Date.class, SpinnerDateEditor.class);
registerTableCellEditor(Timestamp.class, SpinnerDateEditor.class);
registerTableCellEditor(Direction.class, DirectionEditor.class);
registerTableCellEditor(OrthogonalLinkDirectionType.class,
EnumTypeEditor.class.getName() +
"@ " +
TWaverConst.ENUM_ORTHOGONAL +
"|false");
```

With these codes, TPropertySheet can edit most familiar types.

Default Editor

Default Editor

TWaver provides the default implementations for most Java basic types, and register these editors into `CellEditorManager`. All these editors are defined in package `"twaver.table.editor"`. See `JavaDoc` for more details.

TPropertySheet Cell Renderer

TPropertySheet derived from Javax.Swing.JTable. In the same way, it use cell Renderer to render each property cell.

The basic Renderer principle is: when TPropertySheet render each cell, it will try to get the corresponding Renderer of the property class type firstly. Then the Renderer will responsible for the cell render. For more detail about JTable and CellRenderer, please reference Java Swing JTable package.

In TPropertySheet component, each property is different type. Each different type can be rendered by different Renderer. For example, you maybe want render Boolean type with a check box component; render a Color with a colored lable, etc. So, we can define many renderers to render most class type.

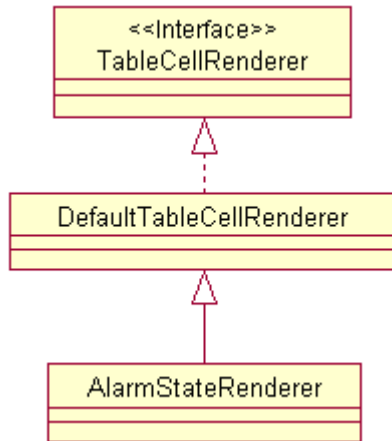
- [Customizing PropertySheet Cell Renderer](#)
- [Managing PropertySheet Cell Renderers](#)
- [Predefined Renderers](#)

Customizing PropertySheet Cell Renderer

You can develop your own renderers to render some specified type. You need:

- Code the new render class
- Register this renderer into the TableCellRendererManager

Now we use AlarmStateRenderer as the example to show you how to create a new Renderer. You can implement javax.swing.table.TableCellRenderer interface, or extends javax.swing.table.DefaultTableCellRenderer directly:



For AlarmStateRenderer, we will get the AlarmState of Element instance, get the highest alarm severity and its color, then use the color to fill the cell area. If no alarm exists, then fill with white color. The source file is:

```

public class AlarmStateRenderer extends DefaultTableCellRenderer {
    public void setValue(Object value) {
        AlarmState state = (AlarmState) value;
        setText("");
        Color color;
        AlarmSeverity severity = state.getHighestAlarmSeverity();
        if (state != null && severity != null) {
            color = severity.getColor();
        } else {
            color = Color.white;
        }
        this.setForeground(color);
        this.setBackground(color);
    }
}
    
```

Then, we register this renderer by following codes:

```

TPropertySheet sheet=new TPropertySheet(new TPropertySheetModel());
sheet.getCellRendererManager ().registerRenderer(AlarmState.class,
AlarmStateRenderer.class);
    
```

Managing PropertySheet Cell Renderers

All TPropertySheet renderers are managed by a CellRendererManager instance. CellRendererManager is a renderer manager and container. You can register a renderer into the manager, or look up the registered render. You can use following methods in class TUIManager to register default renderers:

```
public static void registerTableCellRenderer(Class valueClass, String rendererClass){...}
public static void registerTableCellRenderer(Class valueClass, Class
```

The following codes show you how to register renderers in TWaver:

```
// register default cell renderer
registerTableCellRenderer(Number.class, NumberRenderer.class);
registerTableCellRenderer(Byte.class, NumberRenderer.class);
registerTableCellRenderer(byte.class, NumberRenderer.class);
registerTableCellRenderer(Short.class, NumberRenderer.class);
registerTableCellRenderer(short.class, NumberRenderer.class);
registerTableCellRenderer(Integer.class, NumberRenderer.class);
registerTableCellRenderer(int.class, NumberRenderer.class);
registerTableCellRenderer(Long.class, NumberRenderer.class);
registerTableCellRenderer(long.class, NumberRenderer.class);
registerTableCellRenderer(Float.class, NumberRenderer.class);
registerTableCellRenderer(float.class, NumberRenderer.class);
registerTableCellRenderer(Double.class, NumberRenderer.class);
registerTableCellRenderer(double.class, NumberRenderer.class);
registerTableCellRenderer(BigInteger.class, NumberRenderer.class);
registerTableCellRenderer(BigDecimal.class, NumberRenderer.class);
registerTableCellRenderer(Boolean.class, BooleanRenderer.class);
registerTableCellRenderer(boolean.class, BooleanRenderer.class);
registerTableCellRenderer(java.util.Date.class, DateRenderer.class);
registerTableCellRenderer(java.sql.Date.class, DateRenderer.class);
registerTableCellRenderer(Timestamp.class, DateRenderer.class);
registerTableCellRenderer(Icon.class, IconRenderer.class);
registerTableCellRenderer(ImageIcon.class, IconRenderer.class);
registerTableCellRenderer(Point.class, PointRenderer.class);
registerTableCellRenderer(Color.class, ColorRenderer.class);
registerTableCellRenderer(AlarmState.class, AlarmStateRenderer.class);
registerTableCellRenderer(Font.class, FontRenderer.class);
registerTableCellRenderer(Category.class, CategoryRenderer.class);
registerTableCellRenderer(Direction.class, DirectionRenderer.class);
registerTableCellRenderer(AlarmSeverity.class, AlarmSeverityRenderer.class);
registerTableCellRenderer(AlarmTrendIndication.class, AlarmTrendIndicationRenderer.class);
registerTableCellRenderer(String.class, StringRenderer.class);
registerTableCellRenderer(Object.class, StringRenderer.class);
```

With these codes, TPropertySheet can render most familiar types.

Predefined Renderers

TWaver provides the default implementations for most Java basic types, and register these renderers into `CellRendererManager`. All of these renderers are defined in package `"twaver.table.renderer"`. See JavaDoc for more details.

TWaver and JavaBeans

The JavaBeans component API extends the Java platform's Write Once, Run Anywhere™ capability to reusable component development. In fact, the JavaBeans architecture takes interoperability a major step forward - your code runs on every OS and also within any application environment. A developer who creates components based on the JavaBeans architecture (beans) secures a future in the emerging network software market without losing customers that use proprietary platforms, because beans interoperate with ActiveX, OpenDoc, and LiveConnect.

TWaver defines all Business Classes follow the JavaBeans specification. That is, all Elements like Node, Link, SubNetwork, and Group are all Java bean. You can:

- Find all properties by beans tool
- The properties can be changed in the design time
- Elements can communicate each other by events
- Elements can be persisted into XML file

Using Property Sheet Pane

Property sheet pane is a JPanel, it works with TPropertySheet component. The property sheet pane is designed as a parent container for property sheet component. It contains a predefined toolbar on the top, a scroll pane for the property sheet in the center, and a property description text area in the bottom. The toolbar contains a set of buttons used to operate property sheet, such as sort, expand or collapse categories, display or hide description text, etc.

You can use this pane in your application for a property sheet, or just use the property sheet directly without this panel. You can access the toolbar of this property sheet panel, change the predefined buttons or add more buttons on it. Use a property sheet to create property sheet pane:

```
public TPropertySheetPane(final TPropertySheet sheet)
public TPropertySheetPane(final TPropertySheet sheet, boolean simpleStyle)
```

Simple style:

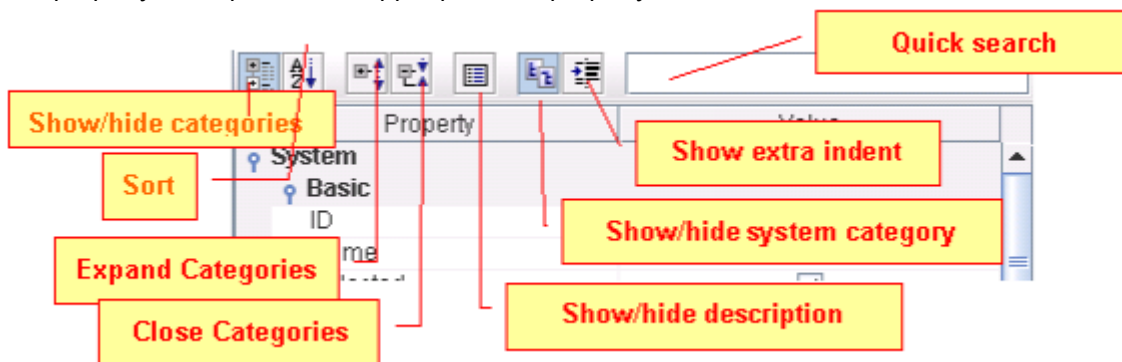


Full style:



```
TDataBox box=new TDataBox();
TPropertySheet sheet=new TPropertySheet();
TPropertySheetPane pane=new TPropertySheetPane(sheet,false);
```

The property sheet pane is a wrapper panel for property sheet.



Using Chart Component

Chart Component is similar with other TWaver components, it contacts a TDataBox instance to get business data and rendering in the Chart. TWaver Chart also provides interacting ability, you can zoom in, zoom out, pan and click the Chart by mouse.

- [Using BarChart](#)
- [Using Bubble Chart](#)
- [Using DialChart](#)
- [Using LineChart](#)
- [Using PercentChart](#)
- [Using PieChart](#)
- [Using Radar Chart](#)

Using BarChart

BarChart is one of the Chart Components. It presents the data in form of bar graphics. It has several different types as follows. The type can be set by the method 'setBarType'.

BAR_TYPE_DEFAULT

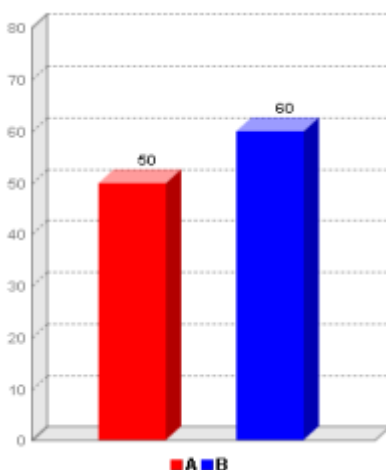
In the condition of BAR_TYPE_DEFAULT type, BarChart reads the ChartValue of the element contained by the DataBox, and then presents the data coordinately in the graphics panel.

Creating a BarChart with BAR_TYPE_DEFAULT type:

```
JFrame frame = new JFrame();
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

TDataBox box = new TDataBox();
BarChart barChart = new BarChart(box);
//Set y ticks visible. Default value is false.
barChart.setYScaleTextVisible(true);
//set y minimum ticks visible. Default value is false
barChart.setYScaleMinTextVisible(true);
//set upper limit of y ticks
barChart.setUpperLimit(80);
//set space between ticks
barChart.setYScaleValueGap(10);
//add a node
Element a = new Node("A");
a.setName("A");
//set chart color
a.putChartColor(Color.RED);
//set chart value
a.putChartValue(50);
box.addElement(a);
Element b = new Node("B");
b.setName("B");
b.putChartColor(Color.BLUE);
b.putChartValue(60);
box.addElement(b);

frame.setContentPane(barChart);
frame.setSize(500, 400);
TWaverUtil.centerWindow(frame);
frame.setVisible(true);
```





BAR_TYPE_DEFAULT

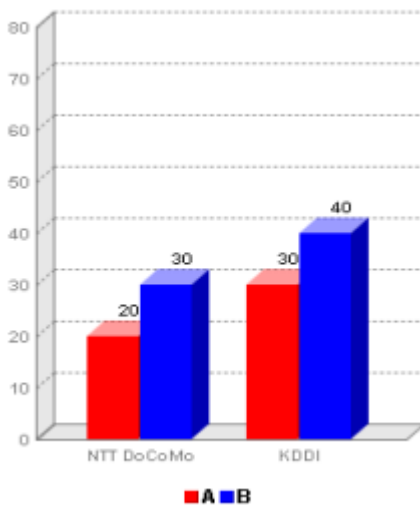
Default bar style. For this default style, Chart Value will be used as the bar value. For other bar styles, Chart List Values will be used. Please check out details in [What's the differences between Chart Value and List Values?](#)

BAR_TYPE_GROUP

In the condition of BAR_TYPE_GROUP type, BarChart reads the ChartValue list of the every element contained by DataBox, and then presents the data list in the graphics panel in form of group. Notice that, you must add the scale text along the X-Axis by the method 'addXScaleText'.

Setting the type as BAR_TYPE_GROUP:

```
//add x ticks
barChart.addXScaleText("NTT DoCoMo");
barChart.addXScaleText("KDDI");
//add chart values
a.addChartValue(20);
a.addChartValue(32);
b.addChartValue(28);
b.addChartValue(40);
//set chart styles
barChart.setBarType(TWaverConst.BAR_TYPE_GROUP);
```

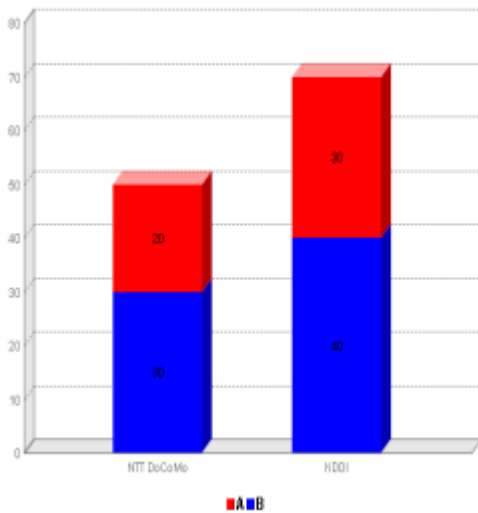


BAR_TYPE_STACK

In the condition of BAR_TYPE_STACK type, BarChart reads the ChartValue list of the every element contained by DataBox, and then presents the data list in the graphics panel in form of stack.

Setting the type as BAR_TYPE_STACK:

```
barChart.setBarType(TWaverConst.BAR_TYPE_STACK);
```

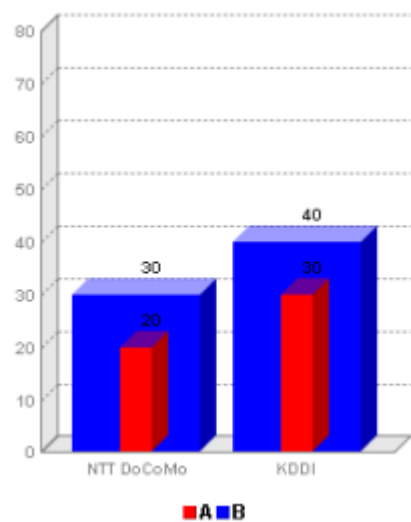


BAR_TYPE_LAYER

In the condition of BAR_TYPE_LAYER type, BarChart reads the ChartValue list of the every element contained by DataBox, and then presents the data list in the graphics panel in form of layer.

Setting the type as BAR_TYPE_LAYER:

```
barChart.setBarType(TWaverConst.BAR_TYPE_LAYER);
```



Using Bubble Chart

A Bubble chart is a type of a scatter chart in which the data points are painted with bubbles. A Bubble chart can be used to draw your data has three data series, each of which contains a set of values. In the Bubble chart, the size of the bubbles is determined by the values in the data points. X and Y can be the categories or something.

TWaver Bubble chart has 4 styles:

- TWaverConst.INFLEXION_STYLE_CIRCLE
- TWaverConst.INFLEXION_STYLE_DIAMOND
- TWaverConst.INFLEXION_STYLE_RECTANGLE
- TWaverConst.INFLEXION_STYLE_TRIANGLE

Bubble Properties

Bubble properties defined by class `twaver.chart.Bubble`, include following properties:

- `x` : value on x
- `y` : value on y
- `value` : value of bubble
- `text` : bubble name

In this example, `x,y,value,text` will be Number of products,Sales,Market share % and company name

X:Number of products	Y:Sales	Value:Market share %
14	\$23,000	23
...

Code:

```
BubbleChart chart = new BubbleChart() {
//Customize ToolTip
protected String getToolTipText(Element element, Bubble bubble, int index) {
    if (!this.isEnabledToolTipText()) {
        return null;
    }
    return bubble.getText();
}
//Format y ticks
protected String getFormattedYScaleText(double value) {
    return NumberFormat.getCurrencyInstance().format(value);
}
//Customize bubble size. By default the radius of the bubble will be the bubble value.
//So most of the time you need to will need to transform into a size fit your screen.
protected double getShapeSize(Element element, Bubble bubble, int index) {
    double chartWidth=this.getBackgroundBounds().getWidth();
    double chartHeight=this.getBackgroundBounds().getHeight();
    double chartMinSize=chartWidth;
    if(chartWidth>chartHeight){
        chartMinSize=chartHeight;
    }
    double result=bubble.getValue()/30*chartMinSize/10;

    return result;
};
};
for (int i = 1; i < 10; i++) {
```

```

double value=10+TWaverUtil.getRandomInt(20);
Bubble bubble = new Bubble("Company_" + i, 8+TWaverUtil.getRandomInt(20),
    7000+ TWaverUtil.getRandomInt(65000),value);
Item item = new Item();
item.addChartBubble(bubble);
item.setColor(Color.blue);
chart.addItem(item);
}
chart.setTitle("Industry market share study");

//Set names for x and y
chart.setXAxisText("Number of products");
chart.setYAxisText("Sales");
//Set grid line visible.
chart.setXScaleLineVisible(true);
chart.setYScaleLineVisible(true);

//set the ticks space for x and y.
chart.setXScaleValueGap(5);
chart.setYScaleValueGap(20000);

//Set minimum limit of y
chart.setLowerLimit(0);
chart.setXScaleMinValue(5);
chart.setXScaleMaxValue(30);

chart.setYScaleMinValue(0);
chart.setYScaleMaxValue(80000);

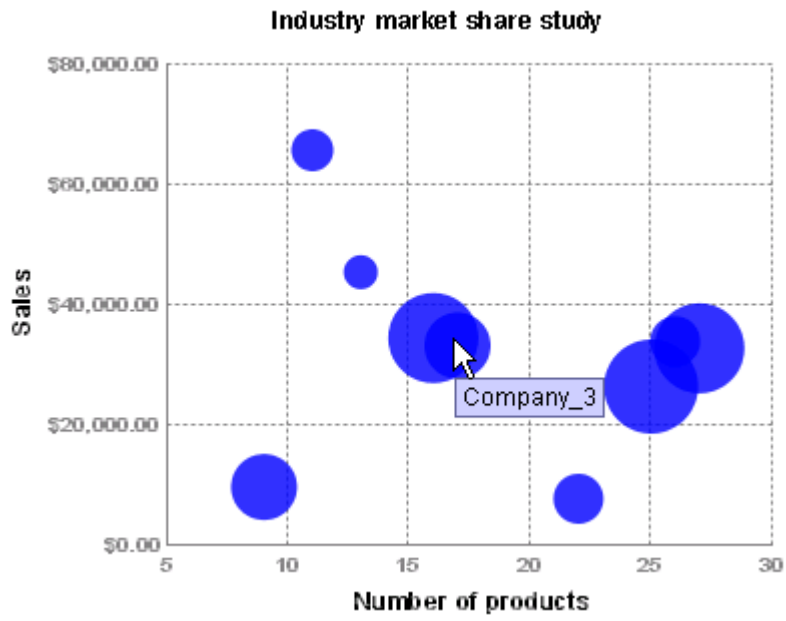
chart.setValueTextVisible(false);
chart.getLegendPane().setVisible(false);
chart.setBubbleGradient(false);

//Set ticks on y visible.
chart.setYScaleTextVisible(true);

//Set no shadow style. If a non-zero value used, then it will display a shadow in the given offset.
chart.setShadowOffset(0);

JFrame frame = new JFrame();
frame.setTitle("bubble chart");
frame.getContentPane().add(chart);
frame.setSize(900, 700);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
TWaverUtil.centerWindow(frame);
frame.setVisible(true);

```



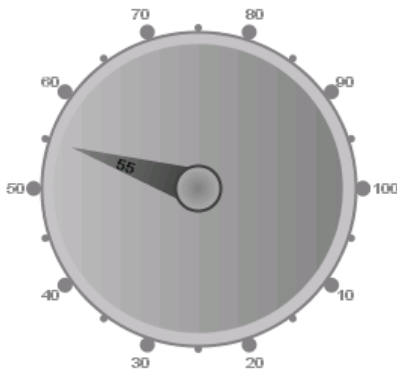
Using DialChart

A gauge chart (dial chart) represents one or more values as needles on a circular or semicircular surface. One dial chart can represents more values. Every value comes from an Element object. You can use `Element.putChartValue(double value)` to set the chart value. The dial needle style can be set by `Element.putChartDialHandStyle(int chartDialHandStyle)`.

Here is an example to use dial chart:

```
TDataBox box=new TDataBox();
DialChart dialChart = new DialChart(box);
//set major scale count
dialChart.setScaleMajorCount(10);
//set minor scale count
dialChart.setScaleMinorCount(1);
Node element = new Node();
element.setName("Element");
//set value
element.putChartValue(55d);
box.addElement(element);

JFrame frame = new JFrame();
frame.getContentPane().add(dialChart, BorderLayout.CENTER);
frame.setSize(400, 400);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
TWaverUtil.centerWindow(frame);
frame.setVisible(true);
```



Each dial has background, guage, needle, scale, values, legend. Many parameters provided for each part. See below example to know how to use them:

```
TDataBox box=new TDataBox();

DialChart dialChart = new DialChart(box);

//set scale major count
dialChart.setScaleMajorCount(10);
//set scale minor count
dialChart.setScaleMinorCount(1);
//enable paint scale inside
dialChart.setScaleInside(true);
//set dial style
//dialChart.setDialType(TWaverConst.DIAL_TYPE_ARC);
//invisible value text
dialChart.setValueTextVisible(false);
```



```
//show text in center.
dialChart.setValueTextCenter(true);
//set scale text font
dialChart.setScaleTextFont(twaver.TUIManager.getDefaultBoldFont().deriveFont(16f));
//set value text color
dialChart.setValueTextColor(Color.RED);
//set sector range
dialChart.setArcRange(180);
//set scale style
dialChart.setScaleStyle(TWaverConst.DIAL_SCALE_STYLE_LINE);
//set minimum scale text visible
//dialChart.setScaleMinTextVisible(true);
//set dial outside border color
dialChart.setRingColor(null);
//set guage fill color
dialChart.setRingFillColor(new Color(45,57,57));

//set scale color
dialChart.setScaleColor(new Color(160,183,178));
//set scale text color
dialChart.setScaleTextColor(new Color(160,183,178));
//dialChart.setRingGradientFactory(TWaverConst.GRADIENT_RADIAL_C);

//set center ball fill gradient color
dialChart.setBallGradientColor(Color.GRAY);
//set center ball color
dialChart.setBallColor(Color.black);
//set center ball size
dialChart.setBallSize(40);
//set center ball border color
dialChart.setBallBorderColor(Color.white);
//set center ball gradient type
dialChart.setBallGradientFactory(TWaverConst.GRADIENT_LINE_NW);

//set dial border inner color
dialChart.setRingBorderColor(new Color(45,57,57));
//disable gradient fill
dialChart.setRingGradient(false);

Node element = new Node();
element.setName("Element");
//set needle length
element.putChartDialHandLength(0.85);
//set needle type to line
element.putChartDialHandStyle(TWaverConst.DIAL_HAND_STYLE_LINE);
//set needle stroke
element.putChartStroke(TWaverConst.STROKE_SOLID_6);
//set needle color
element.putChartColor(new Color(168,166,255));
//set value
element.putChartValue(55d);
box.addElement(element);

JFrame frame = new JFrame();
frame.getContentPane().add(dialChart,BorderLayout.CENTER);
frame.setSize(400, 400);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
TWaverUtil.centerWindow(frame);
frame.setVisible(true);
```



Using LineChart

LineChart is one of the Chart Components. It presents the data in form of line graphics. LineChart reads the ChartVale list of every element contained by DataBox, and then presents the data list in the graphics panel in form of line.

Creating a simple LineChart:

```
TDataBox box = new TDataBox();

LineChart lineChart = new LineChart(box);

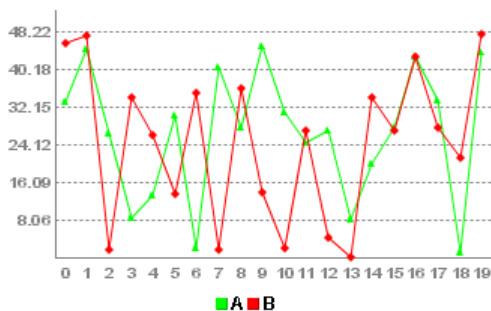
lineChart.setYAxisVisible(true);
lineChart.setYScaleTextVisible(true);
lineChart.setXAxisVisible(true);
lineChart.setXScaleTextVisible(true);
//Display markers for each data value.
lineChart.setInflexionVisible(true);

Element a = new Node();
a.setName("A");
a.putChartColor(Color.GREEN);
//set the value marker style.
a.putChartInflexionStyle(TWaverConst.INFLEXION_STYLE_TRIANGLE);
box.addElement(a);

Element b = new Node();
b.setName("B");
b.putChartColor(Color.RED);
b.putChartInflexionStyle(TWaverConst.INFLEXION_STYLE_DIAMOND);
box.addElement(b);

for(int i=0;i<20;i++){
    lineChart.addXScaleText(""+i);
    a.addChartValue(Math.random()*50);
    b.addChartValue(Math.random()*50);
}

JFrame frame = new JFrame();
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setContentPane(lineChart);
frame.setSize(500, 400);
TWaverUtil.centerWindow(frame);
frame.setVisible(true);
```

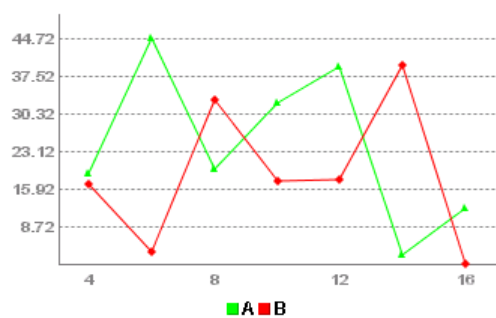


LineChart defines several line types, the type can be set by the method 'setLineType', as follows.

Type	Picture
TWaverConst.LINE_TYPE_POLE	
TWaverConst.LINE_TYPE_AREA	
TWaverConst.LINE_TYPE_DEFAULT	

The `startIndex`, `endIndex` and `spanCount` can be changed to set the display bound of the line chart, as follows.

```
//Sets the start index of the lineChart
lineChart.setStartIndex(4);
//Sets the end index of the lineChart
lineChart.setEndIndex(17);
//Sets the span count of the scale text along x-axis
lineChart.setXScaleTextSpanCount(4);
//Sets the span count of the value along x-axis
lineChart.setValueSpanCount(2);
```



Using PercentChart

PercentChart is one of the Chart Components. It presents the data in form of percent graphics. PercentChart reads the ChartValue list of the every element contained by DataBox, and then presents the data list in form of percent graphics.

Creating a simple PercentChart with three different styles:

- TWaverConst.PERCENT_STYLE_PLANE
- TWaverConst.PERCENT_STYLE_SEGMENT
- TWaverConst.PERCENT_STYLE_SOLID

```
TDataBox box = new TDataBox();
PercentChart percentChart = new PercentChart(box);

Element cpuElement = new Node();
cpuElement.setName("CPU");
cpuElement.putChartValue(50);
//plane style
cpuElement.putChartPercentStyle(TWaverConst.PERCENT_STYLE_PLANE);

Element memoryElement = new Node();
memoryElement.setName("MEMORY");
memoryElement.putChartColor(Color.ORANGE);
memoryElement.putChartValue(80);
//segment style
memoryElement.putChartPercentStyle(TWaverConst.PERCENT_STYLE_SEGMENT);

Element netElement=new Node();
netElement.setName("NETWORK");
netElement.putChartColor(Color.GREEN);
netElement.putChartValue(30);
//default style
netElement.putChartPercentStyle(TWaverConst.PERCENT_STYLE_SOLID);

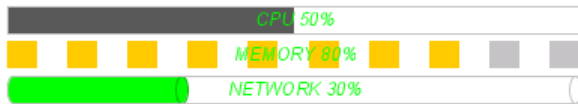
box.addElement(cpuElement);
box.addElement(memoryElement);
box.addElement(netElement);

JFrame frame = new JFrame();
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setContentPane(percentChart);
frame.setSize(400, 100);
TWaverUtil.centerWindow(frame);
frame.setVisible(true);
```



The label color, font and position of the PercentChart can be set as follows:

```
percentChart.setPercentLabelCenter(true);
percentChart.setPercentLabelFont(TUIManager.getDefaultItalicFont());
percentChart.setPercentLabelColor(Color.GREEN);
```



The gradient background can be set as follows:

```
percentChart.setBackgroundVisible(true);
percentChart.setGradient(true);
percentChart.setBackgroundOutlineColor(Color.WHITE);
percentChart.setBackgroundFillColor(Color.PINK);
percentChart.setBackgroundGradientFactory(TWaverConst.GRADIENT_EXTEND_E);
```



Using PieChart

PieChart is one of the Chart Components. It presents the ChartValue list in form of pie graphics. PieChart reads the ChartValue list, and then presents the data list in form of pie graphics.

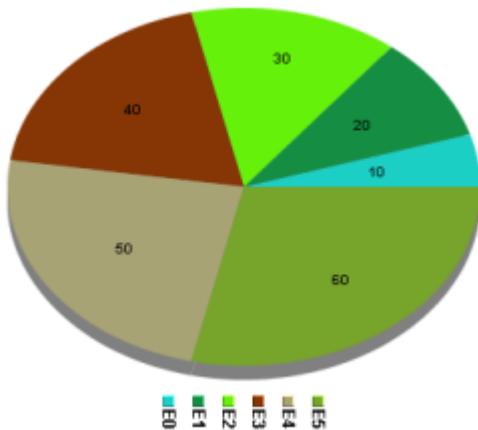
Creating a simple PieChart:

```

TDataBox box = new TDataBox();
PieChart pieChart = new PieChart(box);
pieChart.setLegendOrientation(TWaverConst.LABEL_ORIENTATION_RIGHT);
Random random = new Random();
for (int i = 0; i < 6; i++) {
    Element element = new Node();
    element.setName("E"+i);
    element.putChartValue(10 * (i + 1));
    element.putChartColor(new Color(random.nextInt(255),random.nextInt(255), random.nextInt(255)));
    box.addElement(element);
}

JFrame frame = new JFrame();
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setContentPane(pieChart);
frame.setSize(500, 400);
TWaverUtil.centerWindow(frame);
frame.setVisible(true);

```

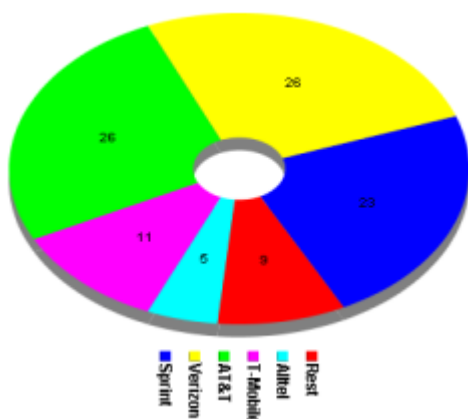


PieChart can be changed to cirque chart as follows:

```

this.setHollow(true);
this.setHollowPercent(0.2);

```

Using Radar Chart

A radar chart, also known as a spider chart or star chart or kiviati diagram, is a two-dimensional chart of three or more quantitative variables represented on axes starting from the same point. The relative position and angle of the axes is uninformative.

Radar charts are usually used to compare performance of different entities on a same set of axes. Radar charts are useful when you want to look at several different factors all related to one item. Radar charts have multiple axes along which data can be plotted.

Example:

```
public static void main(String[] args) {
    TDataBox box = new TDataBox();
    RadarChart chart = new RadarChart(box);
    chart.setTitle("Charm chart");

    //set data
    addElement(box, "Sam", new Color(0,0,255,100), TWaverConst.STROKE_SOLID_1);
    addElement(box, "Kelly", new Color(255,0,0,200), TWaverConst.STROKE_SOLID_1);

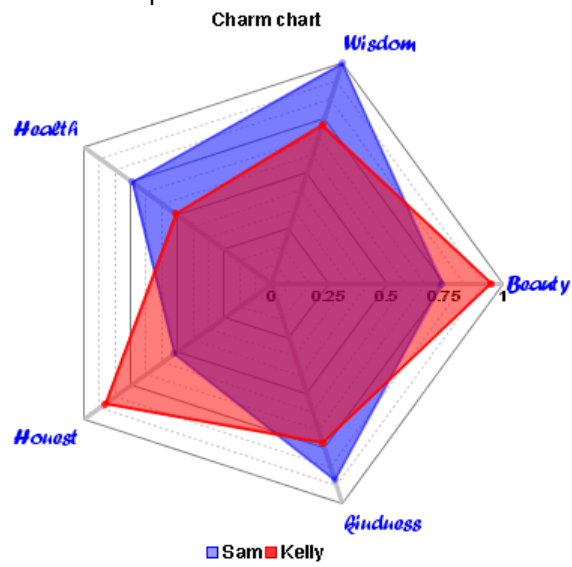
    //set scale
    chart.setScaleMajorCount(4);
    //fill shape
    chart.setShapeFill(true);
    //fill without gradient
    chart.setShapeFillGradient(false);
    //set scale text font
    chart.setScaleMajorTextColor(Color.BLACK);
    //set axis text font
    chart.setAxisTextFont(new Font("Forte", Font.PLAIN, 14));
    chart.addAxisText("Beauty");
    chart.addAxisText("kindness");
    chart.addAxisText("Honest");
    chart.addAxisText("Health");
    chart.addAxisText("Wisdom");

    JFrame frame = new JFrame();
    frame.getContentPane().add(chart, BorderLayout.CENTER);
    frame.setSize(700, 400);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    TWaverUtil.centerWindow(frame);
    frame.setVisible(true);
}

private static void addElement(TDataBox box, String name, Color color, String stroke) {
    Element element = new Node();
    element.setName(name);
    //set chart color
    element.putChartColor(color);
    //set flexion style
    element.putChartInflexionStyle(TWaverConst.INFLEXION_STYLE_CIRCLE);
    //set chart stroke
    element.putChartStroke(stroke);
    //give some random numbers.
    for (int i = 0; i < 5; i++) {
        double value = 0.5 + TWaverUtil.getRandomDouble() * 0.5;
        element.addChartValue(value);
    }
    box.addElement(element);
}
```

```
}
```

Run this example:



Using Alarm

This chapter introduces you how to use alarm in your application. Alarms are related information of element data. It managed by alarm model of DataBox container.

- [Alarm Object Defination](#)
- [Extending Alarm Properties](#)
- [Displaying Alarm with AlarmUI](#)
- [Using AlarmType](#)
- [Using AlarmTrendIndication](#)
- [Using AlarmState](#)
- [Using AlarmSeverity](#)
- [Using Alarm Model](#)
- [Using AlarmStateStatistics](#)
- [Creating Alarm by XML](#)
- [Creating Alarm by API](#)
- [Customizing Alarm Renderer Color](#)

Alarm Object Defination

TWaver defines alarm class in `twaver.Alarm`, that you can directly use it to create alarm object and put it into the alarm model of a `DataBox`. The alarm object will be managed by alarm model, and effect the graphical representation of related element data.

Alarms are used to represent individual alarm conditions occurring on element object. Alarm class defines the following properties.

Property Name	Value Type	Description
alarmID	Object	Unique system identifier of the alarm object.
ackTime	<code>java.util.Date</code>	Time of the last modification of the acknowledged state.
ackUserID	String	Identifier of the author of the last modification of the acknowledged state.
additionalText	String	Additional information about the alarm.
acked	boolean	Whether or not the alarm is acknowledged.
lastChangedTime	<code>java.util.Date</code>	Date of the last modification of alarm attribute values.
cleared	boolean	Whether or not the alarm is cleared.
clearedTime	<code>java.util.Date</code>	Date when the perceived severity of the alarm was changed to Cleared.
raisedTime	<code>java.util.Date</code>	Date when the alarm was raised.
alarmType	<code>twaver.AlarmType</code>	The type of alarm.
clearUserID	String	Identifier of the author of the last request to clear the alarm.
comments	<code>java.util.Collection</code>	All comments of the alarm.
correlatedAlarmIDs	<code>java.util.Collection</code>	The correlated alarm identifiers. This attribute identifies a set of alarms to which this alarm is considered to be correlated.
elementID	Object	The element object instance identifier on which the alarm occurred.
alarmSeverity	<code>twaver.AlarmSeverity</code>	The severity of the alarm. It indicates the relative level of urgency for operator attention.
probableCause	<code>twaver.AlarmProbableCause</code>	The probable cause of the alarm.

		<p><code>twaver.AlarmProbableCause</code> contains definitions for probable causes that are used in reported alarms.</p> <p>This definition correspond to the OSS/J constants defined in <code>javax.oss.fm.monitor.ProbableCause</code>.</p>
<code>proposedRepairAction</code>	<code>String</code>	The proposed repair actions.
<code>specificProblem</code>	<code>String</code>	The specific problem. It provides more information of the alarm than the <code>probableCause</code> .
<code>trendIndication</code>	<code>twaver.AlarmTrendIndication</code>	The trend indication. This attribute indicates whether the observed condition is getting better, worse, or is unchanged
<code>clientProperties</code>	<code>java.util.Map</code>	All attached client properties.

Extending Alarm Properties

Alarm class definition is based on the OSS/J Quality of the Service API and the ITU-T Recommendation X.733. But you may want add more additional informations on the alarm object. Fortunately, Alarm class provides an internal hashtable to carry more client properties. Each client property should have a unique key value. This mechanism is very like the client property of twaver.Element.

```
Alarm alarm=new Alarm();  
//put more client properties on the alarm.  
alarm.putClientProperty("myProperty1", "abc");  
alarm.putClientProperty("myProperty2", "xyz");  
//retrieve all client properties on the alarm.  
Map props=alarm.getClientProperties();
```

Displaying Alarm with AlarmUI

TWaver provides a well designed window to display alarm detail information. Once you have an alarm instance, you can create AlarmUI with the alarm instance, put it into a frame or dialog, and show it. You can find this example in demo.alarm.overview.AlarmUI in TWaver Example application.


AlarmUI also give you two static factory methods for your convenience. It is easier to create AlarmUI with JFrame or JDialog.

Method	Return	Description
AlarmUI.createFrameAlarmUI	JFrame	Create a JFrame instance which contains an AlarmUI panel.
AlarmUI.createDialogAlarmUI	JDialog	Create a JDialog instance which contains an AlarmUI panel.

Following code show you how to create an AlarmUI with an alarm instance.

```
public static void main(String[] args) {
    Alarm alarm = new Alarm("ALM0001");
    alarm.setAckTime(new Date());
    alarm.setAckUserID("Ack user name");
    alarm.setAdditionalText("Additional text");
    alarm.setAcked(true);
    alarm.setLastChangedTime(new Date());
    alarm.setCleared(true);
    alarm.setClearedTime(new Date());
    alarm.setRaisedTime(new Date());
    alarm.setAlarmType(AlarmType.EQUIPMENT_ALARM);
    alarm.setClearUserID("Clear user name");
    alarm.addComment("first comment");
    alarm.addComment("second comment");
    alarm.addCorrelatedAlarmID("ALM0008");
    alarm.addCorrelatedAlarmID("ALM0009");
    alarm.setElementID("node02");
    alarm.setAlarmSeverity(AlarmSeverity.MAJOR);
    alarm.setProbableCause(AlarmProbableCause.LOW_CABLE_PRESSURE);
    alarm.setProposedRepairAction("Junk it now");
    alarm.setSpecificProblem("no specific problem");
    alarm.setTrendIndication(AlarmTrendIndication.MORE_SEVERE);
    alarm.putClientProperty("key1", "This is key1 value.");
    alarm.putClientProperty("key2", new Date());
    alarm.putClientProperty("key3", "test value.");

    //create AlarmUI with JFrame mode
    AlarmUI.createFrameAlarmUI(alarm).setVisible(true);
    //or create AlarmUI with JDialog mode
    AlarmUI.createDialogAlarmUI(alarm).setVisible(true);
}
```



Alarm 'ALM0001' Properties

Alarm ID:

ALM0001

Source element ID:

node02


Alarm severity:

Major

Alarm type:

EquipmentAlarm

Trend indication:



Last change time:

11/21/05 4:44 PM

Raised time:

11/21/05 4:44 PM

Acknowledged:

☒


Acknowledge user ID:

Ack user name

Acknowledged time:

11/21/05 4:44 PM

Cleared:



Clear time:

11/21/05 4:44 PM

Clear user ID:

Clear user name

Probable cause:

Low cable pressure

Proposed repair action:

Junk it now

Specific problem:

no specific problem

Additional text:

Additional text

Comments

Correlated alarms

Client properties

Key	Value
key1	This is key1 value.
key2	Mon Nov 21 16:44:11 CST 2005
key3	test value.



From TWaver 1.3, you can find the source code of class AlarmUI in TWaver Example which released with TWaver product.

Using AlarmType

Class `twaver.AlarmType` identifies all 3G TS 32.111-2 [4] defined alarm event types used by this API. Their semantics are defined by 3GPP. Their encodings for this API are defined here. Its definition is based on the ITU-T Recommendation X.733 and OSS/J Quality of the Service API.

Here lists all alarm types for reference:

AlarmType	Alarm Name	Description
COMMUNICATIONS_ALARM	CommunicationsAlarm	Communications alarm
PROCESSING_ERROR_ALARM	ProcessingErrorAlarm	Processing error alarm
ENVIRONMENTAL_ALARM	EnvironmentalAlarm	Environmental alarm
QUALITY_OF_SERVICE_ALARM	QualityOfServiceAlarm	Quality of Service alarm
EQUIPMENT_ALARM	EquipmentAlarm	Equipment Alarm
INTEGRITY_VIOLATION	IntegrityViolation	Integrity Violation alarm
SECURITY_VIOLATION	SecurityViolation	Security Violation alarm
TIME_DOMAIN_VIOLATION	TimeDomainViolation	Time Domain Violation alarm
OPERATIONAL_VIOLATION	OperationalViolation	OperationalViolation alarm
PHYSICAL_VIOLATION	PhysicalViolation	PhysicalViolation alarm

Using AlarmTrendIndication

twaver.AlarmTrendIndication

This class contains definitions for trend indication types for observed conditions. It's possible to indicate if some observed condition is getting better, worse, or not changing. Its definition is based on the ITU-T Recommendation X.733 and OSS/J Quality of the Service API. TWaver defines following types of twaver.AlarmTrendIndication:

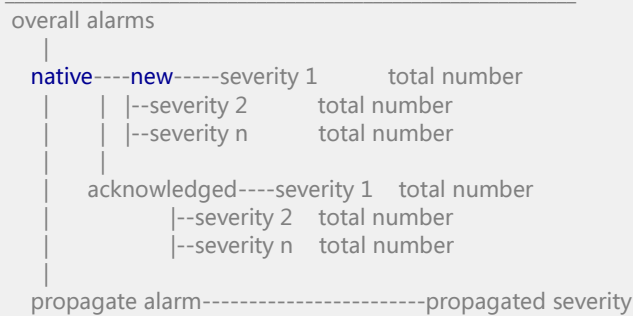
- AlarmTrendIndication.LESS_SEVERE
- AlarmTrendIndication.NO_CHANGE
- AlarmTrendIndication.MORE_SEVERE

Using AlarmState

twaver.AlarmState

This class provides a representation model for generic alarms carried by a telecom object. Each alarm severity present in the alarm system, can have a number of new alarms and a number of acknowledged alarms stored. It also introduces the concept of outstanding alarms, whose number is the sum of all alarms, whether acknowledged or not.

AlarmState data structure:



New alarms will change to acknowledged alarm via acknowledge action:

(new alarm)- acknowledge ->(acknowledged alarm)

public AlarmSeverity getHighestAcknowledgedAlarmSeverity()

Gets the highest acknowledged alarm severity

public AlarmSeverity getHighestNewAlarmSeverity()

Get the highest new alarm severity

public boolean hasLessSevereNewAlarms()

Tells whether has non-highest severe new alarms. When this is true, the alarm bubble will display "+" in the end of the alarm bubble text to indicate that this alarm state has more less severe new alarms.

public AlarmSeverity getHighestOverallAlarmSeverity()

Get the highest overall alarm severity. The highest overall severity is the highest alarm among acknowledged alarms, new alarms and propagated alarms.

public AlarmSeverity getHighestNativeAlarmSeverity()

Get the highest native alarm severity. The highest native severity is the highest severity among acknowledged alarms and new alarms.

public void increaseAcknowledgedAlarm(AlarmSeverity severity, int increment)

Increase acknowledged alarm count for given severity.

public void increaseNewAlarm(AlarmSeverity severity, int increment)

Increase new alarm count for given severity.

public void decreaseAcknowledgedAlarm(AlarmSeverity severity, int decrement)

Decrease acknowledged alarm count for given severity.

public void decreaseNewAlarm(AlarmSeverity severity, int decrement)

Decrease new alarm count for given severity.

public void acknowledgeAlarm(AlarmSeverity severity)

Acknowledges an alarm of a given severity. The alarm is moved from the "new" category to the "acknowledged" category.

public void acknowledgeAllAlarms(AlarmSeverity severity)

Acknowledges all alarms of a given severity.

```
public void acknowledgeAllAlarms()
Acknowledges all alarms.
```

```
public void addAcknowledgedAlarm(AlarmSeverity severity)
Adds a acknowledged alarm with a given severity.
```

```
public void addNewAlarm(AlarmSeverity severity)
Adds a new alarm with given severity.
```

```
public int getAcknowledgedAlarmCount()
Returns the number of acknowledged alarms, all alarm severities included.
```

```
public int getAcknowledgedAlarmCount(AlarmSeverity severity)
Returns the total acknowledged alarm count of specified severity.
```

```
public int getAlarmCount()
Returns the total number of alarms, acknowledged or otherwise, all alarm severities included.
```

```
public int getAlarmCount(AlarmSeverity severity)
Returns the total number of alarms with the specified severity, acknowledged or new.
```

```
public int getNewAlarmCount()
Returns the number of all new alarms, all alarm severities included.
```

```
public int getNewAlarmCount(AlarmSeverity severity)
Returns the total number of all new alarms of the given severity.
```

```
public void setNewAlarmCount(AlarmSeverity severity, int count)
Sets the new alarm total count for given severity.
```

Parameters:
AlarmSeverity the alarm severity constrain
int the new total number for the specified severity.

```
public void removeNewAlarm(AlarmSeverity severity)
Removes a new alarm with the given severity. This is an equivalent with
```

```
decreaseNewAlarm(severity, 1)
Parameters:
AlarmSeverity the alarm severity constraint
```

```
public void removeAllNewAlarms(AlarmSeverity severity)
Removes all new alarms of the given severity.
```

Parameters:
AlarmSeverity the alarm severity constraint

```
public void removeAllNewAlarms()
Removes all new alarms of all severities.
```

```
public void setAcknowledgedAlarmCount(AlarmSeverity severity, int count)
Sets a new total acknowledged alarm number for the given severity.
```

Parameters:
severity AlarmSeverity the alarm severity constraint
count int the new total acknowledged alarm number for the given severity.

```
public void removeAcknowledgedAlarm(AlarmSeverity severity)
Removes an acknowledged alarm on the given severity.
```

Parameters:
severity AlarmSeverity the alarm severity constraint

```
public void removeAllAcknowledgedAlarms(AlarmSeverity severity)
Removes all acknowledged alarms of the given severity.
```


Parameters:
severity AlarmSeverity severity constraint

`public void removeAllAcknowledgedAlarms()`
Removes all acknowledged alarms of all severities.

`public void clear()`
Removes all alarms, include all new, acknowledged or propagated alarms, all alarm severities.


`public void copyFrom(AlarmState alarmState)`
copy from a alarmstate

`public AlarmSeverity getPropagateSeverity()`
Gets the propagate alarm severity. Normally the propagated alarm severity is the highest severity of all children elements of the related element, if you use the default TWaver alarm propagator on the DataBox.

 Please note that AlarmState does not save any number information for the propagate alarm. Only the propagate alarm severity provided.

Returns:
AlarmSeverity the propagated alarm severity.

`public void setPropagateSeverity(AlarmSeverity propagateSeverity)`
Set the propagate alarm severity. Normally the propagated alarm severity is the highest severity of all children elements of the related element, if the default alarm propagator has been used on DataBox.

 In most cases, don't call this method to change the propagated alarm severity out of an alarm propagator(see twaver.AlarmPropagator).
Please note that AlarmState does not save any number information for the propagate alarm. Only the propagate alarm severity provided.

Parameters:
propagateSeverity AlarmSeverity the new propagate alarm severity.

`public boolean isEmpty()`
Tells whether this alarm state object has any alarms.

`public boolean isEnablePropagationFromChildren()`
Returns true if the children's alarm state can propagate to the related element of this alarm state.


`public void setEnablePropagationFromChildren(boolean enable)`
Determines whether or not the children's alarm state can propagate to the related element of this alarm state.

Using AlarmSeverity

- [Customizing AlarmSeverity](#)
- [Customizing AlarmSeverity Comparator](#)
- [Customizing Renderer Color for Different Severity Levels](#)

Customizing AlarmSeverity

twaver.AlarmSeverity defines the alarm severity for alarm. In TWaver system, each alarm object should combine an alarm severity to indicate how severe the alarm is. TWaver predefines several alarm severities in this class for use. This defines correspond with most common use, but developers can remove the predefined alarm severities and define new alarm severities for his system. Developers can also change some default values for the predefined alarm severities such as name, value, color, etc.

 Please note that each alarm severity defines a nickname string (most of the time the nickname is just one letter long), which will displayed on the alarm bubble to indicate the severity.

If the color of the predefined alarm severity changed in the runtime, TWaver will update all user interfaces and graphical components to update to use the new color automatically, you don't need to restart or do any effort to make it effective.

You can customize you own alarm severity level. To define a new alarm severity, the following information needs to be provided:

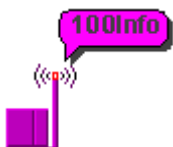
- Name: an unique string to dedicate this alarm severity (like "Critical")
- Nickname: A letter or a short string that will be used to identify the alarm severity.
- Value: An int value to determine the severe level.
- Color: A color used to display this alarm severity.
- Display name: The name text to be displayed.

Before use it, you should register the new alarm severity first. In the following example, a new severity named "INFORMATION" is created:

```
//register this severity to TWaver.
AlarmSeverity.registerAlarmSeverity("Information",
    "Info",
    150,
    Color.magenta,
    "Information");
```

Now you can use it like normal predefined severities:

```
Node node=new Node();
box.addElement(node);
AlarmSeverity severity= AlarmSeverity.getByName("Information");
node.getAlarmState().setNewAlarmCount(severity,100);
```



Customizing AlarmSeverity Comparator

The AlarmSeverity comparator is used to compare alarm severities and find which is more severe. In your application, you may want to change the default compare rule to meet specific needs. By the method AlarmSeverity.setSeverityComparator(Comparator c), the default alarm severity comparator can be changed.

The default TWaver comparator works in following way:

```
new Comparator(){
    public int compare(Object o1, Object o2) {
        AlarmSeverity severity1 = (AlarmSeverity) o1;
        AlarmSeverity severity2 = (AlarmSeverity) o2;
        if (severity1 != null && severity2 != null) {
            return severity1.getValue() - severity2.getValue();
        }
        if (severity1 != null && severity2 == null) {
            return 1;
        }
        if (severity1 == null && severity2 != null) {
            return -1;
        }
        return 0;
    }
}
```

TWaver defines some default alarm severity comparators, as follows

- Sort ascending: twaver.AlarmSeverityComparator.ASCENDING
- Sort descending: twaver.AlarmSeverityComparator.DECENDING
- The default is twaver.AlarmSeverityComparator.ASCENDING.

Customizing Renderer Color for Different Severity Levels

The different severity level alarm has the different renderer color. You can see the default renderer color in section [Using Alarm Severities](#). The renderer color can be set by the method 'setAlarmColorGenerator'.

- Setting the alarm renderer color for the alarm in global scope

```
twaver.TUIManager  
public static void setAlarmColorGenerator(Generator alarmColorGenerator)
```

- Setting the alarm renderer color in the network

```
twaver.TNetwork  
public void setAlarmColorGenerator(Generator alarmColorGenerator)
```

Using Alarm Model

Alarm model is an alarm container and manager, which comes with DataBox. You can get the alarm model by method `TDataBox.getAlarmModel`. You can add, update, clear or delete alarms by alarm model object. The alarm model will work with the connected DataBox to keep the proper graphical representation when alarm is changed.

- [Managing Alarms](#)
- [Listening Alarm Model](#)
- [Listening Alarm Property Changing](#)
- [Using Alarm Mapping](#)
- [Using AlarmModelQuickFinder](#)

Managing Alarms

You can use following methods to manage alarms contained in alarm model:

Method	Description
iterator	Get the iterator of all alarms.
addAlarm	Add an alarm into the alarm model.
getAlarmByID	Retrieve the alarm object by the alarm ID.
clearAlarmByID	Clear the alarm by ID.
acknowledgeAlarmByID	Acknowledge the alarm by specified ID.
confirmAlarmByID	Same with acknowledgeAlarmByID
isAlarmExists	Whether specified alarm is exist in the alarm model.
removeAlarmByID & removeAlarm	Remove alarm from alarm model.
removeAllAlarms	Remove all alarm of the alarm model.
getAlarmsByElement	Get all alarms occurred on the element.
removeAlarmsByElement	Remove all alarms occurred on the element.
recalculateAlarmState	Recalculate and update each element object of the DataBox according to all alarm objects.

Listening Alarm Model

You can add/remove alarm model listener on an alarm model. Listener will be callbacked and an AlarmModelEvent will be sent to the listener when any alarm is added, removed or cleared.

To add/remove alarm model listener on an AlarmModel:

Method	Description
addAlarmModelListener	Add an alarm model listener on the alarm model.
removeAlarmModelListener	Remove the given alarm model listener from alarm model.

Class twaver.AlarmModelAdapter is an empty implementation of interface twaver.AlarmModelListener.

Method	Description
alarmAdded	Called when new alarm instance add into alarm model.
alarmRemoved	Called when alarm instance removed from alarm model.
alarmCleared	Called when alarm model was cleared.



Method AlarmModelListener.alarmCleared(AlarmModelEvent e) will be called when clear the alarm model. In this case, e.getAlarm() will return null.

Listening Alarm Property Changing

You can add/remove alarm property change listener on an alarm model. Listener will be callbacked and an event will be sent to the listener when any property of any alarm is changed. TWaver use java.beans.PropertyChangeListener as the alarm property change listener.

To add/remove alarm property change listener on an AlarmModel

Method	Description
addAlarmPropertyChangeListener	Add an alarm property change listener on the alarm model.
removeAlarmPropertyChangeListener	Remove an alarm property change listener from the alarm model.

Here is an example for using alarm model listener.

```

TDataBox box = new TDataBox();
Node node1 = new Node();
box.addElement(node1);
Node node2 = new Node();
box.addElement(node2);
final AlarmModel model = box.getAlarmModel();
model.addAlarmPropertyChangeListener(new PropertyChangeListener(){
    public void propertyChange(PropertyChangeEvent evt) {
        System.out.println(((Alarm)evt.getSource()).getAlarmID() + "s property changed - " + evt.getPropertyName() + ":" +
            evt.getNewValue());
    }
});
model.addAlarmModelListener(new AlarmModelListener() {
    public void alarmAdded(AlarmModelEvent e) {
        System.out.println("added a alarm:" + e.getAlarm().getAlarmID() + "");
    }

    public void alarmCleared(AlarmModelEvent e) {
        System.out.println("cleared alarm model");
    }

    public void alarmRemoved(AlarmModelEvent e) {
        System.out.println("removed a alarm:" + e.getAlarm().getAlarmID() + "");
    }
});
model.addAlarm(new Alarm("alarm1_1", node1.getID(), AlarmSeverity.getNonClearedRandomSeverity());
model.addAlarm(new Alarm("alarm1_2", node1.getID(), AlarmSeverity.getNonClearedRandomSeverity());
model.addAlarm(new Alarm("alarm2_1", node2.getID(), AlarmSeverity.getNonClearedRandomSeverity());
model.acknowledgeAlarmByID("alarm1_1");
model.clearAlarmByID("alarm1_1", new Date());
model.removeAlarmByID("alarm1_1");

System.out.println("node1's alarm count is " + node1.getAlarmState().getAlarmCount());
model.clear();
System.out.println("after alarm model cleared , node1's alarm count is " + node1.getAlarmState().getAlarmCount());

```

Result:

```

added a alarm:'alarm1_1'
added a alarm:'alarm1_2'

```

```
added a alarm:'alarm2_1'  
alarm1_1's property changed - ackTime:Mon Nov 17 10:25:30 CST 2008  
alarm1_1's property changed - acked:true  
alarm1_1's property changed - clearedTime:Mon Nov 17 10:25:30 CST 2008  
alarm1_1's property changed - cleared:true  
removed a alarm:'alarm1_1'  
node1's alarm count is 1  
cleared alarm model'  
after alarm model cleared , node1's alarm count is 0
```

Using Alarm Mapping

By default, one Alarm object connects to one Element instance, which indicates that this Alarm occurred on that Element object. With class `twaver.AlarmElementMapping`, you can change this relationship to many-to-many.

```
public interface AlarmElementMapping {
    //return all alarms of the Element
    public Object getCorrespondingAlarms(TDataBox box, Element element);
    //return all elements corresponding the Alarm
    public Object getCorrespondingElements(TDataBox box, Alarm alarm);
}
```

Below shows you how to make a many-to-many relationship with a property "mappingID":

```
public class AlarmMappingTest {
    private static String associateKey="mappingID";
    public static void main(String[] args) {
        TDataBox box=new TDataBox();
        TNetwork network=new TNetwork(box);
        box.getAlarmModel().setAlarmElementMapping(new CustomAlarmElementMapping(box));
        for(int j=1; j<=5; j++){
            for(int i=1;i<=2;i++){
                Node node = new Node();
                node.setName("Node_" + i+"_"+j);
                node.setLocation(j*100, 80*i);
                node.putClientProperty("mappingID", j);
                box.addElement(node);
            }
            Alarm alarm = new Alarm("AlarmID_" + j);
            alarm.setRaisedTime(new Date());
            alarm.setAlarmSeverity(TWaverUtil.getRandomNonClearedSeverity());
            alarm.putClientProperty("mappingID", j);
            box.getAlarmModel().addAlarm(alarm);
        }
        TAlarmTable alarmTable = new TAlarmTable(box);
        alarmTable.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
        JSplitPane mainPane=new JSplitPane();
        mainPane.setOrientation(JSplitPane.VERTICAL_SPLIT);
        mainPane.add(network, JSplitPane.TOP);
        mainPane.add(new JScrollPane(alarmTable), JSplitPane.BOTTOM);
        JFrame frame = new JFrame("Alarm element mapping demo");
        frame.getContentPane().add(mainPane);
        frame.setSize(800, 600);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        TWaverUtil.centerWindow(frame);
        frame.setVisible(true);
        mainPane.setDividerLocation(0.8);
    }
    public static class CustomAlarmElementMapping implements AlarmElementMapping {
        private DataBoxQuickFinder finder = null;
        public CustomAlarmElementMapping(TDataBox box) {
            finder = box.createClientPropertyFinder(associateKey);
        }

        //find all "mappingID" matched alarms from AlarmModel
        public Object getCorrespondingAlarms(TDataBox box, Element element) {
            if (element != null){
                Object mappingID = element.getClientProperty(associateKey);
                if(mappingID!=null){
```

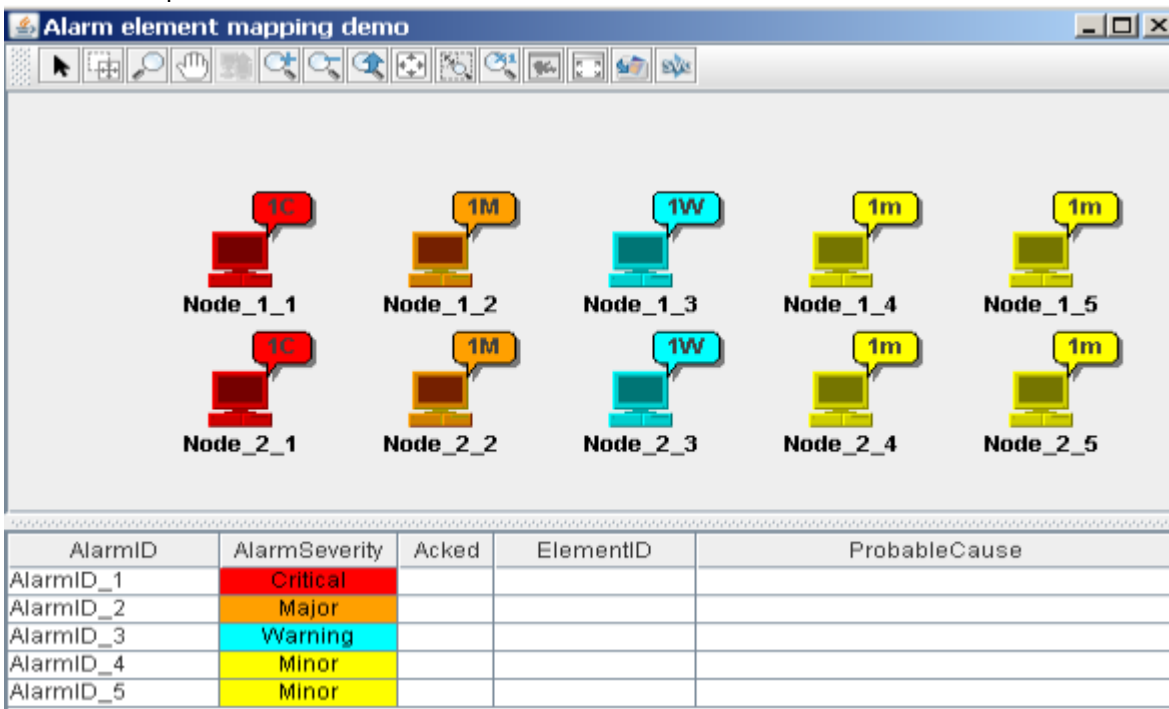


```

List list = new ArrayList();
Iterator it = box.getAlarmModel().iterator();
while (it.hasNext()) {
    Alarm alarm = (Alarm) it.next();
    if (mappingID.equals(alarm.getClientProperty(associateKey))) {
        list.add(alarm);
    }
}
return list;
}
return null;
}
//find all Elements "mappingID" matched from DataBox.
//Here we use twaver.DataBoxQuickFinder
public Object getCorrespondingElements(TDataBox box, Alarm alarm) {
    if (alarm != null) {
        Object mappingID = alarm.getClientProperty(associateKey);
        if (mappingID != null) {
            return finder.find(mappingID);
        }
    }
    return null;
}
}
}

```

Run this example:



AlarmID	AlarmSeverity	Acked	ElementID	ProbableCause
AlarmID_1	Critical			
AlarmID_2	Major			
AlarmID_3	Warning			
AlarmID_4	Minor			
AlarmID_5	Minor			

Using AlarmModelQuickFinder

As an extension of QuickFinder, AlarmModelQuickFinder can help to find the alarm from the alarm model quickly.

twaver.AlarmModel
 public AlarmModelQuickFinder createAlarmFinder(String propertyName)
 Defines an AlarmModelQuickFinder

twaver.AlarmModelQuickFinder
 public List find(Object value)
 Finds the suitable alarm list

public Alarm findFirst(Object value)
 Get sthe first suitable alarm

public boolean interested(Alarm alarm)
 This method determines whether the finder has interesting in the given alarm. By default true is return, so all alarm in the alarm model are considered. User can override this method to customize the interested alarms.

public void dispose()
 Disposes all related resources

```
TDataBox box = new TDataBox();
Node node = new Node();
box.addElement(node);
String key = "alarmKey";
for (int i = 0; i < 17; i++) {
    Alarm alarm = new Alarm();
    alarm.putClientProperty(key, i % 5);
    box.getAlarmModel().addAlarm(alarm);
}

AlarmModelQuickFinder alarmFinder = box.getAlarmModel().createAlarmFinder(key);
System.out.println("find " + alarmFinder.find(new Integer(0)).size() + " alarms with '" + key + "' property is '0'");
System.out.println("find " + alarmFinder.find(new Integer(1)).size() + " alarms with '" + key + "' property is '1'");
System.out.println("find " + alarmFinder.find(new Integer(2)).size() + " alarms with '" + key + "' property is '2'");
```

Results:

```
find 4 alarms with 'alarmKey' property is '0'
find 4 alarms with 'alarmKey' property is '1'
find 3 alarms with 'alarmKey' property is '2'
```

Using AlarmStateStatistics

twaver.AlarmStateStatistics is designed to calculate alarm state of elements in connected DataBox. You can calculate the special alarm state by setting the alarm type and filter.

```
public AlarmStateStatistics()
```

```
public AlarmStateStatistics(TDataBox box)
```

```
public AlarmStateStatistics(TDataBox box, List alarmSeverities)
```

Creates a AlarmStateStatistics instance

```
public void setVisibleFilter(VisibleFilter visibleFilter)
```

Sets a new visible filter for this alarm statistic

```
public void reset()
```

Calls this method to recalculate all alarm count of elements in DataBox

```
public PropertyChangeSupport getPropertyChangeSupport()
```

Gets propertyChangeSupport

```
public int getNewAlarmCount()
```

Gets the count of the new alarm

```
public int getAcknowledgedAlarmCount()
```

Gets the acknowledged alarm count of elements in the connected DataBox

```
public int getTotalAlarmCount()
```

Gets the total alarm count of elements in connected DataBox

```
public int getNewAlarmCount(AlarmSeverity severity)
```

Get new alarm count for specified alarm severity

```
public int getAcknowledgedAlarmCount(AlarmSeverity severity)
```

Gets acknowledged alarm count for specified alarm severity

```
public int getTotalAlarmCount(AlarmSeverity severity)
```

Gets alarm total count for specified alarm severity

The following example shows how the AlarmStateStatistics is used

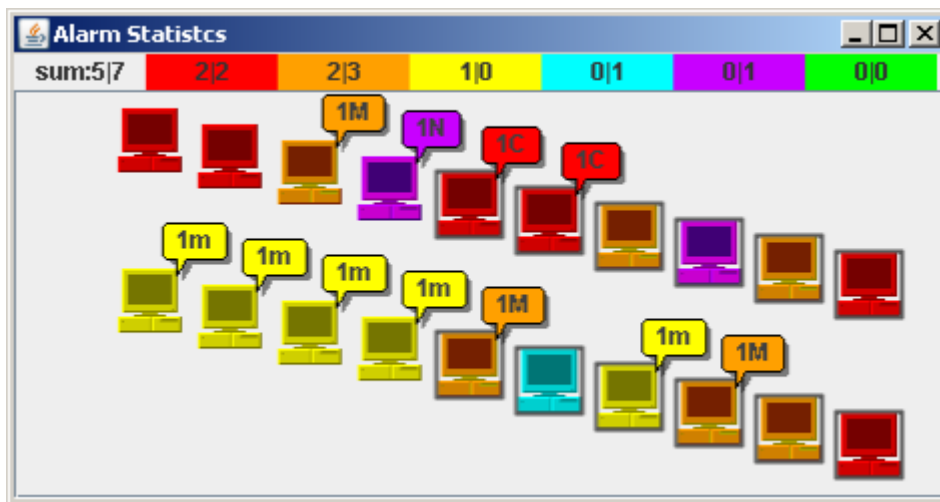
```
public static class AlarmStatisticsLabel extends JPanel {
    private AlarmStateStatistics statistics = null;
    private JLabel sumLabel = new JLabel();
    private Map labels = new HashMap();
    public AlarmStatisticsLabel(final AlarmStateStatistics statistics) {
        this.statistics = statistics;
        statistics.getPropertyChangeSupport().addPropertyChangeListener(TWaverConst.ALARMSTATE, new
        PropertyChangeListener() {
            public void propertyChange(PropertyChangeEvent evt) {
                updateStatistics();
            }
        });
        init();
    }
    private void init() {
        Collection severities = AlarmSeverity.getAllSeverity();
        this.setLayout(new GridLayout(1, severities.size() + 1));
        this.add(sumLabel);
        sumLabel.setHorizontalAlignment(SwingConstants.CENTER);
        Iterator it = severities.iterator();
        while (it.hasNext()) {
            AlarmSeverity severity = (AlarmSeverity) it.next();
            JLabel label = new JLabel();
```

```

label.setHorizontalAlignment(SwingConstants.CENTER);
this.add(label);
labels.put(severity, label);
label.setOpaque(true);
label.setBackground(severity.getColor());
}
}
public void updateStatistics() {
Iterator it = labels.entrySet().iterator();
while (it.hasNext()) {
Entry entry = (Entry) it.next();
AlarmSeverity severity = (AlarmSeverity) entry.getKey();
JLabel label = (JLabel) entry.getValue();
int newCount = statistics.getNewAlarmCount(severity);
int ackCount = statistics.getAcknowledgedAlarmCount(severity);
label.setText(newCount + "|" + ackCount);
}
sumLabel.setText("sum:" + statistics.getNewAlarmCount() + "|"
+ statistics.getAcknowledgedAlarmCount());
}
}

final TDataBox box = new TDataBox();
TNetwork network = new TNetwork(box);
final AlarmStateStatistics statistics = new AlarmStateStatistics(box);
final AlarmStatisticsLabel label = new AlarmStatisticsLabel(statistics);
network.add(label, BorderLayout.NORTH);
box.getSelectionModel().addDataBoxSelectionListener(
new DataBoxSelectionListener() {
public void selectionChanged(DataBoxSelectionEvent e) {
if (box.getLastSelectedElement() == null) {
statistics.setVisibleFilter(null);
}
else { statistics.setVisibleFilter(new VisibleFilter() {
public boolean isVisible(Element element) {
return element.isSelected();
}
});
}
});
for (int i = 0; i < 20; i++) {
Node node = new Node();
node.setLocation(100 + (i % 10) * 40, 80 * (i + 1) / 10);
box.addElement(node);
AlarmSeverity severity = TWaverUtil.getRandomNonClearedSeverity();
Alarm alarm = new Alarm(null, node.getID(), severity);
alarm.setAcked(TWaverUtil.getRandomBool());
box.getAlarmModel().addAlarm(alarm);
}
JFrame frame = new JFrame();
frame.getContentPane().add(network);
frame.setSize(800, 600);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
TWaverUtil.centerWindow(frame);
frame.setTitle("Alarm Statistics");
frame.setVisible(true);

```



Creating Alarm by XML

You can load alarms from XML with the similar way of loading element data.

```
TDataBox box=new TDataBox();
box.parse("alarms.xml");
```

Following is an example of an alarm defined in XML format.

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.2_15" class="java.beans.XMLDecoder">
  <object class="twaver.Alarm">
    <string>ALM0001</string>
    <void property="ackTime">
      <object class="java.util.Date">
        <long>1184656133828</long>
      </object>
    </void>
    <void property="ackUserID">
      <string>Ack user name</string>
    </void>
    <void property="acked">
      <boolean>true</boolean>
    </void>
    <void property="additionalText">
      <string>Additional text</string>
    </void>
    <void property="alarmSeverity">
      <object class="twaver.AlarmSeverity" method="getByName">
        <string>Major</string>
      </object>
    </void>
    <void property="alarmType">
      <object class="twaver.base.enumerable.EnumerableManager"
        method="getEnumerable">
        <class>twaver.AlarmType</class>
        <string>EquipmentAlarm</string>
      </object>
    </void>
    <void property="clearUserID">
      <string>Clear user name</string>
    </void>
    <void property="cleared">
      <boolean>true</boolean>
    </void>
    <void property="clearedTime">
      <object class="java.util.Date">
        <long>1184656133828</long>
      </object>
    </void>
    <void property="elementID">
      <string>node02</string>
    </void>
    <void property="lastChangedTime">
      <object class="java.util.Date">
        <long>1184656133828</long>
      </object>
    </void>
    <void property="probableCause">
```

```

<object class="twaver.base.enumerable.EnumerableManager"
  method="getEnumerable">
  <class>twaver.AlarmProbableCause</class>
  <string>Low cable pressure</string>
</object>
</void>
<void property="proposedRepairAction">
  <string>Junk it now</string>
</void>
<void property="raisedTime">
  <object class="java.util.Date">
    <long>1184656133828</long>
  </object>
</void>
<void property="specificProblem">
  <string>no specific problem</string>
</void>
<void property="trendIndication">
  <object class="twaver.base.enumerable.EnumerableManager"
    method="getEnumerable">
    <class>twaver.AlarmTrendIndication</class>
    <string>More severe</string>
  </object>
</void>
<void method="addComment">
  <string>first comment</string>
</void>
<void method="addComment">
  <string>second comment</string>
</void>
<void method="addCorrelatedAlarmID">
  <string>ALM0008</string>
</void>
<void method="addCorrelatedAlarmID">
  <string>ALM0009</string>
</void>
<void method="putClientProperty">
  <string>key1</string>
  <string>This is key1 value.</string>
</void>
<void method="putClientProperty">
  <string>key2</string>
  <object class="java.util.Date">
    <long>1184656133828</long>
  </object>
</void>
<void method="putClientProperty">
  <string>key3</string>
  <string>test value.</string>
</void>
</object>
</java>

```

Creating Alarm by API

Now we'll create the same alarm instance as above by API, and add it into DataBox.

```
TDataBox box=new TDataBox();

Alarm alarm=new Alarm("alarm0001");
alarm.setAckTime(new Date());
alarm.setAckUserID("Ack user name");
alarm.setAdditionalText("Additional text");
alarm.setAcked(true);
alarm.setLastChangedTime(new Date());
alarm.setClearedTime(new Date());
alarm.setRaisedTime(new Date());
alarm.setAlarmType(AlarmType.EQUIPMENT_ALARM);
alarm.setClearUserID("Clear user name");
alarm.addComment("first comment");
alarm.addComment("second comment");
alarm.addCorrelatedAlarmID("Alarm0008");
alarm.addCorrelatedAlarmID("Alarm0009");
alarm.setElementID("node02");
alarm.setAlarmSeverity(AlarmSeverity.MAJOR);
alarm.setProbableCause(AlarmProbableCause.LOW_CABLE_PRESSURE);
alarm.setProposedRepairAction("Junk it now");
alarm.setSpecificProblem("no specific problem");
alarm.setTrendIndication(AlarmTrendIndication.MORE_SEVERE);

//add the alarm into alarm model of DataBox.
box.getAlarmModel().addAlarm(alarm);
```


Customizing Alarm Renderer Color

The renderer color of the alarm can be changed by following methods. The default color is the color of the maximal alarm.

```
twaver.TNetwork
public void setAlarmColorGenerator(Generator alarmColorGenerator)
```

```
twaver.TUIManager
public static void setAlarmColorGenerator(Generator alarmColorGenerator)
```

As follows:

```
public class AlarmColorGenerator implements Generator {
    public Object generate(Object object) {
        Element element = null;
        if(object instanceof ElementUI){
            ElementUI ui = (ElementUI)object;
            element = ui.getElement();
        }else{
            element = (Element)object;
        }
        AlarmSeverity severity = element.getAlarmState().getHighestNewAlarmSeverity();
        if (severity != null){
            return severity.getColor();
        }
        return null;
    }
}
```

Using TWaverUtil

- [Using Resource Agent](#)

Using Resource Agent

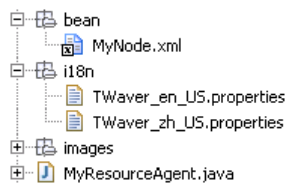
The default i18n property files and Element configurations are stored in path twaver.jar/resource. However you can change to your own resource path by following way:

```

TWaverUtil
public static void init(Locale locale, Class resourceAgent)
public static void setResourceAgent(Class resourceAgent)

```

The structure should looks like below:



- bean folder stores Element property config file.
- i18n folder contains i18n property files.
- MyResourceAgent is a resource locator used to locate resource location. An empty class is ok.

When you need to use resource agent?

Two cases:

1. Change the default i18n translation to your own. You should have all files named like TWaver_*.properties in your i18n folder so TWaver can find them. See [Changing Internal i18n Strings](#) for more details.
2. All predefined property definition of all Elements. The property definition files stored in bean folder and has the same name with the related class. For example, A class "MyNode" should be configured in file "bean/MyNode.xml". You can also do the same thing by calling API:

```

TWaverUtil.registerBeanInfo(MyNode.class, MyResourceAgent.getResource("bean\\MyNode.xml").getFile());

```

Working with XML

XML plays a very important role in TWaver. You do followings by XML:

- Persistenct network data by XML
- Define Element bean properties by XML
- Config TWaver by XML

See below for more information:

- [XML Long-Term Persistence for JavaBeans](#)
- [Using TWaver.XML](#)

XML Long-Term Persistence for JavaBeans

TWaver use the Java XML Long-Term persistence for JavaBeans to save and transfer DataBox data. This section described both the syntax and semantics of the XML schema used by XMLEncoder for saving archives of beans. A DTD that describes the syntax of the schema is given.

Following links can give you more detail information about Java Long-Term persistence for JavaBeans:

<http://java.sun.com/products/jfc/tsc/articles/persistence1/index.html>
<http://java.sun.com/products/jfc/tsc/articles/persistence2/index.html>
<http://java.sun.com/products/jfc/tsc/articles/persistence3/index.html>
<http://java.sun.com/products/jfc/tsc/articles/persistence4/index.html>
<http://java.sun.com/products/jfc/tsc/articles/persistence3/javabeans.dtd>

The article <http://java.sun.com/products/jfc/tsc/articles/persistence3/index.html> described both the syntax and semantics of the XML schema used by XMLEncoder for saving archives of beans. A DTD <http://java.sun.com/products/jfc/tsc/articles/persistence3/javabeans.dtd> that describes the syntax of the schema is in the file javabeans.dtd.

- [Creating Data by XML](#).
- [Updating Data by XML](#)
- [Deleting Data by XML](#)
- [Using XML Persistent Filter](#)
- [Transient Element Property](#)

Creating Data by XML.

Following xml snippet defines a node element. All syntax and semantics are following java beans specifications.

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.2_15" class="java.beans.XMLDecoder">
  <object class="twaver.Node">
    <string>abc</string>  <!--id of the element -->
    <void property="location">
      <object class="java.awt.Point">
        <int>100</int>
        <int>100</int>
      </object>
    </void>
    <void property="name">
      <string>I was created by create.xml</string>
    </void>
  </object>
</java>
```

Updating Data by XML

Following xml snippet can be used to update properties of the node with id "abc". All syntax and semantics are following java beans specifications. The only different thing on updating data with XML is that you should use a specified object class and method:

```
<object class="twaver.DataBoxParserAgent" method="getElementByID">
```

You don't need to know the details of the internal class DataBoxParserAgent and its methods. Just use them to update data by XML like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.2_01" class="java.beans.XMLDecoder">
  <object class="twaver.DataBoxParserAgent" method="getElementByID">
    <string>abc</string> <!--id of the element -->
    <void property="name">
      <string>I was updated by update.xml</string>
    </void>
    <void property="alarmState">
      <object class="twaver.AlarmState">
        <void method="setNewAlarmCount">
          <object class="twaver.AlarmSeverity" method="getByName">
            <string>Major</string>
          </object>
          <int>10</int>
        </void>
      </object>
    </void>
  </object>
</java>
```

Deleting Data by XML

Following xml snippet can be used to delete data from DataBox. All syntax and semantics are following java beans specifications. The only different thing on deleting data with XML is that you should use a specified object class and method:

```
<object class="twaver.DataBoxParserAgent" method="removeElementByID">
```

You don't need to know the details of the internal class DataBoxParserAgent and its methods. Just use them to delete data by XML like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.2_01" class="java.beans.XMLDecoder">
  <object class="twaver.DataBoxParserAgent" method="removeElementByID">
    <string>abc</string> <!--id of the element -->
  </object>
</java>
```


Using XML Persistent Filter

By default, TWaver will encode all Element and all client properties of the Element into XML. But you can use a filter to determine specified Elements and client properties will be saved. Two persistent filters will be used: ClientPropertyPersistentFilter and ElementPersistentFilter.

For example, we want save all nodes which the id starts with 'T' and its client properties which the key start with 'T'.

```
TDataBox box = new TDataBox();
//add some data.
Node node = new Node("T-Node");
node.putClientProperty("key1", "value1");
node.putClientProperty("T-key2", "value2");
node.putClientProperty("key3", "value3");
box.addElement(node);

node = new Node("X-Node");
node.putClientProperty("key1", "value1");
node.putClientProperty("T-key2", "value2");
node.putClientProperty("key3", "value3");
box.addElement(node);

//define an element persistent filter
//this filter only save the element which key start with 'T'
ElementPersistentFilter elementFilter = new ElementPersistentFilter() {
    public boolean isTransient(Element element) {
        return !element.getID().toString().startsWith("T");
    }
};

//define a client property persistent filter
//this filter only save the properties which key start with 'T'
ClientPropertyPersistentFilter propertyFilter = new ClientPropertyPersistentFilter() {
    public boolean isTransient(Element element, Object clientPropertyKey) {
        return !clientPropertyKey.toString().startsWith("T");
    }
};

//use these filters to output XML
DataBoxOutputSetting outputSetting = new DataBoxOutputSetting();
outputSetting.setClientPropertyFilter(propertyFilter);
outputSetting.setElementFilter(elementFilter);
outputSetting.setOutputStream(System.out);

try {
    box.output(outputSetting);
} catch (IOException e) {
    e.printStackTrace();
}
```

The output XML is:

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.2_16" class="java.beans.XMLDecoder">
<object class="twaver.Node">
<void method="putClientProperty">
<string>T-key2</string>
<string>value2</string>
</void>
```

```
</object>  
</java>
```



method name of this filter is isTransient, so *return true means NOT save it.

Transient Element Property

By default, Java XMLEncoder will try to encode all Element properties. But you can make some property transient. DataBox will ignore the transient properties.
Now we define a new attribute for a Node, then we make it transient. Here is the definition of attribute class:

```
public class Attribute {  
    private String value="";  
    public String getValue() {  
        return value;  
    }  
}
```

Then we define a new Node:

```
public class MyNode extends Node{  
    private Attribute attribute;  
    public MyNode(){  
  
    }  
  
    public MyNode(Object id){  
        super(id);  
    }  
  
    public Attribute getAttribute() {  
        return attribute;  
    }  
  
    public void setAttribute(Attribute attribute) {  
        this.attribute = attribute;  
    }  
}
```

Finally, we tell TWaver persistent manager to ignore this attribute:

```
PersistenceManager.registerClassDelegate(Attribute.class, PersistenceManager.TRANSIENT_DELEGATE);
```

Using TWaver.XML

TWaver is designed to present data for various type of networks. There are large number of configuration parameters in TWaver components. Fortunately, most have sensible default values. Also, TWaver is distributed with an example TWaver.xml file twaver.jar that shows the various options. You can put the example file in your classpath and customize it. TWaver will find TWaver.XML file in the package path which the TWaver resource agent point to.

Use following code specifies the new resource agent location:

```
TWaverUtil.setResourceAgent(com.mycompany.myproduct.Test.class);
```

TWaver will try to find the TWaver.XML file in package com.mycompany.myproduct.

- [Alarm Severity Configuration](#)
- [Icon Attachment Definition](#)
- [Property Sheet Category Definition](#)
- [Network Toolbar Configuration](#)
- [UI Default Configuration](#)

Alarm Severity Configuration

You can define alarm severities in TWaver.xml file. Following configuration defines the default alarm severities of TWaver. You can change it to meet your application needs, or add new alarm severity items.

```
<alarm>
<severity name="Critical" nickName="C" value="500" color="255|0|0"/>
<severity name="Major" nickName="M" value="400" color="255|160|0"/>
<severity name="Minor" nickName="m" value="300" color="255|255|0"/>
<severity name="Warning" nickName="W" value="200" color="0|255|255"/>
<severity name="Indeterminate" nickName="N" value="100" color="200|0|255"/>
<severity name="Cleared" nickName="R" value="0" color="0|255|0"/>
</alarm>
```

Icon Attachment Definition

In the above section we know how to create and add an icon attachment to an Element. Now you can define icon attachment in TWaver.xml for following use:

```
<iconattachments>
<attachment name="disabled" iconurl="/myImages/disabled.png"/>
....
</iconattachments>
```

You should specify the attachment name and icon URL. After that, you can use the attachment by following code:

```
element.putClientProperty("StateIcon:disabled", Boolean.TRUE);
```

Property Sheet Category Definition

Following code defines a "system" property category for property sheet component.

```
<propertysheet>  
<category name="system" displayName="" isExpend="true"/>  
</propertysheet>
```

Network Toolbar Configuration

In TWaver.xml you can define new buttons and new toolbars.

Define new toolbar button with TWaver.XML:

```
<toolbars>
<buttons>
<button id="Selection"
class="twaver.network.toolbar.NetworkSelectButton"/>
<button id="GroupSelection" class="twaver.network.toolbar.NetworkGroupSelectionButton"/>
</buttons>
</toolbars>
```

Define new toolbar with TWaver.XML:

```
<toolbars>
<buttons>
....
</buttons>
<toolbar name="default">
<button id="Selection"/>
<button id="GroupSelection"/>
....
</toolbar>
<toolbar name="myToolbar">
...
</toolbar>
</toolbars>
```


UI Default Configuration

The UI objects used to render Element objects. Most of them have various paint options. You can change the default options of them by class `twaver.TUIManager`. But you can also do the same thing by `TWaver.xml`:

```
<uidefault>
<property key="animating.min.interval" value="500"/>
<property key="animating.blink.interval" value="1000"/>
<property key="element.select.color" value="255|255|255"/>
<property key="element.select.stroke" value="2"/>
<property key="link.select.color" value="255|255|255"/>
<property key="link.select.stroke" value="2"/>
....
</uidefault>
```

Here you can change the default settings of any UI classes.

Internationalization

TWaver use standard Java i18n framework to display essential string resources. It includes labels, tooltips, column names, property display names etc of all components. All the string resources are stored in package / resource/i18n of TWaver.jar file.

TWaver currently provides English and Chinese resources. More language will be added in the future. Developers can change these resource files to give a different translation.


Component	Description	i18nable
Network	Canvas tooltip text	Yes
	Tooltip text of toolbar buttons	Yes
PropertySheet	Table column header	Yes
	Display names of all properties	Yes
Other	All kinds of dialogs, messages	Yes

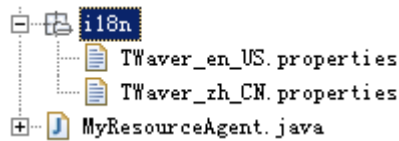
- [Changing Internal i18n Strings](#)
- [Adding i18n Strings](#)
- [Adding I18n Element Property](#)
- [Customizing i18n](#)

Changing Internal i18n Strings

You can update resource files located in the package `/resource/i18n` to change the internal string resources of TWaver. But normally you don't need to do that. TWaver provides a more convenient way to replace the predefined string or add new strings. You can use `TWaverUtil.setResourceAgent` to specify the customized resource agent class:

```
TWaverUtil.setResourceAgent(MyResourceAgent.class);
```

 class `MyResourceAgent` located package should coincident with the resource path of `twaver.jar`. (See following picture) Otherwise, TWaver will not able to find the resources.



Now, when TWaver try to display an i18n string, it will try to find the resource files from where `MyResourceAgent` class located. If no resources found, then TWaver use the internal resources. That means, TWaver provides a way for you to replace the internal defined i18n strings.

Adding i18n Strings

You can also add new i18n strings by method `setResourceAgent`.



Please note that you should avoid giving the new resource's key the same key with TWaver internal resources.

Adding I18n Element Property

When you display an element object with PropertySheet component, all properties will list there with key-value pair in the table. To define a new Element Person and add a new client property "age", you can do that as following steps:

Define Class

```
public static class Person extends Node {
    public Person() {
    };
    public Person(String name, int age) {
        this.setName(name);
        this.putClientProperty("age", age);
    }
}
```

Define XML File

```
<beaninfo>
...
<attribute
clientPropertyKey="age"
displayName=" i18nKey.myAge"/>
</beaninfo>
```

I18n Property

The customized TWaver_en_US.properties

```
myAge = My Age
...
```

The customized TWaver_zh_CN.properties

```
myAge = \u6211\u7684\u5e74\u9f84
...
```

Now the property "age" will be listed in PropertySheet component. The property name will be displayed as i18n string "My Age".

```
TWaverUtil.setResourceAgent(MyResourceAgent.class);
TDataBox box = new TDataBox();
TElementTable table=new TElementTable(box);
table.registerElementClassXML(Person.class, "/demo/i18ntest/table2.xml");
table.setElementClass(Person.class);
box.addElement(new Person("Sam",89));
```

```
box.addElement(new Person("Tom",12));  
box.addElement(new Person("Lisa",20));
```

Customizing i18n

TWaver provides several language translations of all internal used strings. All these translated resources are saved in property files of path /resources/i18n.

However, it is possible for you to change these translations dynamically in the runtime without changing the property files. Use a resource generator `TWaverUtil.setI18nGenerator(Generator generator)` you can hook the resources and return yours:

```
public class MyI18nGenerator implements Generator{
    public Object generate(Object key) {
        if(key.equals("myAge")){
            if(TWaverUtil.getLocale().equals(Locale.CHINA)){
                return "\u6211\u7684\u5e74\u9f84";
            }else{
                return "My Age";
            }
        }
        return null;
    }
}
```

```
TWaverUtil.setI18nGenerator(new MyI18nGenerator());
```

TWaver Java FAQ

TWaver developers should read the complete FAQ. Here we addressed most common questions of our customers. If you still can not find your answer, please feel free to contact us.

- [Chart FAQ](#)
 - [Can I make a popup menu for a Chart?](#)
 - [What's the differences between Chart Value and List Values?](#)
- [Common FAQ](#)
 - [Why can not I change the default locale of TWaver?](#)
- [DataBox Element FAQ](#)
 - [Are element.putClientProperty\("name", "ABC"\) and element.setName\("ABC"\) same?](#)
 - [How to combine a packaged Element class with a non-packaged ElementUI class?](#)
 - [How to use DataBoxQuickFinder to find element or alarm in DataBox?](#)
 - [In TWaver, what's the difference between Border Outline & StateOutline?](#)
 - [What happened after removing a element from databox](#)
 - [Why can not I add element property change listener to a databox](#)
- [License FAQ](#)
 - [How to use license](#)
- [Network FAQ](#)
 - [Can alarm or message attachment be displayed in front of any elements?](#)
 - [Can I change the flashing speed for the blinking elements?](#)
 - [Can I use animated GIF for node image?](#)
 - [I add KeyListener or MouseListener to Network, why it does not work?](#)
- [Others FAQ](#)
 - [How to connect X11 window server?](#)
 - [How to intercept TWaver exception?](#)
 - [How to make interaction between Applet and JavaScript?](#)
 - [How to set image file path?](#)
 - [My web application runs fine on my Windows development machine, but when I deploy it to the Unix or Linux production server, it doesn't work. What is the problem?](#)
- [Swing FAQ](#)
 - [How to change the line style of TWaver Tree nodes?](#)
 - [Why I change table column width, it does not work?](#)
- [Table Tree Sheet FAQ](#)
 - [How to add tree expand listener](#)
 - [How to set tree column's tooltip for tree table](#)
 - [I'm using tristate tree. Can I make the parent node unselected when there is no child node selected?](#)
 - [In TTableModel, what's the difference between methods getPublishedColumn\(\) and getPublishedData\(\)?](#)

Chart FAQ

- [Can I make a popup menu for a Chart?](#)
- [What's the differences between Chart Value and List Values?](#)

Can I make a popup menu for a Chart?

Yes. Use following code:

```
TUIManager.registerDefault(TWaverConst.TCHART_POPUP_MENU_GENERATOR,  
    ChartPopupMenuGenerator.INSTANCE);
```

What's the differences between Chart Value and List Values?

A TWaver Element can contain single chart value, or a set chart values. You can use following methods to handle the values:

twavr.Element:

```
//Set a single chart value
public void putChartValue(double value);
//use "add" to add a set chart values
public void addChartValue(double value);
//this is for bubble chart
public void addChartBubble(Bubble bubble);
```

Chart will use those values to paint graphics. In TWaver, the single chart value called Chart Value. The set of values called Chart List Values. They are separated data for an Element and you can select Chart Value or Chart List Values as the chart data source.

Below explains what data will be used as data source for different chart:

Chart Type	Data Used
BarChart	
BAR_TYPE_DEFAULT	Chart Value
BAR_TYPE_GROUP	Chart List Values
BAR_TYPE_STACK	Chart List Values
BAR_TYPE_LAYER	Chart List Values
BAR_TYPE_PERCENT	Chart List Values
LineChart	
LINE_TYPE_DEFAULT	Chart List Values
LINE_TYPE_AREA	Chart List Values
LINE_TYPE_POLE	Chart List Values
PieChart	Chart Value
PercentChart	
PERCENT_TYPE_HORIZONTAL	Chart Value
PERCENT_TYPE_VERTICAL	Chart Value
DialChart	
DIAL_TYPE_WHOLE	Chart Value
DIAL_TYPE_ARC	Chart Value
RadarChart	Chart List Values

BubbleChart	Chart List Values
-------------	-------------------

Common FAQ

- [Why can not I change the default locale of TWaver?](#)

Why can not I change the default locale of TWaver?

Use following code to change the default locale and resource agent:

```
TWaverUtil.init(TWaverConst.EN_US, DemoResourceAgent.class);
```



Put this snippet before using any classes of TWaver.

DataBox Element FAQ

- [Are element.putClientProperty\("name", "ABC"\) and element.setName\("ABC"\) same?](#)
- [How to combine a packaged Element class with a non-packaged ElementUI class?](#)
- [How to use DataBoxQuickFinder to find element or alarm in DataBox?](#)
- [In TWaver, what's the difference between Border Outline & StateOutline?](#)
- [What happened after removing a element from databox](#)
- [Why can not I add element property change listener to a databox](#)

Are element.putClientProperty("name", "ABC") and element.setName("ABC") same?

No. "Name" property is a Java Bean property of Element which you should use setName to set its value. Use client property to add a "name", this will be a client property, it will go to the client property map of this Element.

In TWaver, an Element has four types of property:

- JAVA Bean
- Client Property
- User property
- Business Object

When you change the property value, a property change event will be fired in different way:

JAVA Bean

For a Java Bean property, for instance "Name", when you change name by method setName(..), it will fire a property change event:

```
public void setName(String name) {
    String oldValue = this.name;
    this.name = name;
    this.firePropertyChange(TWaverConst.PROPERTYNAME_NAME, oldValue, name);
}
```

Client Property

```
public void putClientProperty(Object key, Object value) {
    ...
    String propertyName = TWaverConst.CLIENT_PROPERTY_PREFIX + key.toString();
    firePropertyChange(propertyName, oldValue, value);
}
```

User Property

```
public void putUserProperty(Object key, Object value){
    ...
    Object oldValue = userProperties.get(key);
    ...
    String propertyName = TWaverConst.USER_PROPERTY_PREFIX + key.toString();
    firePropertyChange(propertyName, oldValue, value);
}
```

Business Object

```
public void setBusinessObject(Object businessObject){
    ...
    this.firePropertyChange(TWaverConst.PROPERTYNAME_BUSINESSOBJECT, oldValue, businessObject);
}
```


How to combine a packaged Element class with a non-packaged ElementUI class?

Assumed you defined a new Element object and a related non-package ElementUI as follows: MyNode.java: (defined in package test)

```
package test;
public class MyNode extends Node{
    public MyNode(){super()}
    public MyNode(Object id){super(id)};
    ...
    public String getUIClassID() {
        return "MyNodeUI"; //wrong
        return "/MyNodeUI"; //right.
    }
}
//MyNodeUI.java
public class MyNodeUI extends NodeUI{
    ...
}
```

For the top level package classes, please add "/" to the returned UIClassID string.

How to use DataBoxQuickFinder to find element or alarm in DataBox?

By Class `twaver.DataBoxQuickFinder`, the suitable element list can be found from the `DataBox` quickly. As follows:

```
DataBoxQuickFinder finder = box.createClientPropertyFinder("age");
List result=finder.find (new Integer (22));
//Finds the element list whose displayName is 'Server'
DataBoxQuickFinder finder=box.createJavaBeanFinder("displayName");
List result=finder.find("Server");
```

By Class `twaver.AlarmModelQuickFinder`, the suitable alarm list can be found from the `AlarmModel` of the `DataBox` quickly. Finds all the alarm whose 'alarmKey' is 'Key1'

```
AlarmModelQuickFinder alarmFinder = box.getAlarmModel().createAlarmFinder("alarmKey");
List result=alarmFinder.find ("Key1");
```

In TWaver, what's the difference between Border Outline & StateOutline?

In TWaver, Border means the Element border when it is selected. All properties about the Border is acting only when the Element is selected. Outline/StateOutline is the outline of the Element in the regular status.

What happened after removing a element from databox

Three steps are needed to delete an Element from DataBox:

1. Change some properties of that Element. For example, for a Link, the "start node" and "end node" need to set to null.
2. Change some properties of the related Elements. For example, to delete a Node, need to delete the connected Links first.
3. Delete from SelectionModel.

Here is the code:

```
public void removeElementByID(final Object id) {
    if (id == null) {
        return;
    }
    if (!this.containsByID(id)) {
        return;
    }
    final Element element = (Element) delegate.get(id);
    Node from = null;
    Node to = null;
    //if link, remove from node
    if (element instanceof Link) {
        Link link = (Link) element;
        from = link.getFrom();
        to = link.getTo();
        link.setFrom(null);
        link.setTo(null);
        if (from == to) {
            this.tagLinkIndex(from, to);
        }
    }

    //if have link, remove link first.
    if (element instanceof Node && ((Node) element).getAllLinks() != null) {
        Object[] links = ((Node) element).getAllLinks().toArray();
        for (int i = 0; i < links.length; i++) {
            this.removeElement((Link) links[i]);
        }
    }
    //if have followers, remove relation
    if (element instanceof Node && ((Node) element).getAllFollowers() != null) {
        Object[] followers = ((Node) element).getAllFollowers().toArray();
        for (int i = 0; i < followers.length; i++) {
            Follower follower = (Follower) followers[i];
            follower.setHost(null);
        }
    }
    Node host = null;
    //if have host, remove relation
    if (element instanceof Follower && ((Follower) element).getHost() != null) {
        host = ((Follower) element).getHost();
        ((Follower) element).setHost(null);
    }
    //if BTS, remove all antenna
    if (element instanceof BTS) {
        ((BTS) element).clearAntennas();
    }

    BTS bts = null;
    //if BTSAntenna, remove from BTS
    if (element instanceof BTSAntenna) {
```

```
bts = ((BTSAntenna)element).getBTS();
((BTSAntenna)element).setBTS(null);
}

//if have children, remove children first.
Object[] children = element.getChildren().toArray();
for (int i = 0; i < children.length; i++) {
    Element child = (Element) children[i];
    removeElementByID(child.getID());
}
Element oldParent = element.getParent();
//if parent not null, remove from parent.
if (element.getParent() != null) {
    element.getParent().removeChild(element);
}
//remove from selection model.
if (this.getSelectionModel().contains(element)) {
    this.getSelectionModel().removeSelection(element);
}
delegate.remove(id);
elements.remove(element);
if (this.rootDelegate.containsKey(element.getID())) {
    this.rootDelegate.remove(element.getID());
    this.rootElements.remove(element);
}
if (element instanceof Equipment) {
    tags.remove( ( (Equipment) element).getTag());
}
this.fireElementRemoved(element, oldParent, from, to, host, bts);

//don't listen this element any more.
element.removePropertyChangeListener(this);
}
```

Why can not I add element property change listener to a databox

Use following code to monitor TWaverConst.PROPERTYNAME_LINK_BUNDLE_EXPAND property change event, but does not work.

Wrong usage:

```
box.addElementPropertyChangeListener(new PropertyChangeListener(){
    public void propertyChange(PropertyChangeEvent evt) {
        //wrong usage: do not use evt.getPropertyName() directly.
        //Use twaver.TWaverUtil.getPropertyName(evt) instead.
        if(evt.getPropertyName().equals(TWaverConst.PROPERTYNAME_LINK_BUNDLE_EXPAND)){
            //code unreachable.
            network.forceAdjustCanvasSize();
        }
    }
});
```

The reason is, property name got by evt.getPropertyName() contains prefix. For example, for Client Properties, the property name will be TWaverConst.CLIENT_PROPERTY_PREFIX+key. See [PropertyName](#) for details.

Right usage:

```
box.addElementPropertyChangeListener(new PropertyChangeListener(){
    public void propertyChange(PropertyChangeEvent evt) {
        if(twaver.TWaverUtil.getPropertyName(evt).equals(TWaverConst.PROPERTYNAME_LINK_BUNDLE_EXPAND)){
            //code reachable.
            network.forceAdjustCanvasSize();
        }
    }
});
```

License FAQ

- [How to use license](#)

How to use license

TWaver has three type licenses: Evaluation License, Development License, and Runtime License. To check the license type and information, press short-cut key "Ctrl+Shift+T" to popup a license information dialog.

- Evaluation License: This license grant you to use and evaluate the software for a period. It can not be used for any commercial purpose. When you use Evaluation License, all visual components will paint a watermark "TWaver Evaluation Version".
- Development License: This license grant you the right to use TWaver to develop your software product or project. No any watermark shown but a license information dialog will popup for every 2 hours.
- Runtime License: This is used for your final project/product deployment. No watermark, no dialog popup.

For for license details, check out [TWaver License and Distribution Agreement](#).

TWaver License will provides the permissions for visual components. The premission structure is like below.

```
PermissionAll
|--PermissionSwing
|   |--PermissionNetwork
|   |--PermissionTree
|   |--PermissionSheet
|   |--PermissionTable
|   |--PermissionList
|   |--PermissionChart
|--PermissionWeb
|   |--PermissionSVG
|--PermissionGIS
|   |--PermissionNetwork
|   |--PermissionGIS
```

For swing user:

```
PermissionAll=N
PermissionSwing=Y
```

For user only buy network and tree component:

```
PermissionAll=N
PermissionNetwork=Y
PermissionTree=Y
```

For user only buy web component:

```
PermissionAll=N
PermissionWeb=Y
```

For user only buy GIS component:

```
PermissionAll=N
PermissionNetwork=Y
```



```
PermissionGIS=Y
```

For user only buy swing and web component:

```
PermissionAll=N
PermissionSwing=Y
PermissionWeb=Y
```

TWaver also contains the license period. If no "EndDate" specified, then this is a permanent license.


```
BeginDate=2008-12-01
EndDate=2009-06-30
```

If you encounter any License validation problems, check following steps to resolve:

- Make sure you already purchased the license
- Make sure you are using the correct license files

```
//make sure the file path is correct.
//If license file not in the jar file, use url like file:///c:/resource/license.dat
TWaverUtil.validateLicense("/resource/license.dat");
```

- Check license information, make sure the licensed information is correct.

 Press short-cut "Ctrl+shift+t" to popup the license information dialog.

- Make sure the server side and the client side are all using the correct license file.

Network FAQ

- [Can alarm or message attachment be displayed in front of any elements?](#)
- [Can I change the flashing speed for the blinking elements?](#)
- [Can I use animated GIF for node image?](#)
- [I add KeyListener or MouseListener to Network, why it does not work?](#)

Can alarm or message attachment be displayed in front of any elements?

Yes, you can do it by this:

```
twaver.Element  
putAlarmBalloonShownOnTop(boolean alarmBalloonShownOnTop);  
public void putMessageShownOnTop(boolean messageShownOnTop);
```

Can I change the flashing speed for the blinking elements?

Yes. Please see JavaDoc of class `twaver.base.animation.BlinkingManager` and the source code of TWaver online demo.

Can I use animated GIF for node image?

Yes. TWaver support animated GIF for node image, and icon attachment. There is no any difference when you use GIF image.

Following example shows you how to create a simple network by GIF images:

```
//Add some elements using GIF image.
Node printer = new Node();
printer.setImage("/resource/gif/printer.gif");
...
Node computer = new Node();
computer.setImage("/resource/gif/computer.gif");
...
Node man = new Node();
man.setImage("/resource/gif/man.gif");
...
//Create and set some GIF icon attachments.
IconAttachmentHolder.addAttachment("Attachment1", "/resource/gif/att1.gif");
String key= TWaverConst.ELEMENT_STATE_ICON_PREFIX + "Attachment1";
man.putClientProperty(key, Boolean.TRUE);
```

Animated GIF image example



I add KeyListener or MouseListener to Network, why it does not work?

Please remember Network component is composed by toolbar, scroll pane, canvas etc. Most of the time, we believe you want add listeners on Network canvas. So you can not do this by adding a listener to Network component directly (network.addMouseListener). Please get Network canvas first then add listener on it:

network.getCanvas().add*Listener()

```
TNetwork network=new TNetwork();
network.getCanvas().addMouseListener(new MouseListener(){...});
network.getCanvas().addKeyListener(new KeyListener(){...});
...
```

Others FAQ

- [How to connect X11 window server?](#)
- [How to intercept TWaver exception?](#)
- [How to make interaction between Applet and JavaScript?](#)
- [How to set image file path?](#)
- [My web application runs fine on my Windows development machine, but when I deploy it to the Unix or Linux production server, it doesn't work. What is the problem?](#)

How to connect X11 window server?

You should enable headless for Java. Add following parameters when start your program:

```
-Djava.awt.headless=true
```


How to intercept TWaver exception?

TWaver use twaver.ErrorHandler to handler TWaver exceptions.

twaver.ErrorHandler

```
public interface ErrorHandler {
    public void handle(String errorMessage, Throwable exception);
}

//By default it is definid in twaver.TWaverUtil as follows
public static ErrorHandler errorHandler = new ErrorHandler(){
    public void handle(String errorMessage, Throwable exception) {
        if(errorMessage != null){
            System.err.println(errorMessage);
        }
        if(exception != null){
            exception.printStackTrace();
        }
    }
};
```

You can intercept with your own error handler

```
ErrorHandler myErrorHandler=new ErrorHandler(){
    public void handle(String errorMessage, Throwable exception) {
        ...
    }
};
twaver.TWaverUtil.errorHandler=myErrorHandler;
```

How to make interaction between Applet and JavaScript?

By Class 'netscape.javascript.JSObject' which is provided by JDK, applet and javascript can operate with each other. The JAVA_HOME\jre\lib\plugin.jar must be imported when program is compiled.

```
import netscape.javascript.*;
...
public class MainFrame extends JApplet {
...
//Applet operates the function 'openURL' in Javascript
private void openURL(String name){
    JSObject window = JSObject.getWindow(this);
    String code = "openURL(\"" + name + "\");";
    window.eval(code);
}
public void createServer(String name){
    Node server = new Node();
    server.setName(name);
    box.addElement(server);
}
...
}
```

```
...
<applet
    codebase = "."
    archive = "twaver.jar"
    code = "demo.MainFrame"
    width = "100%"
    height = "75%"
    align = "middle"
    MAYSCRIPT>
</applet>
...
<SCRIPT LANGUAGE="JavaScript">
<!--
function openURL(name){
    alert("hello");
}
//Javascript operates the function 'createServer' in Applet
function createServer(name){
    document.applets[0].createServer('A B C');
}
//-->
</SCRIPT>
```

About how to use the methods, you can reference the document provided by SUN Introduce in JDK1.3: <http://java.sun.com/products/plugin/1.3/docs/jsobject.html>

How to set image file path?

Use `'/c:/1.png'` or `'file:///c:/1.png'` instead `'c:/1.png'`. Please see JavaDoc of class `java.net.URL`.

My web application runs fine on my Windows development machine, but when I deploy it to the Unix or Linux production server, it doesn't work. What is the problem?

Most likely your server does not have X11 running. This is a Java (AWT/Java2D) issue, not something that is specific to JFreeChart. There is some more information at Sun's website:

<http://java.sun.com/products/java-media/2D/forDevelopers/java2dfaq.html>

Swing FAQ

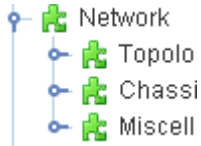
- [How to change the line style of TWaver Tree nodes?](#)
- [Why I change table column width, it does not work?](#)

How to change the line style of TWaver Tree nodes?

By default Java Swing use this as default line style:

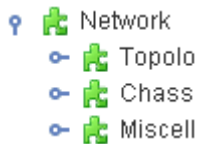
```
tree.putClientProperty("JTree.lineStyle", "Angled");
```

And it looks like below:

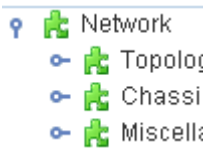


If you want remove or change the line style on tree, use this code:

```
tree.putClientProperty("JTree.lineStyle", "None");
```



```
tree.putClientProperty("JTree.lineStyle", "Horizontal");
```



Why I change table column width, it does not work?

If you get table column, set width in this way:

```
column.setWidth(200);
```

That will not work. You should change to this way:

```
table.getTableHeader().setResizingColumn(column);  
column.setWidth(200);
```

Table Tree Sheet FAQ

- [How to add tree expand listener](#)
- [How to set tree column's tooltip for tree table](#)
- [I'm using tristate tree. Can I make the parent node unselected when there is no child node selected?](#)
- [In TTableModel, what's the difference between methods `getPublishedColumn\(\)` and `getPublishedData\(\)`?](#)

How to add tree expand listener

In Java Swing, javax.swing.JTree provides following methods to listen tree expand event:

```
public void addTreeExpansionListener(TreeExpansionListener tel)
public void addTreeWillExpandListener(TreeWillExpandListener tel)
```

In TWaver, twaver.tree.TTree extends javax.swing.JTree. The tree root node represent the connected DataBox. Other tree nodes represent Elements in the DataBox. You can use following method to get the Element object of the expand event:

```
private static Element getElementFormTreExpansionEvent(TreeExpansionEvent event) {
    TreePath path = event.getPath();
    if (path != null) {
        Object comp = path.getLastPathComponent();
        if (comp instanceof ElementNode) {
            ElementNode node = ((ElementNode) comp);
            return node.getElement();
        }
    }
    return null;
}
```

Here is the whole example:

```
public static void main(String[] args) {
    TDataBox box = new TDataBox();
    Group group = new Group();
    group.setName("group");
    box.addElement(group);
    Node node = new Node();
    node.setName("node_1");
    node.setParent(group);
    box.addElement(node);
    TTree tree = new TTree(box);
    tree.addTreeExpansionListener(new TreeExpansionListener() {
        public void treeCollapsed(TreeExpansionEvent event) {
            Element element=getElementFormTreExpansionEvent(event);
            if(element!=null){
                System.out.println(element.getName()+" collapsed ");
            }
        }
        public void treeExpanded(TreeExpansionEvent event){
            Element element=getElementFormTreExpansionEvent(event);
            if(element!=null){
                System.out.println(element.getName()+" expanded ");
            }
        }
    });
    tree.addTreeWillExpandListener(new TreeWillExpandListener() {
        public void treeWillCollapse(TreeExpansionEvent event) throws ExpandVetoException {
            Element element=getElementFormTreExpansionEvent(event);
            if(element!=null){
                System.out.println(element.getName()+" will collapse ");
            }
        }
    })
}
```

```

public void treeWillExpand(TreeExpansionEvent event) throws ExpandVetoException {
    Element element=getElementFormTreExpansionEvent(event);
    if(element!=null){
        System.out.println(element.getName()+" will expand ");
    }
}
});
JFrame frame = new JFrame();
frame.setContentPane(new JScrollPane(tree));
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(500, 400);
frame.setLocationRelativeTo(null);
frame.setVisible(true);
}
private static Element getElementFormTreExpansionEvent(TreeExpansionEvent event) {
    TreePath path = event.getPath();
    if (path != null) {
        Object comp = path.getLastPathComponent();
        if (comp instanceof ElementNode) {
            ElementNode node = ((ElementNode) comp);
            return node.getElement();
        }
    }
    return null;
}

```

How to set tree column's tooltip for tree table

TTreeTable extends TElementTable, is a combination of TElementTable and TTree. TTreeTable contains a TTree which used to paint the hierarchical column. All operations of that tree column can be down by this TTree instance. Here is an example to set the tooltip of the tree column.

```
TDataBox box=new TDataBox();
TTreeTable treeTable=new TTreeTable(box);
treeTable.setElementClass(Node.class);
Generator elementToolTipTextGenerator=new Generator(){
    public Object generate(Object object) {
        String name=((Element)object).getName();
        return "tree-" + (name==null?"":name);
    }
};
treeTable.getTree().setElementToolTipTextGenerator(elementToolTipTextGenerator);
for(int i=0;i<10;i++){
    Element element=new Node();
    element.setName("node_" + i);
    box.addElement(element);
}
JFrame frame = new JFrame();
frame.setTitle("Tree table tooltip");
frame.getContentPane().add(new JScrollPane(treeTable),BorderLayout.CENTER);
frame.setSize(600, 400);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
TWaverUtil.centerWindow(frame);
frame.setVisible(true);
```

Run the code:

	ID	N
DataBox		
node_0	ad64376b7...	node
node_1	ad64376b7...	node
tree-node_0	ad64376b7...	node
node_3	ad64376b7...	node

I'm using tristate tree. Can I make the parent node unselected when there is no child node selected?

Yes. Try this code:

```
tree.setTTreeSelectionMode(TTree.CHECK_DESCENDANT_ANCESTOR_SELECTION);
tree.setEnableTristateCheckBox(true);
final TDataBox box=tree.getDataBox();
box.addElementPropertyChangeListener(new PropertyChangeListener(){
    boolean isAdjusting = false;
    public void propertyChange(PropertyChangeEvent evt) {
        if(isAdjusting || !evt.getPropertyName().equals("selected")){
            return;
        }
        final Element element = (Element)evt.getSource();
        if(!element.isSelected()){
            javax.swing.SwingUtilities.invokeLater(new Runnable(){
                public void run() {
                    isAdjusting = true;
                    Element parent = element.getParent();
                    while(parent != null){
                        if(parent.isSelected()){
                            Iterator it = parent.children();
                            boolean nonSelected = true;
                            while(it.hasNext()){
                                Element child = (Element)it.next();
                                if(child.isSelected()){
                                    nonSelected = false;
                                    break;
                                }
                            }
                        }
                        if(nonSelected){
                            box.getSelectionModel().removeSelection(parent);
                        }
                        parent = parent.getParent();
                    }
                    isAdjusting = false;
                }
            });
        }
    }
});
```

In TTableModel, what's the difference between methods `getPublishedColumn()` and `getPublishedData()`?

Developers may confused the methods `getPublishedColumn()` and `getPublishedData()`

- `getPublishedColumn()` return published data only for visible columns
- `getPublishedData()` return all published data for all columns (include visible columns and invisible columns)

Appendix

- [TWaver Element Property List](#)
- [Class ElementAttribute](#)
- [Major Interfaces in TWaver](#)
- [TWaver Property Key List](#)
- [TWaver License and Distribution Agreement](#)

TWaver Element Property List

In TWaver, there are two kind of Element properties: standard JavaBean properties and ClientProperty properties. In order to clear them, we use different method name to change the value of these properties:

- Standard JavaBean Property: set*(value)
- ClientProperty property: put*(value)

Please note the get methods all have the same name pattern: get*() or is*()


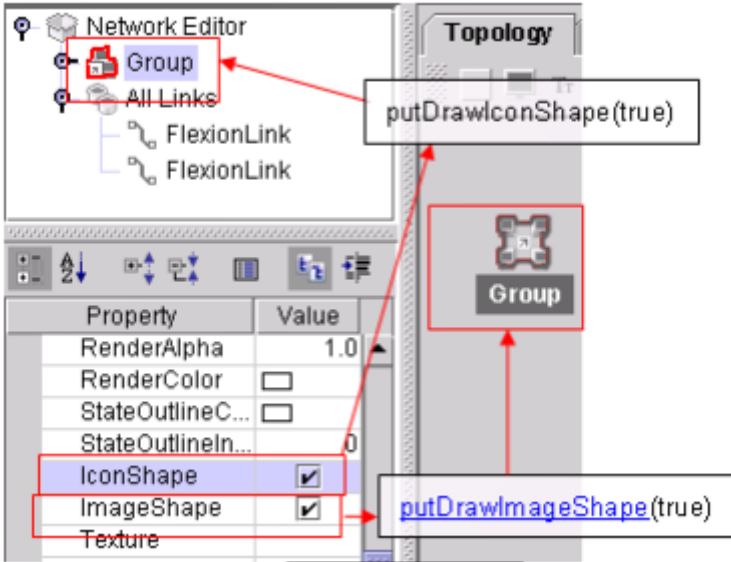
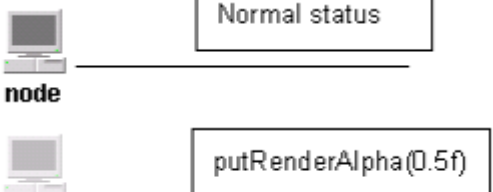

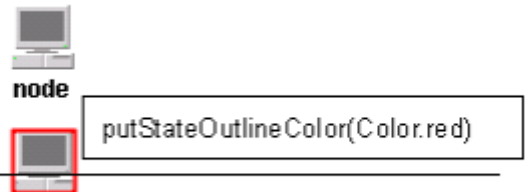
- [AbstractElement Property List](#)
- [BaseElement Property List](#)
- [Node Property List](#)
- [Group Property List](#)
- [Link Property List](#)
- [Other Nodes](#)
- [Other Links](#)


AbstractElement Property List









Method	Description
setDisplayNames [String]	Set element display name. This name will be displayed on components. It may be translated into more user friendly language.
setAlarmState [twaver.AlarmState]	<p>This class provides a representation model for generic alarms carried by a telecom object. Each alarm severity present in the alarm system, can have a number of new alarms and a number of acknowledged alarms stored. It also introduces the concept of outstanding alarms, whose number is the sum of all alarms, whether acknowledged or not. AlarmState data structure:</p> <pre> overall alarms native----new-----severity 1 total number --severity 2 total number --severity n total number acknowledged---severity 1 total number --severity 2 total number --severity n total number propagate alarm-----propagated severity </pre> <p>New alarms will change to acknowledged alarm via acknowledge action: (new alarm) -- acknowledge - (acknowledged alarm)</p>
setPropertyValues [twaver.TPropertyDescriptor,Object]	Sets new value of this element with the specified property.
setEnabledAlarmPropagationFromChildren [boolean]	Determines whether or not the children's alarm state can propagate to this element.
setLayerID [Object]	Sets the layer id of the element.
setName [String]	Set the name value.
setParent [twaver.Element]	Set new parent to this element. This element will be removed from the old parent first. If null means no parent.
setVisible [boolean]	Shows or hides this element depending on the value of parameter visible.
setIcon [String]	Set this element's icon url, that may be used to render in tree as substitute for default icon, if url is null then element's icon will be replaced by default icon
setToolTipText	Set the tooltip text value to the element.













[String]	
setSelected [boolean]	Set the element to new selection status.
setImage [String]	Set this element's image url that may be used to render in network as substitute for default image. If url is null then element's image will be replaced by default image. If url is TWaverConst.NONE_IMAGE then getImage will return null.
setUserObject [Object]	Set an user object to this element. An element object can carry any type user object for additional information. This object may bind some business or user defined information.
putLabelFont [java.awt.Font]	Set the label font
putLabelColor [java.awt.Color]	Set the label text color
putLabelVisible [boolean]	Whether label text is visible
putLabelBorder [boolean]	Whether label border is visible
putLabelUnderline [boolean]	Whether paint underline for label text
putLabelPosition [int]	<p>All available constants in TWaverConst:</p> <ul style="list-style-type: none"> • POSITION_HOTSPOT POSITION_CENTER • POSITION_TOP • POSITION_BOTTOM • POSITION_LEFT • POSITION_RIGHT • POSITION_TOPLEFT • POSITION_TOPRIGHT • POSITION_BOTTOMLEFT • POSITION_BOTTOMRIGHT • POSITION_INNER_BOTTOM • POSITION_INNER_TOP • POSITION_INNER_LEFT • POSITION_INNER_RIGHT • POSITION_INNER_TOPLEFT • POSITION_INNER_TOPRIGHT • POSITION_INNER_BOTTOMLEFT • POSITION_INNER_BOTTOMRIGHT
putLabelOrientation [int]	<p>All available constants in TWaverConst:</p> <ul style="list-style-type: none"> • LABEL_ORIENTATION_HORIZONTAL • LABEL_ORIENTATION_VERTICAL • LABEL_ORIENTATION_LEFT • LABEL_ORIENTATION_RIGHT
putLabelXOffset [int]	Label position offset in X coordinate
putLabelYOffset [int]	Label position offset in Y coordinate

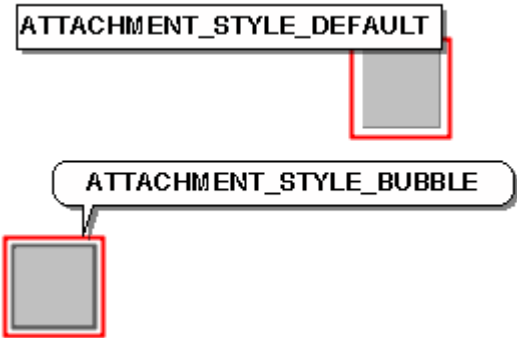
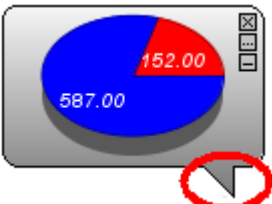
putLabelBorderStroke [String]	All available constants in TWaverConst <ul style="list-style-type: none"> • STROKE_SOLID_THINNEST • STROKE_SOLID_THIN • STROKE_SOLID_MIDDLE • STROKE_SOLID_THICK • STROKE_SOLID_THICKEST • STROKE_DOT_THINNEST • STROKE_DOT_THIN • STROKE_DOT_MIDDLE • STROKE_DOT_THICK • STROKE_DOT_THICKEST • STROKE_SQUARE_THINNEST • STROKE_SQUARE_THIN • STROKE_SQUARE_MIDDLE • STROKE_SQUARE_THICK • STROKE_SQUARE_THICKEST • STROKE_DASH_THINNEST • STROKE_DASH_THIN • STROKE_DASH_MIDDLE • STROKE_DASH_THICK • STROKE_DASH_THICKEST • STROKE_DASH_DOT_THINNEST • STROKE_DASH_DOT_THIN • STROKE_DASH_DOT_MIDDLE • STROKE_DASH_DOT_THICK • STROKE_DASH_DOT_THICKEST • STROKE_DASH_DOT_DOT_THINNEST • STROKE_DASH_DOT_DOT_THIN • STROKE_DASH_DOT_DOT_MIDDLE • STROKE_DASH_DOT_DOT_THICK • STROKE_DASH_DOT_DOT_THICKEST • STROKE_ZIGZAG_NARROWEST • STROKE_ZIGZAG_NARROW • STROKE_ZIGZAG_MIDDLE • STROKE_ZIGZAG_WIDE • STROKE_ZIGZAG_WIDEST
putLabelBorderColor [java.awt.Color]	Set label border color
putLabelUnderlineStroke [String]	Set label underline stroke style
putLabelUnderlineColor [java.awt.Color]	Set label underline color
putLabelSelectable [boolean]	Whether mouse click label can select Element. By default this is true.
putLabelHighlightable [boolean]	Whether show highlighted color for label when Element is selected.
putLabelHighlightBackground [java.awt.Color]	Set the highlighted background color when Element is selected
putLabelHighlightForeground [java.awt.Color]	Set the highlighted foreground color when Element is selected
putLabelBackground [java.awt.Color]	Set label text background color

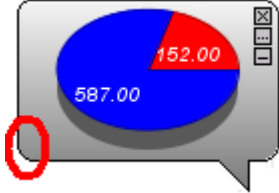
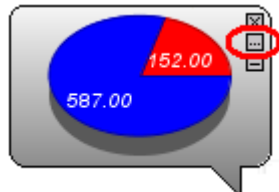
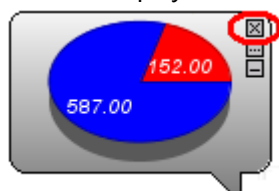
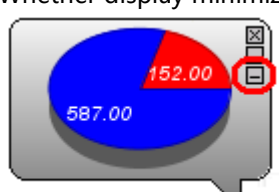
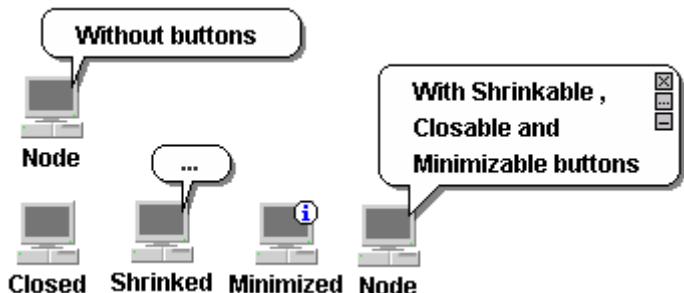
	 Node
putLabelMaxLength [int]	Set label maximal width
putDrawIconShape [boolean]	 <p>putDrawIconShape(true)</p> <p>putDrawImageShape(true)</p>
putRenderAlpha [float]	 <p>Normal status</p> <p>node</p> <p>putRenderAlpha(0.5f)</p>
putElementTreeIcon [javax.swing.Icon]	Set the icon used to paint on the tree component.
putRenderColor [java.awt.Color]	 <p>node</p> <p>putRenderColor(Color.red)</p>
putStateOutlineColor [java.awt.Color]	 <p>node</p> <p>putStateOutlineColor(Color.red)</p>
putAttachmentPosition	Set the position of Element attachments See Position

[int]	
putAttachmentOrientation [int]	<p>Set all attachments layout direction Here is all orientation definition in TWaverConst:</p> <pre> /** orientation North (up) */ public static final int ORIENTATION_NORTH = 1; /** orientation north-east (upper right) */ public static final int ORIENTATION_NORTH_EAST = 2; /** orientation east (right) */ public static final int ORIENTATION_EAST = 3; /** orientation south-east (lower right) */ public static final int ORIENTATION_SOUTH_EAST = 4; /** orientation south (down) */ public static final int ORIENTATION_SOUTH = 5; /** orientation south-west (lower left) */ public static final int ORIENTATION_SOUTH_WEST = 6; /** orientation west (left) */ public static final int ORIENTATION_WEST = 7; /** orientation north west (upper left) */ public static final int ORIENTATION_NORTH_WEST = 8; </pre>
putAttachmentXOffset [int]	Set attachments offset in X coordinate direction
putAttachmentYOffset [int]	Set attachments offset in Y coordinate direction
putAttachmentXGap [int]	Set the gap between attachments in X coordinate direction
putAttachmentYGap [int]	Set the gap between attachments in Y coordinate direction
putAlarmBalloonPosition [int]	<p>Set Element alarm balloon position See Position</p>
putAlarmBalloonDirection [int]	<p>Set Element alarm balloon direction. All available constants defined in TWaverConst :</p> <ul style="list-style-type: none"> • ATTACHMENT_DIRECTION_TOP_LEFT  • ATTACHMENT_DIRECTION_TOP_RIGHT  • ATTACHMENT_DIRECTION_BOTTOM_LEFT  • ATTACHMENT_DIRECTION_BOTTOM_RIGHT  • ATTACHMENT_DIRECTION_LEFT_TOP 

	<ul style="list-style-type: none"> • ATTACHMENT_DIRECTION_LEFT_BOTTOM  • ATTACHMENT_DIRECTION_RIGHT_TOP  • ATTACHMENT_DIRECTION_RIGHT_BOTTOM  • ATTACHMENT_DIRECTION_TOP  • ATTACHMENT_DIRECTION_BOTTOM  • ATTACHMENT_DIRECTION_LEFT  • ATTACHMENT_DIRECTION_RIGHT 
putAlarmBalloonXoffset [int]	Set alarm balloon X coordinate offset
putAlarmBalloonYoffset [int]	Set alarm balloon Y coordinate offset
putAlarmBalloonVisible [boolean]	Whether alarm balloon is visible. Default is true
putAlarmBalloonAlpha [float]	 putAlarmBalloonAlpha(0.5f)
putAlarmBalloonTextFont [java.awt.Font]	Set alarm balloon text font
putAlarmBalloonTextColor [java.awt.Color]	Set alarm balloon text color
putAlarmBalloonTextBlinkable [boolean]	<p>Whether alarm balloon text is blinking, default is true. Note: in order to make alarm balloon blinking, you need to useblinking rule for Network component:</p> <pre> network.setBlinkingRule(new BlinkingRule{ public boolean isBodyBlinking(Element element) { return false; } public boolean isOutlineBlinking(Element element) { // blink when element has propagate severity AlarmState state=element.getAlarmState(); return state.getPropagateSeverity() != null; } }); </pre> <p>Please see demo.network.topo.blinking.BlinkingDemo.java</p>
putAlarmBalloonOutlineColor	Set alarm balloon outline color. Default is black color.

[java.awt.Color]	
putAlarmBalloonShadowColor [java.awt.Color]	Set alarm balloon shadow color
putAlarmBalloonShadowOffset [int]	Set alarm balloon shadow offset. Default value is 3
putTextureFactory [String]	<p>Set Element fill texture pattern. All available texture defined in TWaverConst:</p> <ul style="list-style-type: none"> • TEXTURE_STIPPLE_LOOSE  • TEXTURE_STIPPLE_MIDDLE  • TEXTURE_STIPPLE_DENSE  • TEXTURE_GRID_DIAMOND  • TEXTURE_GRID_SQUARE  • TEXTURE_GRID_TRIANGLE  • TEXTURE_LINE_CROSS  • TEXTURE_LINE_VERTICAL  • TEXTURE_LINE_HORIZONTAL  • TEXTURE_LINE_DIAGONAL  • TEXTURE_LINE_ANTIDIAGONAL 
putBorderAntialias [boolean]	Whether use antialias to paint border. Default is true. Note: Border is only painted when Element is selected
putBorderVisible [boolean]	Whether show border when Element is selected
putBorderStroke [String]	Set the border stroke Note: Border is only painted when Element is selected
putBorderColor [java.awt.Color]	Set border color Note: Border is only painted when Element is selected
putBorderInsets [int]	<p>Set border insets to Element bounds.</p> <div style="background-color: yellow; padding: 5px; border: 1px solid black;">  Border is only painted when Element is selected. This property will be ignored for twaver.Group </div>
putStateOutlineInsets [int]	Set state outline insets to Element bounds
putMessageContent [String]	<p>Set message content Note: when the MessageComponent is TWaverConsts. MESSAGE_COMPONENT_LABEL, the message content support HTML format; when MessageComponent is TWaverConsts.MESSAGE_COMPONENT_TEXTPANE, message content support line wrap but do not support HTML format.</p>
putMessageStyle	Set message style. The default style is:

[int]	<p>TWaverConst.ATTACHMENT_STYLE_BUBBLE All available constants defines in TWaverConst:</p> 
putMessageComponent [int]	<p>Set message component type. All available constants in TWaverConsts:</p> <ul style="list-style-type: none"> • MESSAGE_COMPONENT_LABEL • MESSAGE_COMPONENT_TEXTPANE <p>Note:when the MessageComponent is TWaverConsts. MESSAGE_COMPONENT_LABEL, the message content support HTML format; when MessageComponent is TWaverConsts.MESSAGE_COMPONENT_TEXTPANE, message content support line wrap but do not support HTML format.</p>
putMessageWidth [int]	<p>Set the message component width limit. Default is -1, no limit. Note:</p> <ul style="list-style-type: none"> • if you set a value ≤ 0, then it means no width limit • When the message component is TWaverConsts. MESSAGE_COMPONENT_LABEL, if this value less than the actual text width, the text outside will invisible; when message component is TWaverConsts.MESSAGE_COMPONENT_TEXTPANE, the text will be line wrapped automatically.
putMessageHeight [int]	<p>Set message height limit. The default value is -1, no height limit. Note:</p> <ul style="list-style-type: none"> • When height limit ≤ 0, means no limit • When actual text height bigger than the height limit, the outside text will invisible.
putMessageTail [int]	<p>Set the length of the message tail. Default value is 15.</p> 
putMessageArc [int]	<p>Set message round arc diameter. Default value is 15</p>

	
putMessageShrinkable [boolean]	Whether display shrink button. Default is no. 
putMessageClosable [boolean]	Whether display close button. Default is no 
putMessageMinimizable [boolean]	Whether display minimize button. Default is no 
	Comprehensive example 
putMessageAutoAdjustDirection [boolean]	Whether enable automatic adjust message tail direction to make sure the message is always visible. Note: The automatic adjustment does not work during Element drag/drop
putMessagePosition [int]	Set message position See Position
putMessageDirection [int]	Set message direction See Direction
putMessageFont [java.awt.Font]	Set message font

putMessageForeground [java.awt.Color]	Set message foreground color
putMessageBackground [java.awt.Color]	Set message background color
putMessageOpaque [boolean]	Whether message is opaque. Default is true. Note: when this value set to true, the message background will be ignored.
putMessageXOffset [int]	Set message position offset in X coordinate
putMessageYOffset [int]	Set message position offset in Y coordinate
putMessageXGap [int]	Set the gap in X coordinate for messages when Element has more than one message.
putMessageYGap [int]	Set the gap in Y coordinate for messages when Element has more than one message.
putMessageMinimizedIcon [String]	Set message icon when it is minimized
putMessageShadowVisible [boolean]	Whether show message shadow.
putMessageShadowColor [java.awt.Color]	Set message shadow color. Default is Color.gray
putMessageBorderColor [java.awt.Color]	Set message border color. Default is Color.black
putMessageBorderVisible [boolean]	Whether message border is visible. Default is visible.
putMessageBorderStroke [String]	Set message border stroke. See StrokeConst

BaseElement Property List

Method	Description
setCenterLocation [double double]	Sets the element's bounds center point.
setLocation [int int]	Set new location of the element. Location determine the element position of left top point. Use this method to move an element. Please note that element location is the left-top point of the element.
setLocation [java.awt.Point]	
setLocation [java.awt.geom.Point2D\$Double]	
setLocation [double double]	
putBodyColor [java.awt.Color]	Set element body color. Note: if Element has image or customDraw is true, then this property will be ignored. Use property RenderColor instead
putBodyRaised [boolean]	Whether paint raised border. Default is false.
putBodyFill [boolean]	Whether fill Element body. Default is true.
putCustomDraw [boolean]	Whether enable CustomDraw. Default is false.
putCustomDrawShapeFactory [int]	Set CustomDraw shape. Default is SHAPE_CIRCLE All available shapes defined in TWaverConst: <ul style="list-style-type: none"> • SHAPE_CIRCLE • SHAPE_RECTANGLE • SHAPE_ROUND_RECTANGLE • SHAPE_ROUND_RECTANGLE_HALF • SHAPE_ROUND_RECTANGLE_QUARTER • SHAPE_ROUND_VAULT • SHAPE_ROUND_VAULT_HALF • SHAPE_ROUND_VAULT_QUARTER • SHAPE_DIAMOND • SHAPE_TRIANGLE • SHAPE_STAR
putCustomDrawDefaultBorder [boolean]	Whether use default selection border. Default is false. If this value is true, then use the default selection border (a rectangle); if false, then use the actual shape as the selection border.
putCustomDrawAntialias [boolean]	Whether enable antialias when CustomDraw enabled. Default is true.
putCustomDrawFill [boolean]	Whether fill Element when CustomDraw enabled. Default is true

putCustomDrawFill3D [boolean]	Whether use 3D fill when CustomDraw enabled, default is false. Note: this property works only when CustomDrawFill is true
putCustomDrawFillColor [java.awt.Color]	Set the fill color for CustomDraw
putCustomDrawOutline [boolean]	Whether draw outline when CustomDraw enabled
putCustomDrawOutline3D [boolean]	Whether draw 3D outline when CustomDraw enabled
putCustomDrawOutlineColor [java.awt.Color]	Set outline color for CustomDraw
putCustomDrawOutlineStroke [String]	Set outline stroke when CustomDraw enabled See AbstractElement Property List#StrokeConst
putCustomDrawGradient [boolean]	Whether enable gradient fill when CustomDraw enabled, default is true
putCustomDrawGradientFactory [int]	Set the gradient fill pattern for CustomDraw. All available gradients defined in TWaverConst: <div> <pre> /** define linear gradient from south west */ public static final int GRADIENT_LINE_SW = 1; /** define linear gradient from south east */ public static final int GRADIENT_LINE_SE = 2; /** define linear gradient from north west */ public static final int GRADIENT_LINE_NW = 3; /** define linear gradient from north east */ public static final int GRADIENT_LINE_NE = 4; /** define linear gradient from north */ public static final int GRADIENT_LINE_N = 5; /** define linear gradient from south */ public static final int GRADIENT_LINE_S = 6; /** define linear gradient from west */ public static final int GRADIENT_LINE_W = 7; /** define linear gradient from east */ public static final int GRADIENT_LINE_E = 8; /** define radial gradient from center */ public static final int GRADIENT_RADIAL_C = 10; /** define radial gradient from south west */ public static final int GRADIENT_RADIAL_SW = 11; /** define radial gradient from south east */ public static final int GRADIENT_RADIAL_SE = 12; /** define radial gradient from north west */ public static final int GRADIENT_RADIAL_NW = 13; /** define radial gradient from north east */ public static final int GRADIENT_RADIAL_NE = 14; /** define radial gradient from north */ public static final int GRADIENT_RADIAL_N = 15; /** define radial gradient from south */ public static final int GRADIENT_RADIAL_S = 16; /** define radial gradient from west */ public static final int GRADIENT_RADIAL_W = 17; /** define radial gradient from east */ public static final int GRADIENT_RADIAL_E = 18; /** define extend gradient at horizontal*/ public static final int GRADIENT_EXTEND_HORIZONTAL = 19; </pre> </div>

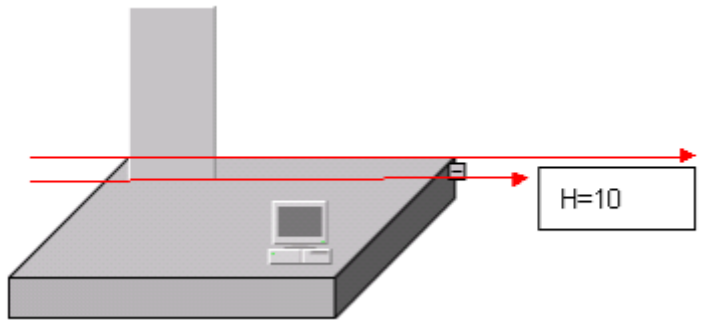



	<pre> /** label extend gradient at vertical*/ public final static int GRADIENT_EXTEND_VERTICAL = 20; /** label extend gradient at diagonal*/ public final static int GRADIENT_EXTEND_DIAGONAL = 21; /** label extend gradient at antidiagonal*/ public final static int GRADIENT_EXTEND_ANTIDIAGONAL = 22; /** define extend gradient at north*/ public static final int GRADIENT_EXTEND_N = 23; /** label extend gradient at south*/ public final static int GRADIENT_EXTEND_S = 24; /** label extend gradient at west*/ public final static int GRADIENT_EXTEND_W = 25; /** label extend gradient at east*/ public final static int GRADIENT_EXTEND_E = 26; </pre>
putCustomDrawGradientColor [java.awt.Color]	Set fill gradient color for CustomDraw enabled Element

Node Property List






Method	Description
putDrawImageShape [boolean]	Whether use image shape as the selection border. By default the selection border is a rectangle. See DrawIconShape

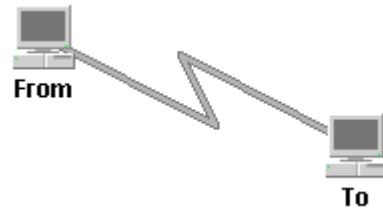
Group Property List

Method	Description
setExpand [boolean]	Whether expand Group. Default is true
setGroupType [int]	<p>Set group shape type. The default value is TWaverConst.RECTANGLE All available group types defined in TWaverConst:</p> <pre> /** rectangle group type */ public final static int GROUP_TYPE_RECTANGLE = 1; /** ellipse group type */ public final static int GROUP_TYPE_ELLIPSE = 2; /** parallelogram group type */ public final static int GROUP_TYPE_PARALLELOGRAM = 3; /** round group type */ public final static int GROUP_TYPE_ROUND = 4; /** round rectangle group type */ public final static int GROUP_TYPE_ROUND_RECTANGLE = 5; </pre>
putGroupOutline [boolean]	whether show outline
putGroupOutlineStroke [String]	See StrokeConst
putGroupFill [boolean]	whether fill group body
putGroupOpaque [boolean]	Whether group background is opaque, default is translucent
putGroupOutlineColor [java.awt.Color]	Set the Group outline color
putGroupFillColor [java.awt.Color]	Set the group fill color
putGroupChildrenOutcrop [int]	Set group children out crop value (see picture). The default value is -1, that means all children will completely covered by Group shape: putGroupChildrenOutcrop(10)

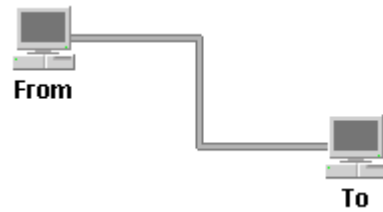
	<p>H=10</p> 
putGroupInsets [java.awt.Insets]	Set the gap for Group and its children, default value is: Insets(3, 3, 3, 3)
putGroupHandlerExpandIcon [String]	Set the group handler expand icon. Default value is "  "
putGroupHandlerCloseIcon [String]	Set the group handler close icon. Default value is "  "
putGroupHandlerEmptyIcon [String]	Set the group handler icon when group is empty. Default value is "  "
putGroupDoubleClickEnabled [boolean]	Whether allow close/expand group when double click it. Default is true.
putGroupAngle [int]	Set group angle. Default is 30. Note: this property only work for group with type Parallelogram
putGroup3D [boolean]	Whether paint group in 3D. Default is true
putGroupAntialias [boolean]	Whether enable antialias when paint Group. Default is true.
putGroupDeep [int]	The deep of Group. Default value is 20.
putGroupHandlerVisible [boolean]	Whether show handler icon. Default is true.
putGroupHandlerPosition [int]	Position of Group handler. See Position
putGroupHandlerXoffset [int]	Group handler offset in X coordinate
putGroupHandlerYoffset [int]	Group handler offset in Ycoordinate
putGroupGradient [boolean]	Whether use gradient fill for Group. Default is false.
putGroupGradientColor [java.awt.Color]	Set the fill gradient color for Group.
putGroupGradientFactory [int]	Set fill gradient type for Group.

Link Property List

Method	Description
setFrom [twaver.Node]	Node th Link start from
setTo [twaver.Node]	Node the Link end to
setLinkType [int]	<p>Link type. Default value is: TWaverConst. LINK_TYPE_STRAIGHT All available link type defined in TWaverConst:</p> <ul style="list-style-type: none"> public final static int LINK_TYPE_PARALLEL = 0;  <ul style="list-style-type: none"> public final static int LINK_TYPE_STRAIGHT = 1;  <ul style="list-style-type: none"> public final static int LINK_TYPE_FLEXIONAL = 2;  <ul style="list-style-type: none"> public final static int LINK_TYPE_ORTHOGONAL = 3;  <ul style="list-style-type: none"> public final static int LINK_TYPE_CURVOUS = 4;  <ul style="list-style-type: none"> public final static int LINK_TYPE_ZIGZAG = 5;



- `public final static int LINK_TYPE_XSPLIT = 6;`



- `public final static int LINK_TYPE_YSPLIT = 7;`



- `public final static int LINK_TYPE_TOP = 8;`



- `public final static int LINK_TYPE_BOTTOM = 9;`


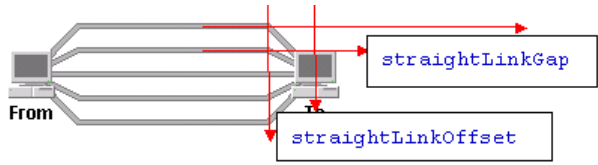
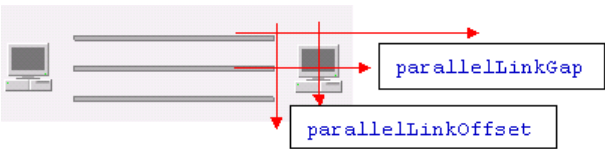


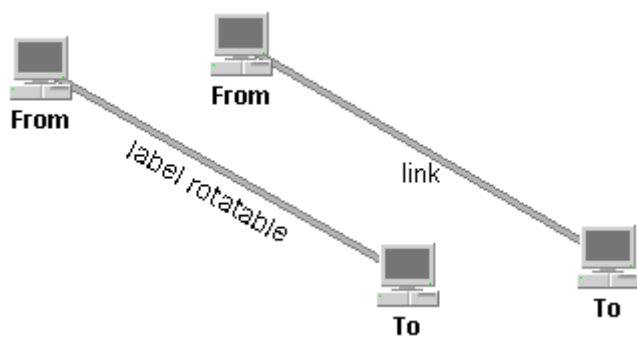
- `public final static int LINK_TYPE_LEFT = 10;`



- `public final static int LINK_TYPE_RIGHT = 11;`

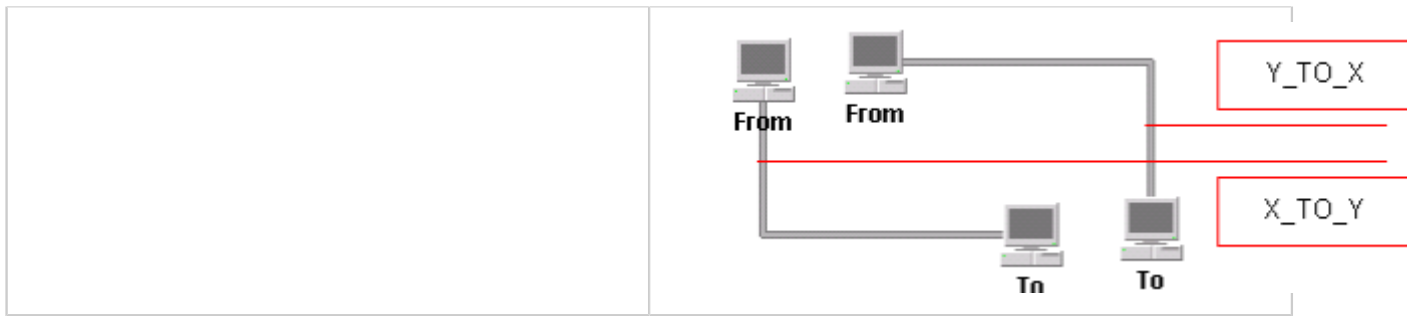


linkBundleCompact	<p>When have more links between two nodes, whether display in an impacted mode. Default value is false. Impacted mode:</p>  <p>Default mode:</p> 
straightLinkGap	<p>See picture below. Default value is 16</p> 
straightLinkOffset	<p>Default value is 32.</p>
parallelLinkGap	<p>Default value is 16.</p> 
parallelLinkOffset	<p>Default value is 20</p>
setCenterLocation [double double]	<p>do nothing now</p>
setBundleExpand [boolean]	<p>Set the bundle expand status</p>
putLinkStyle [String]	<p>Set Link style All available values in TWaverConst:</p> <ul style="list-style-type: none"> • LINK_STYLE_SOLID • LINK_STYLE_DASH • LINK_STYLE_CHAIN • LINK_STYLE_DOT • LINK_STYLE_ZIGZAG
putLinkColor [java.awt.Color]	<p>Set link color. Default is Color.gray</p>
putLinkOutlineColor [java.awt.Color]	<p>Set link outline color. Defaut value is color "#7D7D7DFF"</p>
putLinkOutlineWidth [int]	<p>Set link outline width.</p>
putLinkWidth [int]	<p>Set link width</p>
putLinkExtend [int]	<p>Set link flexion point extend value</p>

putLinkProportion [double]	Set link flexion point proportion. Value is from 0 to 1
putLinkFlowingWidth [int]	Set link flowing width
putLinkFromArrow [boolean]	Whether show link from arrow, default false.
putLinkToArrow [boolean]	Whether show link to arrow, default false.
putLinkFlowing [boolean]	Whether show link flowing. Default false.
putLinkFlowingConverse [boolean]	Whether reverse the link flowing direction. Default direction is "from-to". Default value of this property is false.
putLinkBlinking [boolean]	Whether link is blinking.
putLinkBlinkingColor [java.awt.Color]	Set the link blinking color.
putLinkFlowingColor [java.awt.Color]	Set the link flowing color
putLinkBundleExpand [boolean]	<p>This flag used to determine when more than one links connected between two Nodes, expand them or bundle them by default. The default value is true. Use this set it to false:</p> <div> <p>Object</p> <pre>k=TWaverConst.PROPERTYNAME_LINK_BUNDLE_EXPAND; TUIManager.registerDefault(k, Boolean.FALSE)</pre> </div>
putLink3D [boolean]	Whether paint 3D link
putLinkLabelRotatable [boolean]	<p>Link label whether rotated with link</p> 
putLinkAntialias [boolean]	Whether enable antialias when paint
putLinkFromArrowStyle [int]	Set the from arrow style. All available definitions in TWaverConst:

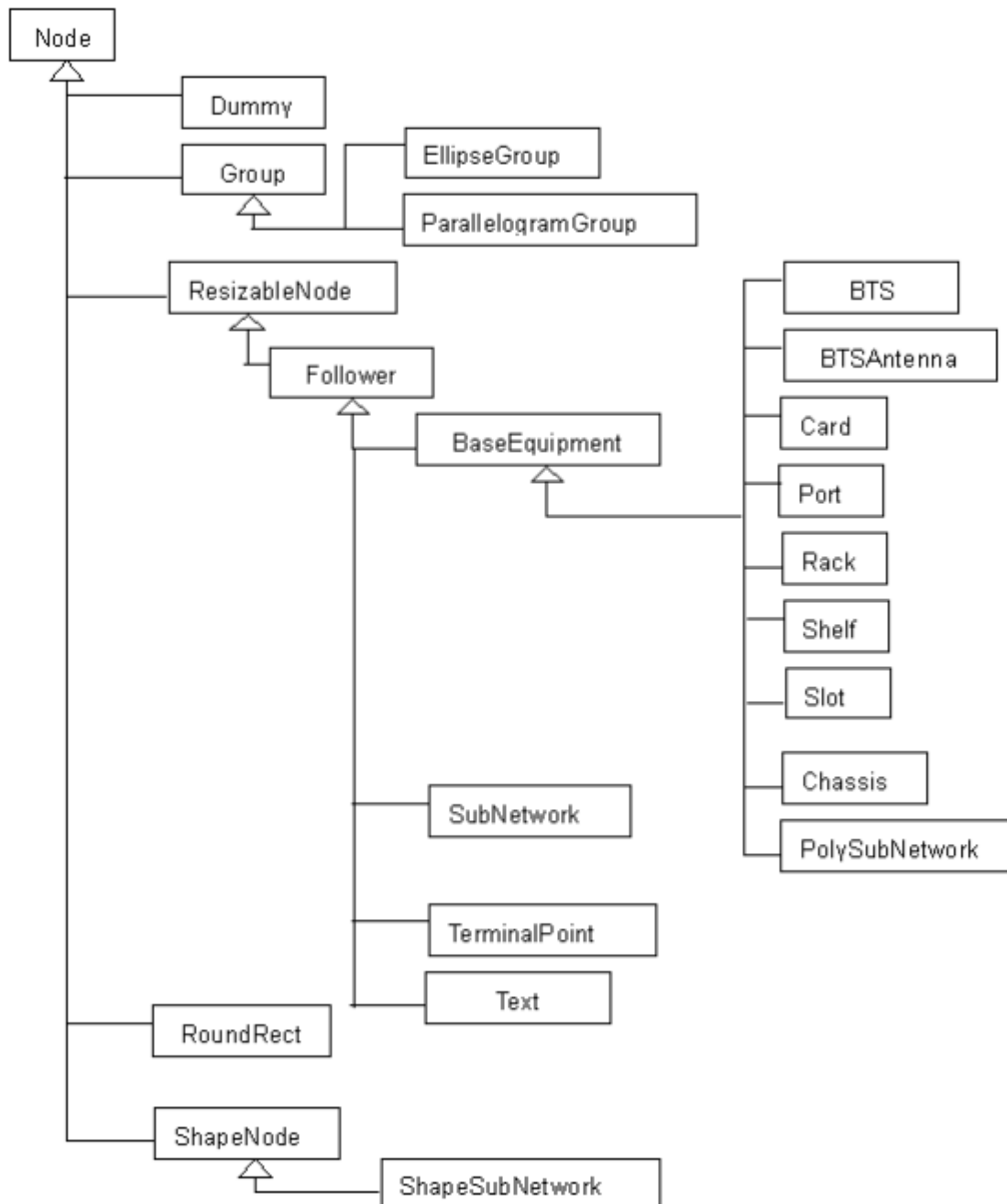
	<pre> /** standard small arrow type */ public final static int ARROW_STANDARD_SMALL = 1; /** standard arrow type */ public final static int ARROW_STANDARD = 2; /** standard great arrow type */ public final static int ARROW_STANDARD_GREAT = 3; /** delta small arrow type */ public final static int ARROW_DELTA_SMALL = 4; /** delta arrow type */ public final static int ARROW_DELTA = 5; /** delta great arrow type */ public final static int ARROW_DELTA_GREAT = 6; /** diamond small arrow type */ public final static int ARROW_DIAMOND_SMALL = 7; /** diamond arrow type */ public final static int ARROW_DIAMOND = 8; /** diamond great arrow type */ public final static int ARROW_DIAMOND_GREAT = 9; /** short small arrow type */ public final static int ARROW_SHORT_SMALL = 10; /** short arrow type */ public final static int ARROW_SHORT = 11; /** short great arrow type */ public final static int ARROW_SHORT_GREAT = 12; /** circle small arrow type */ public final static int ARROW_CIRCLE_SMALL = 13; /** circle arrow type */ public final static int ARROW_CIRCLE = 14; /** circle great arrow type */ public final static int ARROW_CIRCLE_GREAT = 15; /** star small arrow type */ public final static int ARROW_STAR_SMALL = 16; /** star arrow type */ public final static int ARROW_STAR = 17; /** star great arrow type */ public final static int ARROW_STAR_GREAT = 18; /** rectangle small arrow type */ public final static int ARROW_RECTANGLE_SMALL = 19; /** rectangle arrow type */ public final static int ARROW_RECTANGLE = 20; /** rectangle great arrow type */ public final static int ARROW_RECTANGLE_GREAT = 21; </pre>
putLinkToArrowStyle [int]	Set to arrow style See ArrowStyle
putLinkFromArrowColor [java.awt.Color]	Set from arrow color
putLinkToArrowColor [java.awt.Color]	Set to arrow color
putLinkFromArrowOutline [boolean]	Whether paint outline for start arrow. Default is true
putLinkToArrowOutline [boolean]	Whether paint outline for end arrow. Default is true

putLinkFromArrowOutlineColor [java.awt.Color]	Set link start arrow outline color. Default is black.
putLinkToArrowOutlineColor [java.awt.Color]	Set link end arrow outline color. Default is black.
putLinkHollow [boolean]	Whether the link body is hollow. Default is false.
putLinkHandlerPosition [int]	Set link bundle handler position See Position
putLinkHandlerXOffset [int]	Set link bundle handler offset in X coordinate
putLinkHandlerYOffset [int]	Set link bundle handler offset in Y coordinate
putLinkFromArrowXOffset [int]	Set start arrow offset in X coordinate
putLinkFromArrowYOffset [int]	Set start arrow offset in Y coordinate
putLinkToArrowXOffset [int]	Set end arrow offset in X coordinate
putLinkToArrowYOffset [int]	Set end arrow offset in Y coordinate
putLinkToPosition [int]	Set end point position See Position
putLinkFromPosition [int]	Set start point position See Position
putLinkToXOffset [int]	Set link end point offset in X coordinate
putLinkToYOffset [int]	Set link end point offset in Y coordinate
putLinkFromXOffset [int]	Set link start point offset in X coordinate
putLinkFromYOffset [int]	Set link start point offset in Y coordinate
putLinkOrthogonalDirection [twaver.base.OrthogonalLinkDirectionType]	Set orthogonal direction. Default value is: X_TO_Y Other definations: <ul style="list-style-type: none"> twaver.base.OrthogonalLinkDirectionType.Y_TO_X twaver.base.OrthogonalLinkDirectionType.X_TO_Y For example:



Other Nodes

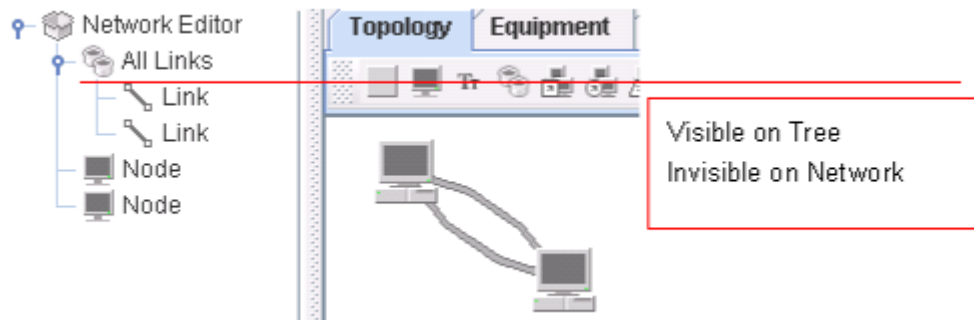
Extend relationship



Dummy

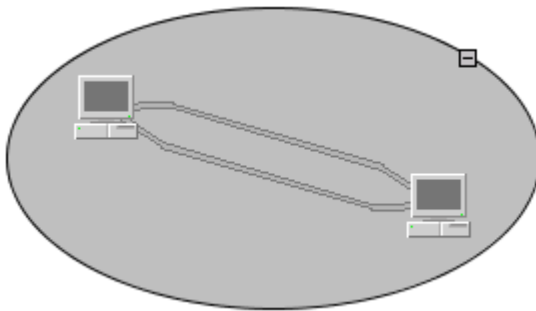
Dummy is an network-invisible container element. It normally used to represent an abstract, invisible container or group which may contains other visible elements. Dummy is invisible on network but visible on other components like tree, table, property sheet etc.

Things like "all links", "fault module", "my routers" in the OSS application can be represented by Dummy element. You can use Dummy object to reorganize the tree structure and avoid to affect network. Add children elements into the Dummy like regular element.



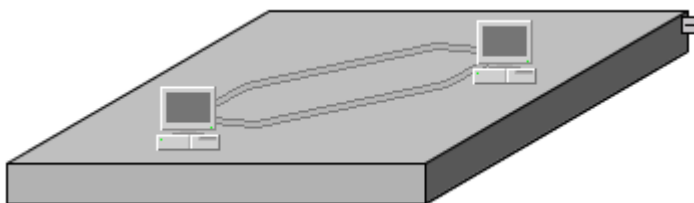
EllipseGroup

This class defines an ellipse shape group. When the group expanded, it shows all children elements in an ellipse which can enclose all elements.



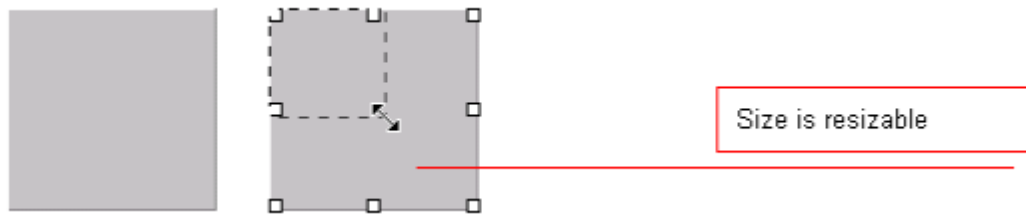
ParallelogramGroup

This class defines a group with a parallelogram shape. This group also provides parameters for the parallelogram angle and deep values. If the deep value is bigger than 0, a 3d parallelogram will be used to paint this group.



ResizableNode

This is a resizable node. This node implements interface Resizable and provides the APIs to resize this node size. Use this node to replace default node if you want user can resize the node on network canvas by mouse or API. Please note that if you set an image for this resizable node, the image will be stretched to fill the element size.



Follower

This class defines a follower node. A follower node can be specified a node as its 'host', when the host node moved on network component, the follower will stick itself on the host node and moved together. Besides, the follower can be moved separately on network canvas freely. Nodes can be a follower each other, or followed in turn as a line or a ring. In these cases the related nodes will be stucked together and moved together on network canvas.



BaseEquipment

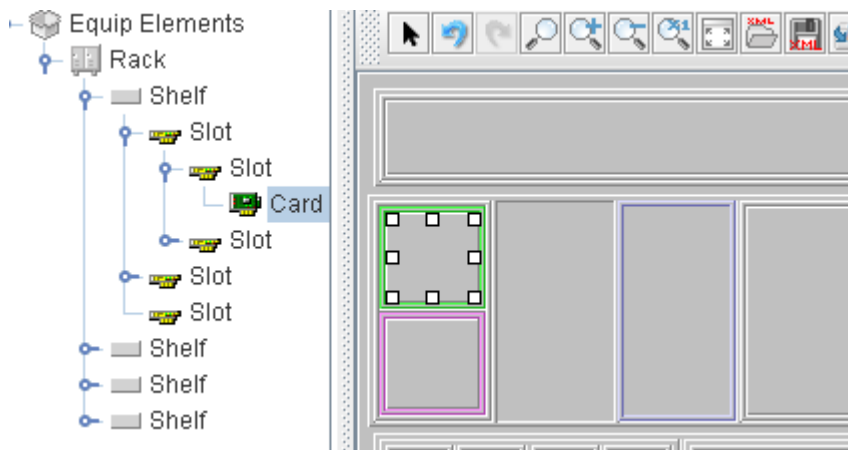
This class is a basic implementation for interface Equipment. All equipment elements are subclasses of this class. This class extends from Follower so all equipment elements are able to follow its parent on moving.

BTS & BTSAntenna



Card

This is a class defines the telecom objects that play the role of cards of telecom equipment. A card can be either contained in a rack, shelf or in a slot. This card is extends from BaseEquipment, it can carry alarm state information.



Port

This class models a port of the elements, which are telecom equipment connectors. As an equipment element, Port class extends BaseEquipment and defines its graphic representation. Basically, Port is a very simple class used to represent all kinds of equipment connectors. TWaver does not define more rules or limitations for how to use or layout the ports. Developers can use images or subclasses to extend more specific ports for their OSS applications.

So far, the port does not define any new or specific method, but this class may supports more features in the future.

Rack

This class models an equipment rack element, which is a physical rack for telecom equipment. As an equipment element, Rack class extends BaseEquipment and defines its graphic representation. Basically, Rack is a very simple class used to represent all kinds of equipment holders. TWaver does not define more rules or limitations for how to use or layout the racks. Developers can use images or subclasses to extend more specific racks for their OSS applications.

So far, the rack does not define any new or specific method, but this class may supports more features in the future.

Shelf

This class models an equipment shelf element, which is a physical shelf for telecom equipment. As an equipment element, Shelf class extends BaseEquipment and defines its graphic representation. Basically, Shelf is a very simple class used to represent all kinds of equipment holders. TWaver does not define more rules or limitations for how to use or layout the Shelves. Developers can use images or subclasses to extend more specific Shelves for their OSS applications.

Slot

SubNetwork

Subnetwork is a predefined managed object that is used to represent a network layer that all resources is layered logically or geographically. Subnetwork looks just like a normal node in the located layer. But the default behaviour is that you can double clic the subnetwork to drill down into the deeper layer to get in a new map environment. The background object of the subnetwork will be used to paint as the current background of this network component.

Subnetwork can be nested. Subnetwork has a background object as its background. It can be bitmap image, vector shape, single color or texture image etc. More information about background please see Background.

Subnetwork has a datasource property. This string property defines where to load all elements data of this layer. Use this datasource to enable lazy loading ability of this subnetwork. Only when the first time this subnetwork is drill down, the data will be loaded from the specified xml datasource. After that, the data loaded flag will be set to true so data will not be loaded again. Of course, you can set the data loaded flag to false to reload the data when this subnetwork is drilled down next time.



Chassis

This class defines an telecom equipment chassis.

What Is a Chassis

The equipment chassis is a rigid sheet-metal structure that houses all of the equipment hardware components. For example, some router has the front and rear chassis.

In TWaver, chassis defines as a subclass of SubNetwork to gain the drill-down ability. Normally you can create a chassis object and set its parent to a Node instance to present the equipment chassis of this equipment. You can double click Node on the network component to drill down to the chassis view.

Please note that chassis itself is an invisible element. It is just an abstract 'container' for all equipment components of the telecom equipment. Normally you need to add Rack or Card instances into the chassis to display the detail structure of this equipment.

PolySubNetwork

Poly subnetwork is a special subnetwork, it can be specified a shape object as the subnetwork's painting shape. You can define a complex geometry shape for the subnetwork instead of a bitmap image. For example, you can define a shape for a region or a country, and assign this shape to a poly subnetwork to represent the region or country.

Poly subnetwork save two shape objects: base shape and transformed shape. The base shape is a shape used to describe what shape the element looks like. The transformed shape is the transformed shape by base shape according to current element location. For example, you can specify a rectangle (0,0,100,100) for the subnetwork. When the subnetwork moved to location (100,100), then the base rectangle is the same but the transformed shape is a new rectangle with value (100,100,100,100).

Poly subnetwork defines a lot of methods to customize how to draw and fill the subnetwork shape.



TerminalPoint

This class defines a terminal point element. Terminal point defines as the termination of a link. This class does not define any extra properties and methods currently.



Text

this is a text element

this is a text element too

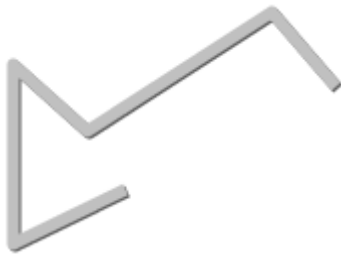
RoundRect

This class defines a node in round rectangle shape. No special properties or methods extended in this class.

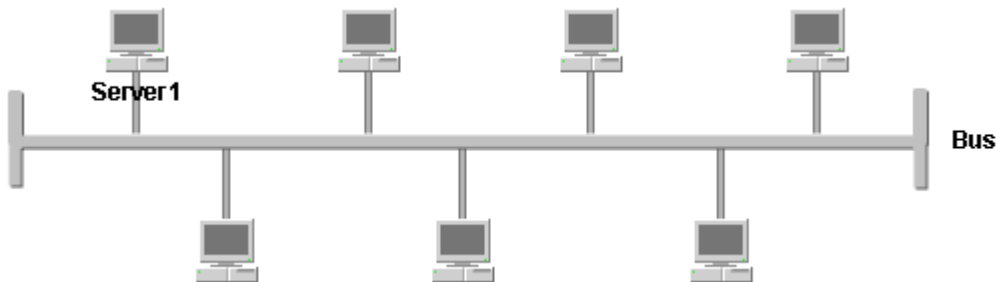
ShapeNode

This class defines a node comes with a set of control points, the points will be used to create a shape painted on network component to represent the node. Subclasses can override method createShape to return customized shape.

Line ShapeNode



Orthogonal ShapeNode

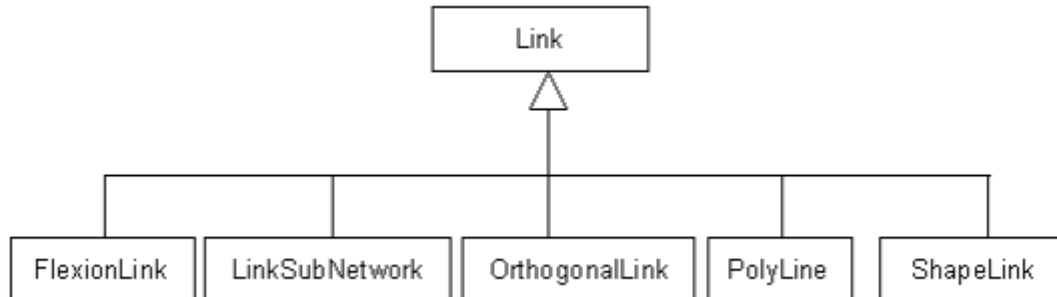


ShapeSubNetwork



Other Links

Link Structure



FlexionLink

This type of Link has two flexional points close to the two terminal points of it

LinkSubNetwork

A link extends TSubNetwork so it support data drill-down.

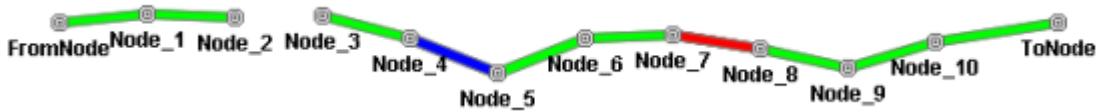
OrthogonalLink

A link with an orthogonal path.

PolyLine

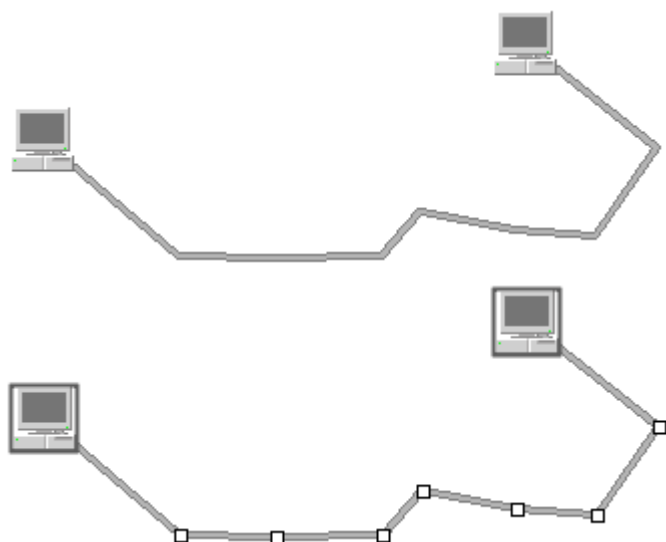
A special link element is combined by a set of straight lines. It can contain multiple branches.

Note: PolyLine has no start point and end point, that is, `line.getFrom()` and `link.getTo()` always null.



ShapeLink

This is a link contains a set of points



Class ElementAttribute

- [ElementAttribute Overview](#)
- [Register ElementAttribute](#)
- [Element Attribute Property Key](#)
- [More Settings of Element Attribute](#)
- [An Example for Using Element Attribute](#)

ElementAttribute Overview

In TWaver, class `twaver.ElementAttribute` defines a property of `Element`.

When we need to show `Element` in `PropertySheet` or table, we need to show the properties of the `Element` such as name, id, description etc. We need to define how to display these properties in components. Class `ElementAttribute` is used to describe all information about a property. It normally often used by component `TElementTable` and `TPropertySheet`.

For example, if we want define a "name" property, and connect this property to `Element.getName()` JavaBean property. we want display this property name with "Name", we want display this property in table cell with blue background color, we want put it into the "basic" property category, the preferred width of this property in table cell is "200"we can use following code to make it:

```
//define an "name" attribute
ElementAttribute nameAttribute = new ElementAttribute();
nameAttribute.setClientProperty(false); //See#PropertyKey
nameAttribute.setName("name"); //bind property key
nameAttribute.setDisplayName("Name"); //set display name
nameAttribute.setEditable(false); //set not editable
nameAttribute.setFillColor(Color.blue); //set background color in table components
nameAttribute.setWidth(200); //set cell width in table components

Category.registerCategory("basic", "BASIC", true); //register a new category
nameAttribute.setCategoryName("basic"); //put it into the new category.
```

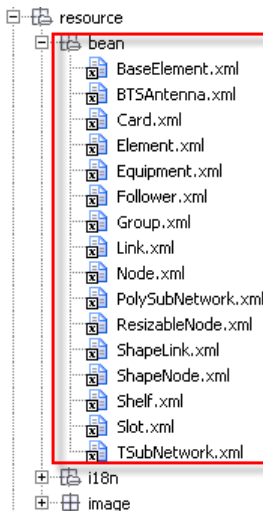
This class `ElementAttribute` encapsulates an element property information. The element property can be a standard JavaBeans property or a TWaver `ClientProperty` property.

Register ElementAttribute

When you defined an element attribute, you need a way to register this property to TWaver so PropertySheet or ElementTable are able to know it. You can do this by XML or API.

Using Resource Agent to Auto-Configuration

Every predefined Element in TWaver has a default property configuration file locate folder twaver.jar/resource/bean/.



TWaver using the class name to find the config file. So you can put property config file in that folder to auto-configure, or relocate the resource to your own resource folder and put all your resources in that new folder. Please note that all property files of Element should in bean folder (see above picture). The config file must has ".xml" extend name and match the class name. For example, for class "MyNode.class", a file with name "MyNode.xml" must be put in bean folder to let TWaver know all predefined properties of class MyNode. For more information, please see [Using Resource Agent](#).

Using API Register Element Attributes

Global Register

Register the element attribute into TWaver in global scope. In this case, all TWaver components can see it.

For example, we have an Element class Person, we want remove all TWaver default properties and register some new defined attributes. We can use following code to do it:

```
List attributes = new ArrayList();
ElementAttribute nameAttribute = new ElementAttribute();
...//define details for this attribute.
attributes.add(nameAttribute);
...//may add more attributes into the list.

//use this method to remove the TWaver default defined attributes for class Person
TWaverUtil.registerBeanInfoWithoutDefault(Person.class);
//register attributes for class Person.
TWaverUtil.registerBeanInfo(Person.class, attributes);

//now we can use it in TWaver components.
TElementTable table=new TElementTable()
```

```
...
//set this element table data's class type to Person
//so all defined attributes will displayed as columns in table
table.setElementClass(Person.class);
```

Local Register

Register the element attribute into a TWaver component so only this component can see it.

For the above example, we can change it to follow:

```
List attributes = new ArrayList();
ElementAttribute nameAttribute = new ElementAttribute();
attributes.add(nameAttribute);
...
//register to a Element Table
TElementTable table=new TElementTable()
table.setElementClass(Person.class);
table.registerElementClassAttributes(Person.class, attributes);
//register to a property sheet
sheet.registerElementClassAttributes(Person.class, attributes);
```

Using XML Register Element Attributes

We can register element attributes with API, however we can do the same thing via XML.

Global Register

```
List attributes = new ArrayList();
...
TWaverUtil.registerBeanInfoWithoutDefault(Person.class);
TWaverUtil.registerBeanInfo(Person.class, "/demo/swing/table.xml");
//or TWaverUtil.registerBeanInfo(Person.class, attributes);
table.setElementClass(Person.class);
```

Local Register

```
List attributes = new ArrayList();
...
//register to a ElementTable
table.setElementClass(Person.class);
table.registerElementClassXML(Person.class, "/demo/swing/table.xml");
//or table.registerElementClassAttributes(Person.class, attributes);
//register to a PropertySheet
sheet.registerElementClassXML(Person.class, "/demo/swing/table.xml");
//or sheet.registerElementClassAttributes(Person.class, attributes);
```

Please see other sections learn how to write XML for an Element.

Element Attribute Property Key

The property key can be the key of ClientProperty, or property name the JavaBean property.




For ClientProperty attribute, if use "name" as the element attribute key, and not set "readMethod" and "writeMethod", by default TWaver will use `element.getClientProperty("name")` and `element.putClientProperty("name",value)` to set/get this attribute value:





```
attribute.setClientPropertyKey("name");  
attribute.setClientProperty(true);
```



For a standard JavaBean attribute, if use "name" as the element attribute key, and not set "readMethod" and "writeMethod", by default TWaver will use `element.getName()/isName()` and `element.setName(value)` to set/get this attribute value:





```
attribute.setName("my name");
```

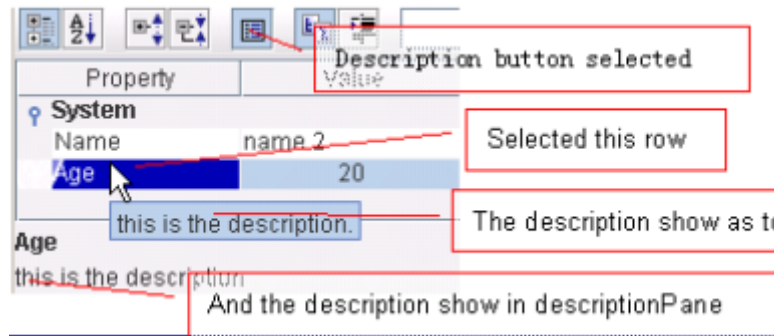

More Settings of Element Attribute

Attribute	Description
JavaClass [java.lang.Class]	<p>Class type of this property value. TWaver provides many internal table cell editors and renderers for the most often used class types. So you just tell TWaver what class type this property is, the bound editor and renderer will be used automatically.</p> <p>You can register your own renderer and editor that will be used in table or property sheet components:</p> <pre>sheet.getCellRendererManager().registerRenderer(javaClass, yourRendererClass);</pre> <p>You can make a global editor/renderer register for class type:</p> <pre>TUIManager.registerTableCellRenderer (javaClass , yourRendererClass);</pre>
ClientProperty [boolean]	<p>Whether this attribute is a ClientProperty. Set to false if this attribute is a standard JavaBean property.</p> <p> when use API define element attribute, please set this to "true" explicitly:</p> <pre>ElementAttribute ageAttribute = new ElementAttribute(); ageAttribute.setClientProperty(true); ageAttribute.setClientPropertyKey("age");</pre>
ReadMethod [java.lang.String]	<p>The read method name used to read attribute value.</p> <p> this is only work for standard JavaBean attribute. For ClientProperty, TWaver always use "getClientProperty" and "putClientProperty" to get/set value.</p>
WriteMethod [java.lang.String]	<p>The set method name used to set attribute value.</p> <p> this is only work for standard JavaBean attribute. For ClientProperty, TWaver always use "getClientProperty" and "putClientProperty" to get/set value.</p>

EnableBatch [boolean]	<p>Whether enable batch edit Sets whether this property is enable batch reading/writing. if set to true, this property will be display on property sheet when more than one element is selected.</p> <p> this setting only work for PropertySheet component.</p>
ClientPropertyKey [java.lang.String]	Key of the ClientProperty
DisplayName [java.lang.String]	<p>Display name of this attribute. For table component, this will displayed as column header; for property sheet component, this will displayed as property name.</p>
CategoryName [java.lang.String]	<p>Category name For example:</p> <pre>Category.registerCategory("age", "Age", true); attribute.setCategoryName("age");</pre> <div>  when set/change category name for attribute, it may cause change of the table column order. Attributes in the same category is continued Order of categories is the same with the registered order </div>
SortComparator [java.util.Comparator]	<p>Sort comparator The sort comparator used to compare attribute values to determine how to sort them. Use a customized comparator you can change the default order of this column.</p>
FillColor [java.awt.Color]	<p>Table cell background color. Use this color you can highlight this attribute.</p> <p> this setting only work for PropertySheet component</p>
ForeColor [java.awt.Color]	<p>Table cell foreground color. Use this color you can highlight this attribute.</p> <p> this setting only work for PropertySheet component</p>
EditorClass [java.lang.String]	<p>Editor class name. This defines the editor class used to edit this attribute value.</p>

	<p>Can passing information to a constructor as follows : EditorClassName@StringParameter1 StringParameter2 StringParameter3</p> <pre>attribute.setEditorClass("twaver.table.editor.EnumTypeEditor@owner fal</pre>
<p>RendererClass [java.lang.String]</p>	<p>Renderer class name. This defines the renderer class used to paint this attribute value. Can passing information to a constructor as follows : RendererClassName@StringParameter1 StringParameter2 StringParameter3</p> <pre>attribute.setRendererClass("twaver.table.renderer.EnumTypeRenderer@</pre>
<p>CategoryNames [java.util.List]</p>	<p>Category name this attribute located. You can set multi-level categories in property sheet component. For example:</p> <pre>Category.registerCategory("basic", "BASIC", true); Category.registerCategory("age", "Age", true); ageAttribute.setCategoryName("age"); List categoryNames = new ArrayList(); categoryNames.add(0, "basic"); categoryNames.add(1, "age"); attribute.setCategoryNames(categoryNames);</pre> <div data-bbox="826 1245 1401 1344">  </div> <div data-bbox="879 1386 1390 1449">  this setting only work for property sheet component </div>
<p>FontStyle [int]</p>	<p>The displayed font style of this attribute All available values defined in TWaverConst:</p> <pre>/** default font style */ public final static int FONT_STYLE_DEFAULT = 1; /** plain font style */ public final static int FONT_STYLE_PLAIN = 2; /** blod font style */ public final static int FONT_STYLE_BOLD = 3; /** italic font style */ public final static int FONT_STYLE_ITALIC = 4; /** italic and bold font style */ public final static int FONT_STYLE_ITALIC_BOLD = 5;</pre> <p>When use this in XML, write like this way:</p>

	<div>fontStyle=bold</div> <p>The "bold" can be following values: default plain bold italic italicblod</p> <div> this setting only work for property sheet component</div>
Sortable [boolean]	<p>Whether this column can be sort in table component. If false, the column header will displayed in a darker background.</p> <div> this setting only work for property sheet component</div>
Name [java.lang.String]	Attribute name
Visible [boolean]	Whether visible. Default is true
Width [int]	Column width for table component. Default width is 75.
Icon [javax.swing.Icon]	<p>Attribute icon</p> <div> this setting only work for property sheet Cell height will not change by the icon size</div>
Editable [boolean]	<p>Whether this attribute is editable in components. If you want this attribute is editable in compoenents, first you need to make the component itself editable (TElementTable default is editable and TPropertySheet default is not editable).</p> <div>sheet.setEditable(true); table.setEditable(true);</div>
Description [java.lang.String]	<p>A description for this attribute. It will be displayed as the tooltip text of this cell for property sheet component. When using TPropertySheetPane, it also will be displayed on the bottom description panel.</p> <div> this setting only work for property sheet component.</div>

	<div data-bbox="845 268 1415 470" data-label="Text"> <pre>ElementAttribute ageAttribute = new ElementAttribute(); ageAttribute.setClientProperty(true); ageAttribute.setClientPropertyKey("age"); ageAttribute.setDisplayName("Age"); ... ageAttribute.setDescription("this is the description.");</pre> </div> <div data-bbox="817 544 1596 878" data-label="Image">  </div>
<p>Params [java.util.Map]</p>	<div data-bbox="813 909 1441 1057" data-label="Text"> <p>Customer defined parameters This used to sets the additional parameters of this attribute. This is a map so you can carry arbitrary number of parameters for these properties. It can so be defined in the XML files like this:</p> </div> <div data-bbox="845 1124 1364 1321" data-label="Text"> <pre><attribute clientPropertyKey="MyProp1" displayName="PropertyA" categoryName="mine"> <param key="minLength" value="5"/> <param key="message" value="PropertyA's min length is 5."/> </attribute></pre> </div> <div data-bbox="869 1429 1388 1547" data-label="Text"> <p> the "key" and "value" here will be regarded as a java.lang.String. If you want use other Object as the key/ value, please operate by APIs.</p> </div> <div data-bbox="813 1568 1422 1601" data-label="Text"> <p>Use can also use API to add additional parameters:</p> </div> <div data-bbox="845 1666 1185 1753" data-label="Text"> <pre>attribute.addParam(key , vlaue); or attribute.setParams(params);</pre> </div> <div data-bbox="813 1814 1447 1879" data-label="Text"> <p>In property sheet, you can get this parameter by this way:</p> </div> <div data-bbox="845 1942 1412 2060" data-label="Text"> <pre>TPropertyDescriptor property; //the index here is displayed index, include category rows. //This index will be changed after sort.</pre> </div>

	<pre> property = sheet.getPropertyDescriptorByRowIndex(rowIndex); //or property=sheet.getPropertyDescriptorByPropertyName(myProp); //If ClientProperty is true, the property name will //be the prefix+ clientPropertyName: String prefix= TWaverConst.CLIENT_PROPERTY_PREFIX; String name= prefix +clientPropertyName; property = sheet.getPropertyDescriptorByPropertyName(name); ElementAttribute attribute= property.getElementAttribute(); //or ElementAttribute attribute= sheet.getElementAttribute(rowIndex) Object param=attribute.getParam(paramName); </pre> <pre> //For example: ageAttribute.addParam("node", new Node("node_A")); ... sheet.addMouseListener(new MouseAdapter(){ public void mouseReleased(MouseEvent e) { if(e.getClickCount() == 2){ int rowIndex = sheet.rowAtPoint(e.getPoint()); TPropertyDescriptor property = sheet.getPropertyDescriptorByRowIndex(rowIndex); Node node=(Node)property.getElementAttribute().getParam("node"); JOptionPane.showMessageDialog(sheet, rowIndex+ " " +node.getID()); } } }); </pre>
rowPackParticipable [boolean]	Checks whether participate with packing table rows.
extraWidthAssignable [boolean]	Checks whether to be assigned extra with after packing

An Example for Using Element Attribute

Now we give an example how to use XML to config Element attributes in TElementTable component.

```
public static void main(String[] args) {
    TDataBox box = new TDataBox();
    TElementTable table = new TElementTable(box);
    final TPropertySheet sheet = new TPropertySheet(box);
    List attributes = new ArrayList();
    //register IntRenderer
    TUIManager.registerTableCellRenderer(Integer.class, IntRenderer.class);

    //define a "name" attribute
    ElementAttribute nameAttribute = new ElementAttribute();
    nameAttribute.setClientProperty(false);
    nameAttribute.setName("name");
    nameAttribute.setDisplayName("Name");
    nameAttribute.setEditable(false);
    attributes.add(nameAttribute);

    //config "age" attribute
    ElementAttribute ageAttribute = new ElementAttribute();
    ageAttribute.setEditable(true);
    ageAttribute.setJavaClass(Integer.class);
    ageAttribute.setClientProperty(true);
    ageAttribute.setClientPropertyKey("age");
    ageAttribute.setDisplayName("Age");
    ageAttribute.setFillColor(Color.blue);
    ageAttribute.setForecolor(Color.white);
    //register category
    Category.registerCategory("basic", "BASIC", true);
    Category.registerCategory("age", "Age", true);
    List categoryNames = new ArrayList();
    categoryNames.add(0, "basic");
    categoryNames.add(1, "age");
    ageAttribute.setSortable(false);
    ageAttribute.setCategoryNames(categoryNames);
    ageAttribute.setDescription("this is the description.");
    final String paramName="node";
    ageAttribute.addParam(paramName, new Node("node_A"));
    ageAttribute.setIcon(TWaverUtil.getImageIcon("/on.png"));
    attributes.add(ageAttribute);

    //config description attribute
    ElementAttribute descriptionAttribute = new ElementAttribute();
    descriptionAttribute.setClientProperty(true);
    descriptionAttribute.setClientPropertyKey("description");
    descriptionAttribute.setDisplayName("Description");
    descriptionAttribute.setWidth(300);
    descriptionAttribute.setFontStyle(TWaverConst.FONT_STYLE_BOLD);
    descriptionAttribute.setEditable(false);
    attributes.add(descriptionAttribute);

    table.setElementClass(Person.class);
    //1. you can use API register attributes
    table.registerElementClassAttributes(Person.class, attributes);
    sheet.registerElementClassAttributes(Person.class, attributes);
    //2. or, you can use XML file to register attributes
    sheet.registerElementClassXML(Person.class, "/demo/swing/table.xml");
    table.registerElementClassXML(Person.class, "/demo/swing/table.xml");

    sheet.setEditable(true);
}
```

```

table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
for (int i = 0; i < 10; i++) {
    box.addElement(new Person("name " + i * 10, "Description at time " + i + " " + new Date().getTime()));
}

//make a window show it.
JFrame frame = new JFrame("Table element attribute test");
frame.getContentPane().add(new JScrollPane(table));
frame.setSize(500, 300);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);
JFrame frame2 = new JFrame("Sheet element attribute test");
frame2.setLocation(0,150);
frame2.getContentPane().add(new TPropertySheetPane(sheet));
frame2.setSize(500, 300);
frame2.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame2.setVisible(true);

sheet.addMouseListener(new MouseAdapter() {
    public void mouseReleased(MouseEvent e) {
        if (e.getClickCount() == 2) {
            int rowIndex = sheet.rowAtPoint(e.getPoint());
            TPropertyDescriptor property = sheet.getPropertyDescriptorByRowIndex(rowIndex);
            Object param = property.getElementAttribute().getParam(paramName);
            if (param != null) {
                JOptionPane.showMessageDialog(sheet, "clicked at row " + rowIndex + " param is " + param);
            }
        }
    }
});
}

public static class Person extends Node {
    public Person() {
    };
    public Person(String name, int age, String description) {
        this.setName(name);
        this.putClientProperty("age", age);
        this.putClientProperty("description", description);
    }
}

public static class IntRenderer extends DefaultTableCellRenderer {
    public IntRenderer() {
        TWaverUtil.setHorizontalAlignment(this, "CENTER");
    }
}

```

The XML config file:

```

<beaninfo>
<attribute
name="name"
displayName="Name"/>
<category name="basic">
<category name="age">
<attribute
editable="true"
javaClass="java.lang.Integer"
clientPropertyKey="age"
displayName="Age"
fillColor="blue"
foreColor="white"

```

```

sortable="false"
description="this is the description."
icon="/on.png">
<param key="node" value="node only string"/>
</attribute>
</category>
</category>
<attribute
clientPropertyKey="description"
displayName="Description"
width="300"
fontStyle="bold"
editable="false"/>
</beaninfo>

```

Run this example:

Table element attribute test		
Name	Description	Age
name 0	Description at time 0 1189148008640	0
name 1	Description at time 1 1189148008718	10
name 2	Description at time 2 1189148008718	20
name 3	Description at time 3 1189148008718	30
name 4	Description at time 4 1189148008718	40
name 5	Description at time 5 1189148008718	50

Sheet element attribute test	
Property	Value
System	
Name	name 2
Description	Description at time 2 1189148008718
BASIC	
Age	
Age	20

Major Interfaces in TWaver

Here to introduce some major and often used interfaces defined in TWaver Java. You can find more detailed information in JavaDoc contained in your TWaver Java Product CD.

- [Equipment Interface Method List](#)
- [Resizable Interface Method List](#)
- [TSubNetwork Interface Method List](#)
- [Network Related Interfaces](#)

Equipment Interface Method List

twaver.Equipment

This interface defines a telecom equipment object. Equipment refers to all elements displayed on the chassis. Rack, Shelf, Slot, Card, Port etc are all instance of Equipment interface.

Equipment object extends Element interface and add several properties which is often used when display the equipment chassis.

Method	Description
get/setTag(String tag)	The tag information of this equipment object.
is/setExist(boolean exist)	Whether this equipment object is 'exist' on chassis.

Resizable Interface Method List

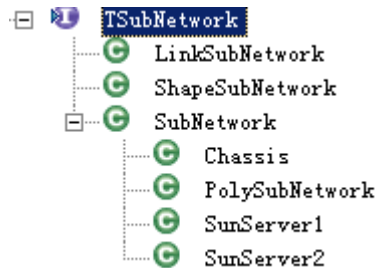
twaver.Resizable

This interface defines an element that is resizable. Width and height of a resizable element can be changed by extended APIs.

Method	Description
get/setSize(int width, int height) get/setSize(Dimension size)	size of this resizable element.
get/setWidthSize(int width);	the width value for this resizable element.
get/setHeightSize(int height);	the height value for this resizable element

TSubNetwork Interface Method List

TSubNetwork is an extended Element Interface, it describes a model of a "SubNetwork", and a drill-down enabled Element. It defines the data source, the data loading policy. Any Element implements this Interface will provide the ability to drill-down, lazy loading and other "SubNetwork" behaviour.



twaver.TSubNetwork

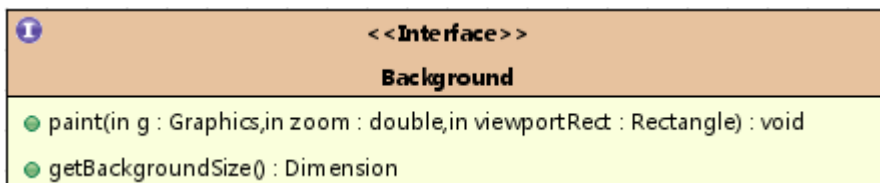
Method	Description
get/setViewPoint(Point viewPoint)	The left-top scroll viewport location of this subnetwork.
get/setBackground(Background background)	Background of this subnetwork
get/setDataSource(String dataSource)	Data source of this subnetwork In lazy load mode, this data source will tell TWaver where get data for this subnetwork. The data will be loaded when the subnetwork first explored.
is/setDataLoaded(boolean dataLoaded)	Whether the data source of this subnetwork has been loaded. In lazy load mode, this flag will tell TWaver that data of this subnetwork has been loaded, no need to load this subnetwork again. You can set this flag to false to reload or refresh subnetwork data.

Network Related Interfaces

twaver.network.background.Background

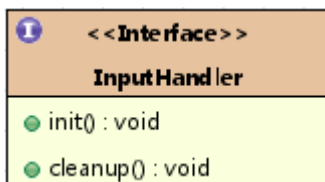
Network component uses a Background object to paint map background. The background can be image, texture, color or even GIS background. If the background setted to the network component, network will paint background first, then paint all elements. So background will always at the bottom layer.

The developer need only subclass this interface and implements the paint method, paint anything you want.



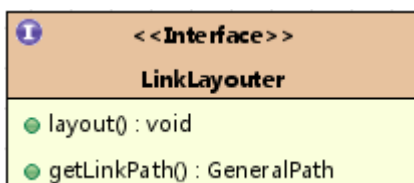
twaver.network.inputhandler.InputHandler

Define a input event listener include both mouse and keyboard input event. No any new method defined.



twaver.network.layout.LinkLayouter

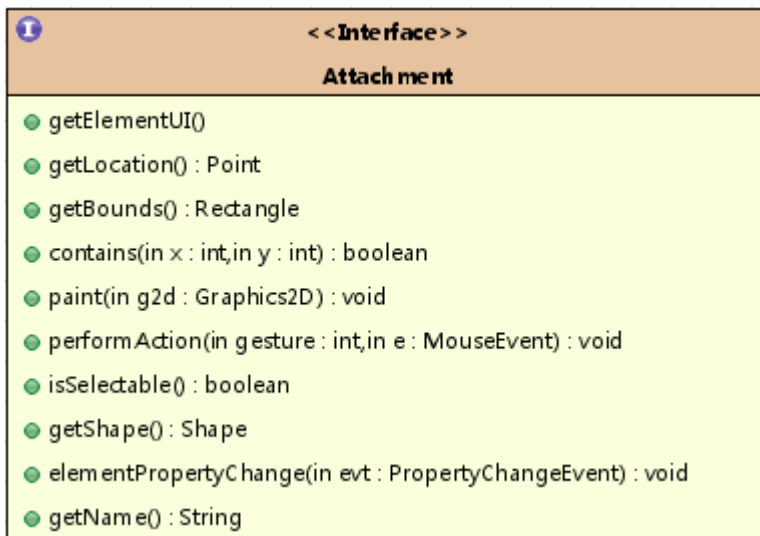
This interface defines a link layouter. It applies a layout algorithm that routes the links of the graph, leaving the node positions unchanged.



twaver.network.ui.Attachment

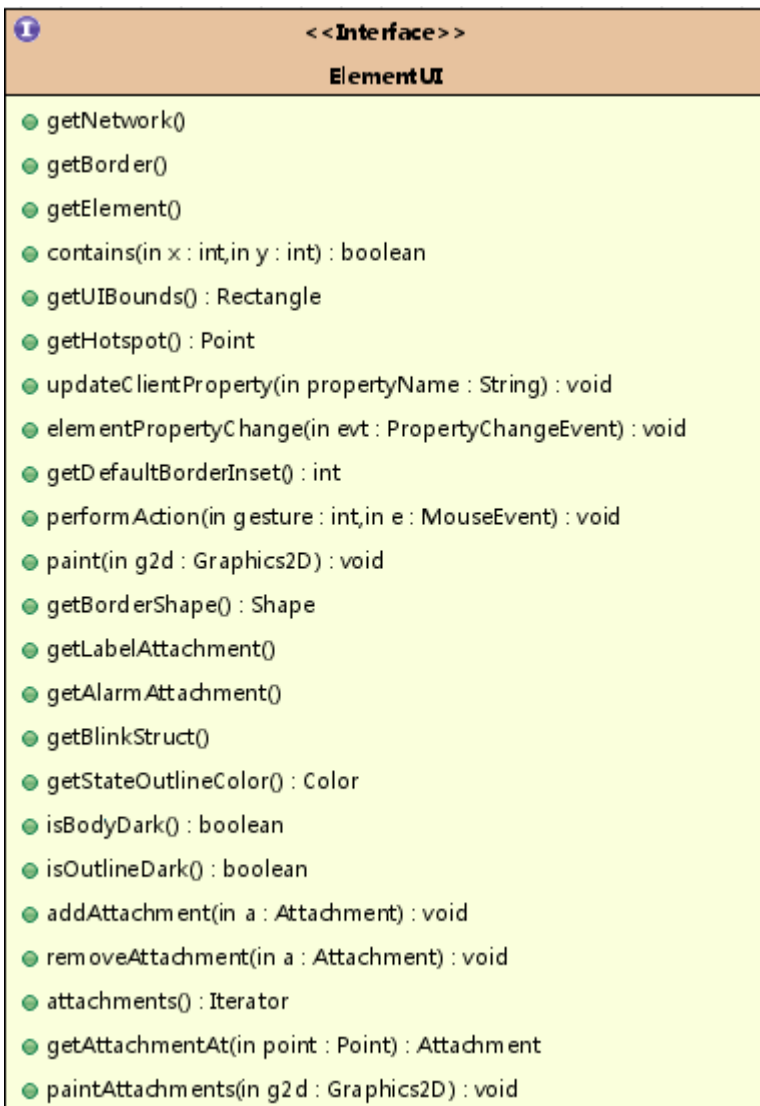
This interface defines the visualize graph element which can be attached on an ElementUI object. The ElementUI objects may use some Attachment objects to paint and represent the Element properties. As part of ElementUI, the attachment is absolute act as View of MVC architecture. It used by ElementUI to reresent some special Element properties. For example, NameAttachment used to paint element's display name, TipAttachment used to paint element tip information.

Developer my write new classes implemented this interface, and related it with an element object's properti, to extends the ui class representation.



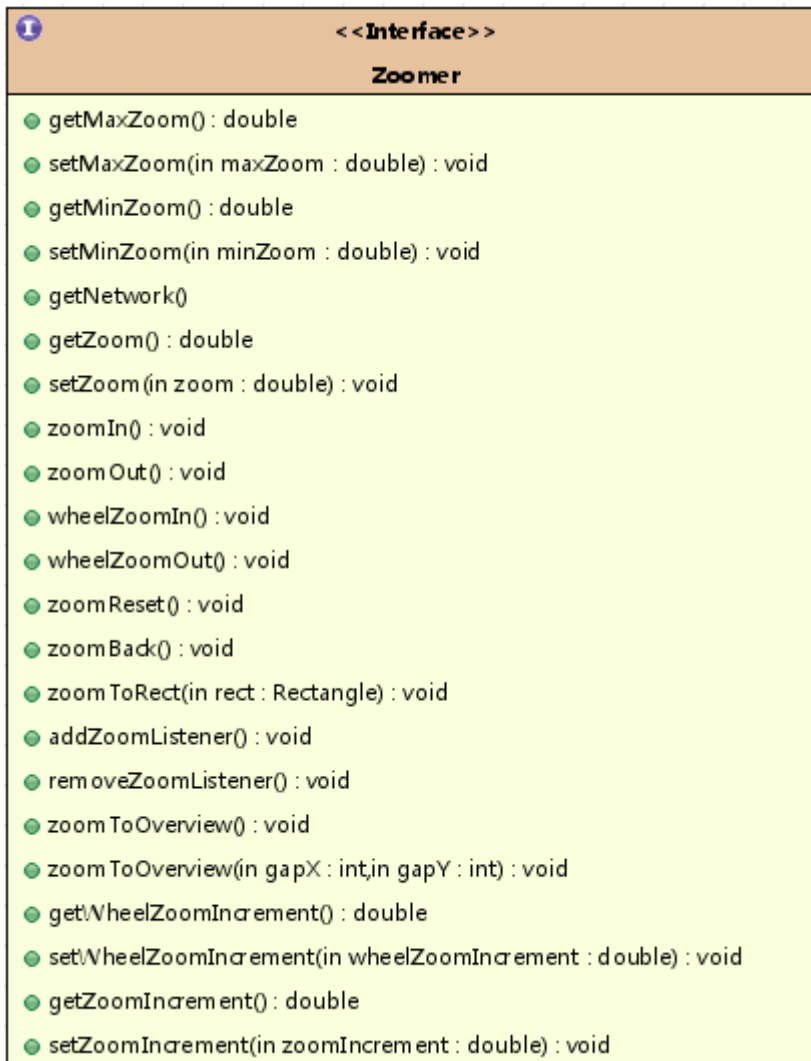
twaver.network.ui.ElementUI

ElementUI defines an element UI delegate painter. The UI class is used to pluggable network element display style.



twaver.network.zoom.Zoomer

This interface defines the zoom policy for network component.

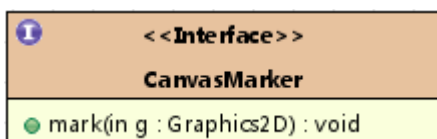


twaver.network.CanvasCushion



twaver.network.CanvasMarker

This interface defines a network canvas marker. A canvas marker can draw something over the network canvas.If developers want draw some additional information on the network canvas like mouse trace or other marker signal, they can implements this interface and write the drawing code in "mark" method.



twaver.network.InteractionListener

The interface is created for an object that can listen to interaction events.



TWaver Property Key List

In TWaver, Element property change will send a `PropertyChangeEvent` out to the listeners:

```
public void setName(String name) {
    String oldValue = this.name;
    this.name = name;
    this.firePropertyChange(TWaverConst.PROPERTYNAME_NAME,oldValue,name);
}
```

We can install a `PropertyChangeListener` on `DataBox` to catch these event and perform our actions. In fact we can catch all property change events for all Elements:

```
box.addElementPropertyChangeListener(new PropertyChangeListener(){
    public void propertyChange(PropertyChangeEvent evt) {
        String propertyName=evt.getPropertyName();
        Element source=(Element)evt.getSource();
        Object newValue=evt.getNewValue();
        Object oldValue=evt.getOldValue();
    }
});
```

All property names predefined in TWaver can be found in class `TWaverConst`:

KEY	VALUE
PROPERTYNAME_LAYERID	layerID
PROPERTYNAME_ICON	icon
PROPERTYNAME_IMAGE	image
PROPERTYNAME_ALARMSTATE	alarmState
PROPERTYNAME_ENABLEALARMPROPAGATIONFROMCHILDREN	enableAlarmPropagationFromChildren
PROPERTYNAME_CHILDREN	children
PROPERTYNAME_PARENT	parent
PROPERTYNAME_NAME	name
PROPERTYNAME_DISPLAYNAME	displayName
PROPERTYNAME_USEROBJECT	userObject
PROPERTYNAME_TOOLTIPTEXT	toolTipText
PROPERTYNAME_SELECTED	selected
PROPERTYNAME_VISIBLE	visible
PROPERTYNAME_LOCATION	location
PROPERTYNAME_SIZE	size

PROPERTYNAME_WIDTH	width
PROPERTYNAME_HEIGHT	height
PROPERTYNAME_EXPAND	expand
PROPERTYNAME_BASESHAPE	baseShape
PROPERTYNAME_TRANSFORMEDSHAPE	transformedShape
PROPERTYNAME_HOST	host
PROPERTYNAME_EXIST	exist
PROPERTYNAME_TAG	tag
PROPERTYNAME_FROM	from
PROPERTYNAME_TO	to
PROPERTYNAME_FROMAGENT	fromAgent
PROPERTYNAME_TOAGENT	toAgent
PROPERTYNAME_LINKTYPE	linkType
PROPERTYNAME_GROUPTYPE	groupType
PROPERTYNAME_SHAPENODETYPE	shapeNodeType
PROPERTYNAME_BTSANTENNAPOWER	power
PROPERTYNAME_BEAMDIRECTION	beamDirection
PROPERTYNAME_BEAMWIDTH	beamWidth
PROPERTYNAME_BEAMALPHA	beamAlpha
PROPERTYNAME_ANTENNA	antenna
PROPERTYNAME_ANTENNATYPE	antennaType
PROPERTYNAME_BTS	BTS
PROPERTYNAME_SHAPELINKPOINTS	points
PROPERTYNAME_SHAPENODEPOINTS	points
PROPERTYNAME_GROUPMODELSHAPE	groupModelShape
PROPERTYNAME_GROUPUISHAPE	groupUIShape
PROPERTYNAME_POLYLINESEGMENT	segment
PROPERTYNAME_ANTENNASHAPE	antennaShape
PROPERTYNAME_POLYLINEBLINKINGOBJECT	blinkingObject
PROPERTYNAME_EQUIPINDEX	equipIndex
PROPERTYNAME_EQUIPCOUNT	equipCount
PROPERTYNAME_TSUBNETWORKBACKGROUND	background

PROPERTYNAME_TSUBNETWORKDATASOURCE	dataSource
PROPERTYNAME_TSUBNETWORKDATALOADED	dataLoaded
PROPERTYNAME_TSUBNETWORKVIEWPOINT	viewPoint
PROPERTYNAME_ELEMENT_TREE_ICON	element.tree.icon
PROPERTYNAME_DRAW_IMAGE_SHAPE	draw.image.shape
PROPERTYNAME_DRAW_ICON_SHAPE	draw.icon.shape
PROPERTYNAME_RENDER_ALPHA	render.alpha
PROPERTYNAME_RENDER_COLOR	render.color
PROPERTYNAME_STATE_OUTLINE_COLOR	state.outline.color
PROPERTYNAME_STATE_OUTLINE_INSETS	state.outline.insets
PROPERTYNAME_TEXTURE_FACTORY	texture.factory
PROPERTYNAME_BODY_COLOR	body.color
PROPERTYNAME_BODY_RAISED	body.raised
PROPERTYNAME_BODY_FILL	body.fill
PROPERTYNAME_BORDER_ANTIALIAS	border.antialias
PROPERTYNAME_BORDER_VISIBLE	border.visible
PROPERTYNAME_BORDER_STROKE	border.stroke
PROPERTYNAME_BORDER_COLOR	border.color
PROPERTYNAME_BORDER_INSETS	border.insets
PROPERTYNAME_LABEL_FONT	label.font
PROPERTYNAME_LABEL_COLOR	label.color
PROPERTYNAME_LABEL_VISIBLE	label.visible
PROPERTYNAME_LABEL_BORDER	label.border
PROPERTYNAME_LABEL_BORDER_STROKE	label.border.stroke
PROPERTYNAME_LABEL_BORDER_COLOR	label.border.color
PROPERTYNAME_LABEL_UNDERLINE	label.underline
PROPERTYNAME_LABEL_UNDERLINE_STROKE	label.underline.stroke
PROPERTYNAME_LABEL_UNDERLINE_COLOR	label.underline.color
PROPERTYNAME_LABEL_POSITION	label.position
PROPERTYNAME_LABEL_XOFFSET	label.xoffset
PROPERTYNAME_LABEL_YOFFSET	label.yoffset
PROPERTYNAME_LABEL_DIRECTION	label.direction

PROPERTYNAME_LABEL_ORIENTATION	label.orientation
PROPERTYNAME_LABEL_SELECTABLE	label.selectable
PROPERTYNAME_LABEL_HIGHLIGHTABLE	label.highlightable
PROPERTYNAME_LABEL_HIGHLIGHT_BACKGROUND	label.highlight.background
PROPERTYNAME_LABEL_HIGHLIGHT_FOREGROUND	label.highlight.foreground
PROPERTYNAME_LABEL_MAXLENGTH	label.maxlength
PROPERTYNAME_MESSAGE_CONTENT	message.content
PROPERTYNAME_MESSAGE_FONT	message.font
PROPERTYNAME_MESSAGE_FOREGROUND	message.foreground
PROPERTYNAME_MESSAGE_BACKGROUND	message.background
PROPERTYNAME_MESSAGE_POSITION	message.position
PROPERTYNAME_MESSAGE_XOFFSET	message.xoffset
PROPERTYNAME_MESSAGE_YOFFSET	message.yoffset
PROPERTYNAME_MESSAGE_XGAP	message.xgap
PROPERTYNAME_MESSAGE_YGAP	message.ygap
PROPERTYNAME_MESSAGE_WIDTH	message.width
PROPERTYNAME_MESSAGE_HEIGHT	message.height
PROPERTYNAME_MESSAGE_COMPONENT	message.component
PROPERTYNAME_MESSAGE_STYLE	message.style
PROPERTYNAME_MESSAGE_DIRECTION	message.direction
PROPERTYNAME_MESSAGE_OPAQUE	message.opaque
PROPERTYNAME_MESSAGE_TAIL	message.tail
PROPERTYNAME_MESSAGE_ARC	message.arc
PROPERTYNAME_MESSAGE_MINIMIZED_ICON	message.minimized.icon
PROPERTYNAME_MESSAGE_SHADOW_VISIBLE	message.shadow.visible
PROPERTYNAME_MESSAGE_SHADOW_COLOR	message.shadow.color
PROPERTYNAME_MESSAGE_BORDER_VISIBLE	message.border.visible
PROPERTYNAME_MESSAGE_BORDER_COLOR	message.border.color
PROPERTYNAME_MESSAGE_BORDER_STROKE	message.border.stroke
PROPERTYNAME_MESSAGE_CLOSABLE	message.closable
PROPERTYNAME_MESSAGE_SHRINKABLE	message.shrinkable
PROPERTYNAME_MESSAGE_MINIMIZABLE	message.minimizable

PROPERTYNAME_MESSAGE_AUTO_ADJUST_DIRECTION	message.auto.adjust.direction
PROPERTYNAME_ATTACHMENT_POSITION	attachment.position
PROPERTYNAME_ATTACHMENT_ORIENTATION	attachment.orientation
PROPERTYNAME_ATTACHMENT_XOFFSET	attachment.xoffset
PROPERTYNAME_ATTACHMENT_YOFFSET	attachment.yoffset
PROPERTYNAME_ATTACHMENT_XGAP	attachment.xgap
PROPERTYNAME_ATTACHMENT_YGAP	attachment.ygap
PROPERTYNAME_ALARM_BALLOON_ALPHA	alarm.balloon.alpha
PROPERTYNAME_ALARM_BALLOON_POSITION	alarm.balloon.position
PROPERTYNAME_ALARM_BALLOON_DIRECTION	alarm.balloon.direction
PROPERTYNAME_ALARM_BALLOON_XOFFSET	alarm.balloon.xoffset
PROPERTYNAME_ALARM_BALLOON_YOFFSET	alarm.balloon.yoffset
PROPERTYNAME_ALARM_BALLOON_VISIBLE	alarm.balloon.visible
PROPERTYNAME_ALARM_BALLOON_TEXT_FONT	alarm.balloon.text.font
PROPERTYNAME_ALARM_BALLOON_TEXT_COLOR	alarm.balloon.text.color
PROPERTYNAME_ALARM_BALLOON_TEXT_BLINKABLE	alarm.balloon.text.blinkable
PROPERTYNAME_ALARM_BALLOON_OUTLINE_COLOR	alarm.balloon.outline.color
PROPERTYNAME_ALARM_BALLOON_SHADOW_COLOR	alarm.balloon.shadow.color
PROPERTYNAME_ALARM_BALLOON_SHADOW_OFFSET	alarm.balloon.shadow.offset
PROPERTYNAME_POLY_OUTLINE_WIDTH	poly.outline.width
PROPERTYNAME_POLY_OUTLINE_COLOR	poly.outline.color
PROPERTYNAME_POLY_FILL	poly.fill
PROPERTYNAME_POLY_3D	poly.3d
PROPERTYNAME_LINK_PROPORTION	link.proportion
PROPERTYNAME_LINK_EXTEND	link.extend
PROPERTYNAME_LINK_SHAPE	link.shape
PROPERTYNAME_LINK_STYLE	link.style
PROPERTYNAME_LINK_HOLLOW	link.hollow
PROPERTYNAME_LINK_COLOR	link.color
PROPERTYNAME_LINK_OUTLINE_COLOR	link.outline.color
PROPERTYNAME_LINK_OUTLINE_WIDTH	link.outline.width
PROPERTYNAME_LINK_WIDTH	link.width

PROPERTYNAME_LINK_FLOWING_WIDTH	link.floating.width
PROPERTYNAME_LINK_FROMARROW	link.fromArrow
PROPERTYNAME_LINK_TOARROW	link.toArrow
PROPERTYNAME_LINK_FLOWING	link.floating
PROPERTYNAME_LINK_FLOWING_CONVERSE	link.floating.converse
PROPERTYNAME_LINK_BLINKING	link.blinking
PROPERTYNAME_LINK_BLINKING_COLOR	link.blinking.color
PROPERTYNAME_LINK_3D	link.3d
PROPERTYNAME_LINK_FLOWING_COLOR	link.floating.color
PROPERTYNAME_LINK_BUNDLE_EXPAND	link.bundle.expand
PROPERTYNAME_LINK_BUNDLE_INDEX	link.bundle.index
PROPERTYNAME_LINK_BUNDLE_SIZE	link.bundle.size
PROPERTYNAME_LINK_BUNDLE_OFFSET	link.bundle.offset
PROPERTYNAME_LINK_BUNDLE_GAP	link.bundle.gap
PROPERTYNAME_LINK_LABEL_ROTATABLE	link.label.rotatable
PROPERTYNAME_LINK_ANTIALIAS	link.antialias
PROPERTYNAME_LINK_FROM_ARROW_STYLE	link.from.arrow.style
PROPERTYNAME_LINK_TO_POSITION	link.to.position
PROPERTYNAME_LINK_TO_XOFFSET	link.to.xoffset
PROPERTYNAME_LINK_TO_YOFFSET	link.to.yoffset
PROPERTYNAME_LINK_FROM_POSITION	link.from.position
PROPERTYNAME_LINK_FROM_XOFFSET	link.from.xoffset
PROPERTYNAME_LINK_FROM_YOFFSET	link.from.yoffset
PROPERTYNAME_LINK_TO_ARROW_STYLE	link.to.arrow.style
PROPERTYNAME_LINK_FROM_ARROW_COLOR	link.from.arrow.color
PROPERTYNAME_LINK_TO_ARROW_COLOR	link.to.arrow.color
PROPERTYNAME_LINK_FROM_ARROW_OUTLINE	link.from.arrow.outline
PROPERTYNAME_LINK_TO_ARROW_OUTLINE	link.to.arrow.outline
PROPERTYNAME_LINK_FROM_ARROW_OUTLINE_COLOR	link.from.arrow.outline.color
PROPERTYNAME_LINK_TO_ARROW_OUTLINE_COLOR	link.to.arrow.outline.color
PROPERTYNAME_LINK_TO_ARROW_XOFFSET	link.to.arrow.xoffset
PROPERTYNAME_LINK_TO_ARROW_YOFFSET	link.to.arrow.yoffset

PROPERTYNAME_LINK_FROM_ARROW_XOFFSET	link.from.arrow.xoffset
PROPERTYNAME_LINK_FROM_ARROW_YOFFSET	link.from.arrow.yoffset
PROPERTYNAME_LINK_HANDLER_POSITION	link.handler.position
PROPERTYNAME_LINK_HANDLER_XOFFSET	link.handler.xoffset
PROPERTYNAME_LINK_HANDLER_YOFFSET	link.handler.yoffset
PROPERTYNAME_LINK_ORTHOGONAL_DIRECTION	link.orthogonal.direction
PROPERTYNAME_CARD_BOLT_TOP	card.bolt.top
PROPERTYNAME_CARD_BOLT_BOTTOM	card.bolt.bottom
PROPERTYNAME_CARD_BOLT_LEFT	card.bolt.left
PROPERTYNAME_CARD_BOLT_RIGHT	card.bolt.right
PROPERTYNAME_GROUP_OUTLINE	group.outline
PROPERTYNAME_GROUP_OUTLINE_STROKE	group.outline.stroke
PROPERTYNAME_GROUP_FILL	group.fill
PROPERTYNAME_GROUP_OPAQUE	group.opaque
PROPERTYNAME_GROUP_OUTLINE_COLOR	group.outline.color
PROPERTYNAME_GROUP_FILL_COLOR	group.fill.color
PROPERTYNAME_GROUP_ANGLE	group.angle
PROPERTYNAME_GROUP_3D	group.3d
PROPERTYNAME_GROUP_ANTIALIAS	group.antialias
PROPERTYNAME_GROUP_DEEP	group.deep
PROPERTYNAME_GROUP_INSETS	group.insets
PROPERTYNAME_GROUP_CHILDREN_OUTCROP	group.children.outcrop
PROPERTYNAME_GROUP_DOUBLE_CLICK_ENABLE	group.double.click.enable
PROPERTYNAME_GROUP_HANDLER_EXPAND_ICON	group.handler.expand.icon
PROPERTYNAME_GROUP_HANDLER_CLOSE_ICON	group.handler.close.icon
PROPERTYNAME_GROUP_HANDLER_EMPTY_ICON	group.handler.empty.icon
PROPERTYNAME_GROUP_HANDLER_VISIBLE	group.handler.visible
PROPERTYNAME_GROUP_HANDLER_POSITION	group.handler.position
PROPERTYNAME_GROUP_HANDLER_XOFFSET	group.handler.xoffset
PROPERTYNAME_GROUP_HANDLER_YOFFSET	group.handler.yoffset
PROPERTYNAME_GROUP_GRADIENT	group.gradient
PROPERTYNAME_GROUP_GRADIENT_COLOR	group.gradient.color

PROPERTYNAME_GROUP_GRADIENT_FACTORY	group.gradient.factory
PROPERTYNAME_EQUIPDIRECTION	equip.direction
PROPERTYNAME_SHELF_BORDER	shelf.border
PROPERTYNAME_SHELF_INNER_BORDER	shelf.inner.border
PROPERTYNAME_SLOT_BORDER	slot.border
PROPERTYNAME_SLOT_INNER_BORDER	slot.inner.border
PROPERTYNAME_SHAPENODE_ALWAYS_SHOW_POINTS	shapenode.always.show.points
PROPERTYNAME_SHAPENODE_SHOW_DASH_LINE	shapenode.show.dash.line
PROPERTYNAME_CUSTOM_DRAW	custom.draw
PROPERTYNAME_CUSTOM_DRAW_SHAPE_FACTORY	custom.draw.shape.factory
PROPERTYNAME_CUSTOM_DRAW_DEFAULT_BORDER	custom.draw.default.border
PROPERTYNAME_CUSTOM_DRAW_ANTIALIAS	custom.draw.antialias
PROPERTYNAME_CUSTOM_DRAW_FILL	custom.draw.fill
PROPERTYNAME_CUSTOM_DRAW_FILL_3D	custom.draw.fill.3d
PROPERTYNAME_CUSTOM_DRAW_FILL_COLOR	custom.draw.fill.color
PROPERTYNAME_CUSTOM_DRAW_OUTLINE	custom.draw.outline
PROPERTYNAME_CUSTOM_DRAW_OUTLINE_3D	custom.draw.outline.3d
PROPERTYNAME_CUSTOM_DRAW_OUTLINE_COLOR	custom.draw.outline.color
PROPERTYNAME_CUSTOM_DRAW_OUTLINE_STROKE	custom.draw.outline.stroke
PROPERTYNAME_CUSTOM_DRAW_GRADIENT	custom.draw.gradient
PROPERTYNAME_CUSTOM_DRAW_GRADIENT_FACTORY	custom.draw.gradient.factory
PROPERTYNAME_CUSTOM_DRAW_GRADIENT_COLOR	custom.draw.gradient.color