



TWaver[®] Flex

Developer Guide

Version 2.0

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

- Home 
 - TWaver Flex Introduction
 - At a Glance
 - Getting Started
 - Convention
 - TWaver Flex Quick Start
 - Setup Environment
 - Basic Knowledge
 - The Design Model and Frame of TWaver Flex
 - The Data Model of TWaver Flex
 - The Data Element of TWaver Flex
 - The Data Container of TWaver Flex
 - The View Components of TWaver Flex
 - Network Introduction
 - Table Introduction
 - Tree Introduction
 - Data Serialization
 - Data Element
 - Basic Data Element
 - Alarm Data Element
 - Layer Data Element
 - Topology Element
 - twaver.Node
 - twaver.Link
 - Links Binding
 - Links Type
 - twaver.Follower
 - twaver.ISubNetwork
 - twaver.Group
 - twaver.ShapeNode
 - twaver.Grid
 - Data Container
 - Events of Data Container
 - Quick Finder Mechanism
 - Selection Model Appliance
 - Network Component
 - Network Hierarchy
 - Network Adding Background
 - Network Filter
 - Network Function for Style Rule
 - Network GUI Interaction
 - Appendix
 - Network Element Stylesheet

Home

This documentation introduce you how to use TWaver Flex to build Flex based software.

- [TWaver Flex Introduction](#)
- [At a Glance](#)
- [Getting Started](#)
- [Basic Knowledge](#)
- [Data Element](#)
- [Data Container](#)
- [Network Component](#)
- [Tree Component](#)
- [Table Component](#)
- [Alarm](#)
- [Application Development](#)
- [Examples](#)
- [FAQ](#)
- [Appendix](#)

TWaver Flex Introduction

TWaver Flex is a branch of TWaver based on Adobe Flex technology. TWaver Flex provides the same functions as TWaver Java but with Flex technical features. TWaver is a bunch of professional visualization products mainly designed for Telecom OSS (Operation Support System) developers. TWaver also can be used to build common GUI. Currently, TWaver provides product branch for Java Swing, Adobe Flex, Web AJAX/SVG and Silverlight technologies.

TWaver Flex provides a set of well-designed professional graphical components, a MVC framework and APIs for developers to use. With TWaver, you can create network topology view, equipment panel view, map view, charts and other complex GUI very easily, in minute.

TWaver Flex Structure

1. Data container component;
2. Predefined managed objects;
3. Predefined graphical components;
4. XML input/output;
5. Auto-layout interface;
6. API;

Main Features

1. DataBox Component: DataBox maintains all managed objects. As the data source and data container, DataBox controls the data change and monitors the data property changes.
2. Predefined Object: TWaver Flex predefined a set of managed objects (often used in the Telecom industry) such as node, link, port, rack and etc. Developers can use those objects to build their Telecom software quickly.
3. TWaver Flex Object: TWaver Flex offers a set of graphical components including network, tree, table, map and etc.
4. GUI Special Effects: TWaver Flex takes full advantage of Flex technology to display abundance effects such as video, animation, skin, gradients and etc.
5. Auto-layout: TWaver Flex offers many auto-layout algorithms for developers to layout the network topology. This will help to better organize the network data in a more understandable way.
6. Integration Framework to Server side: Integrated with Java or .NET server technologies.
7. API and detailed documents.

TWaver Flex Runtime Environment

- Hardware: CPU: Pentium II 450+; Memory: 128M+; Hard disk: 20G+;
- Software: All flash enabled browser/OS;

TWaver Flex Development

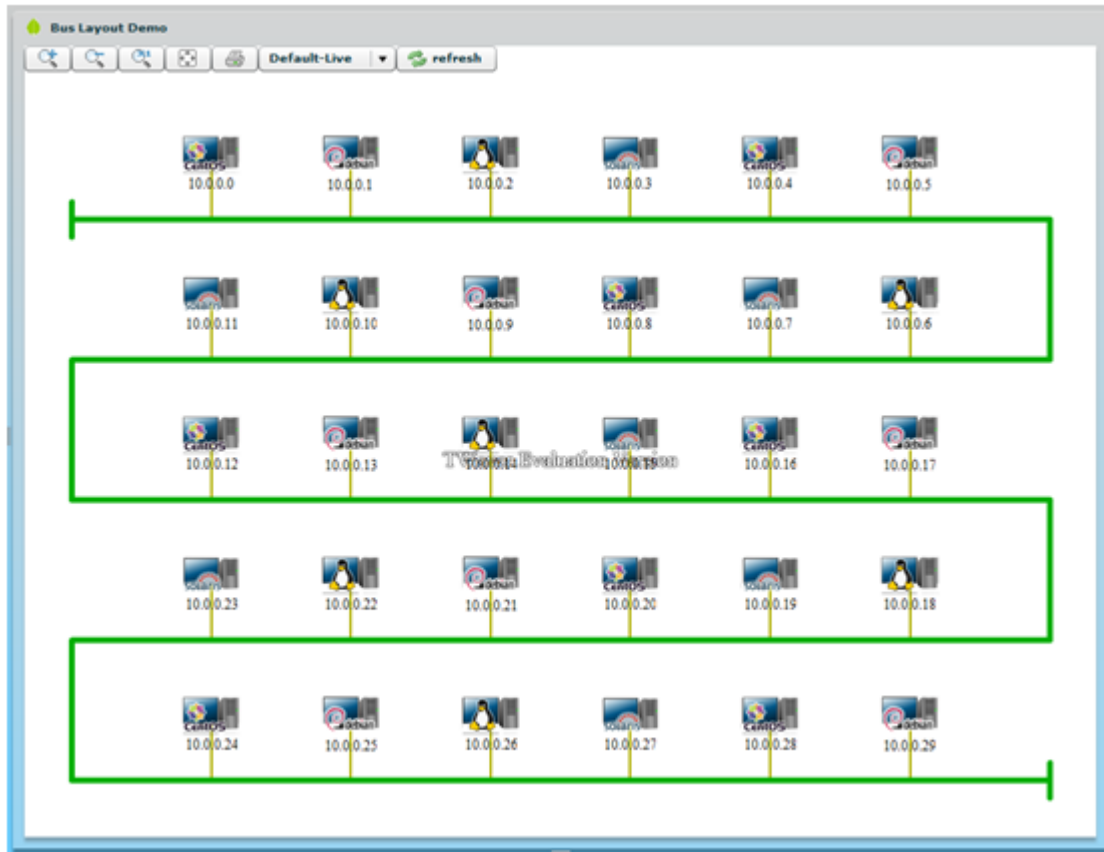
TWaver Flex supports Adobe Flex ActionScript 3.0 and Adobe Flex SDK 3.4.1 or later.

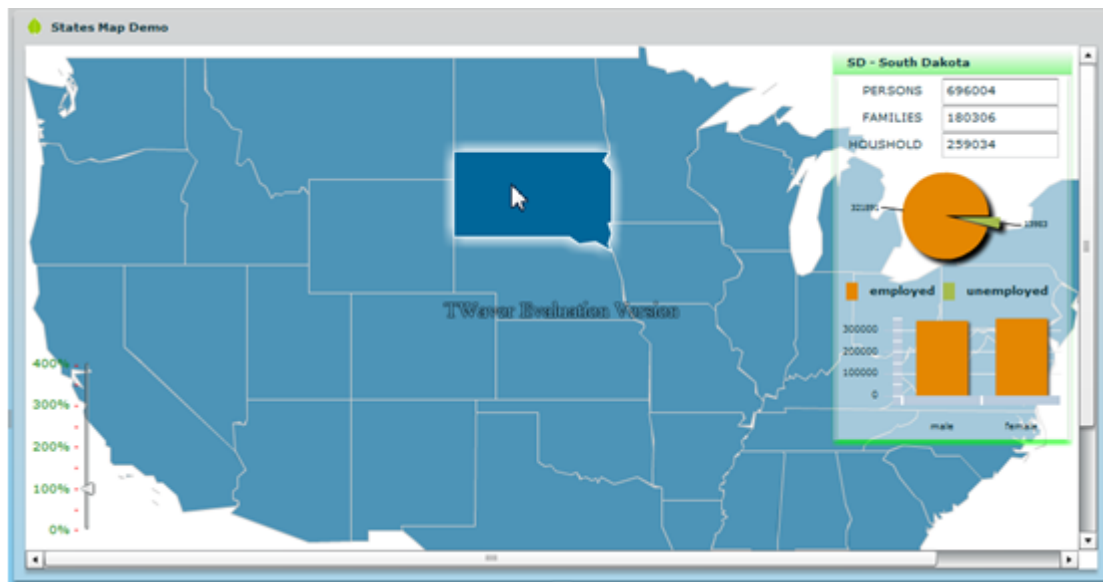
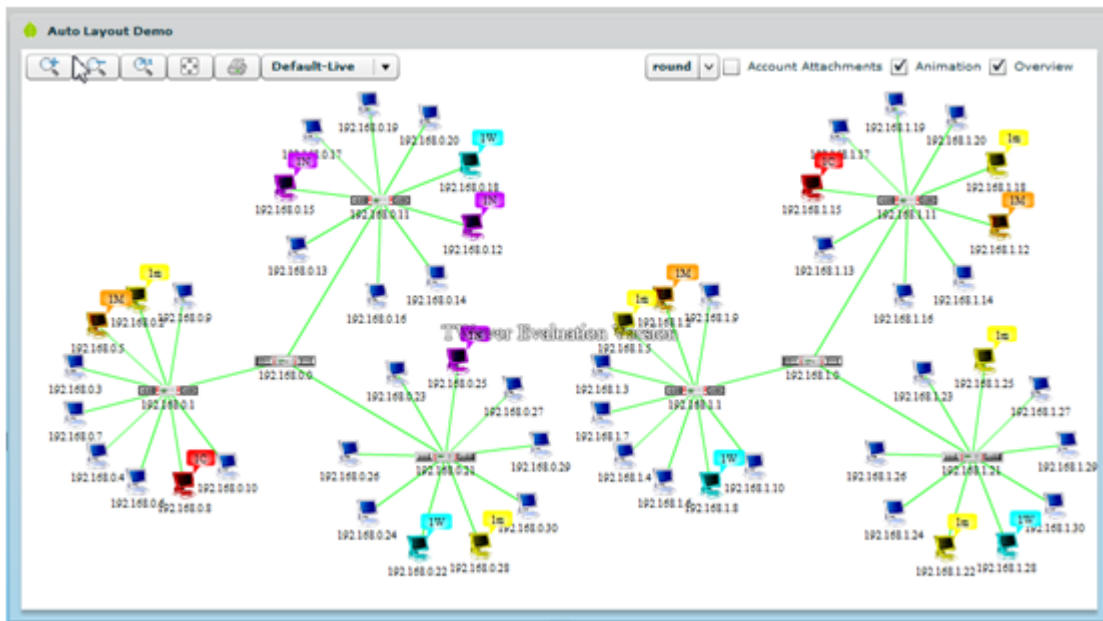
At a Glance

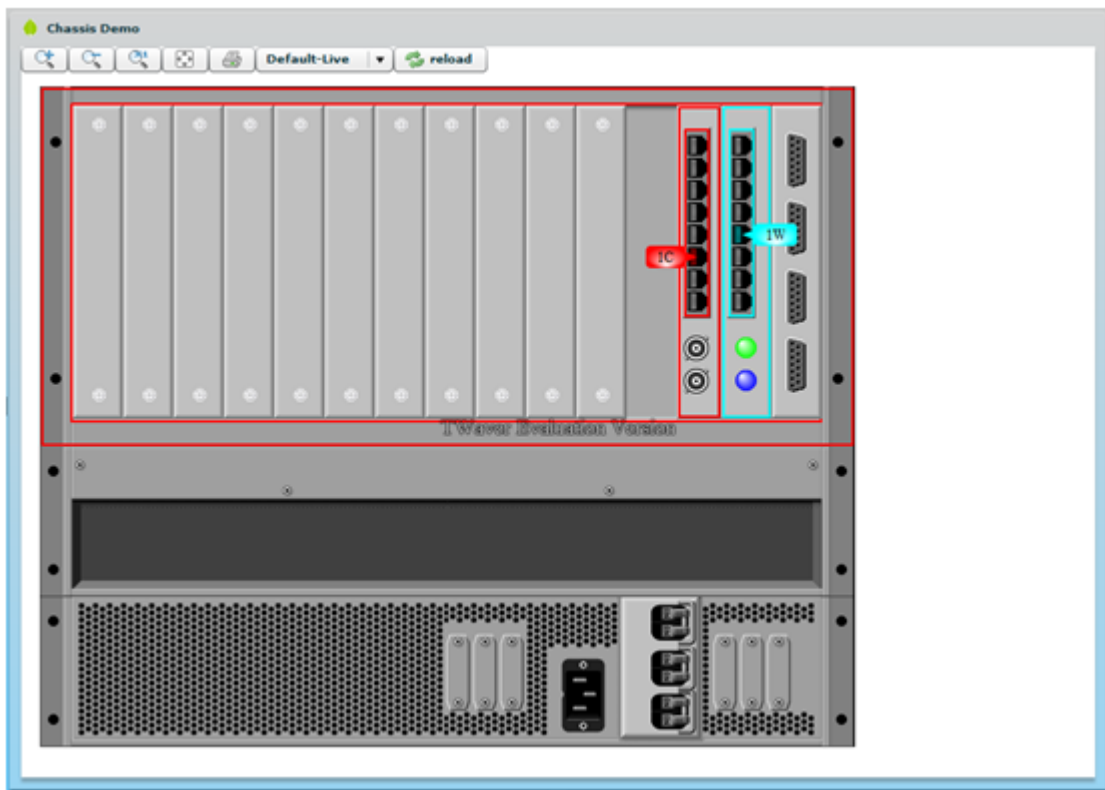
TWaver Flex consists of a series of visual graphic components. It offers a complete set of easy-to-use API that developers use to redevelop and customize. Data of TWaver Flex is maintained by DataBox designed to drive all visual components.

Network Component

Network component is the core component of TWaver Flex. It displays all kinds of topology, maps, equipment and etc. Displays below:







Tree Component

Tree Component is a customized tree which can display the hierarchy relationships of data. Display below:

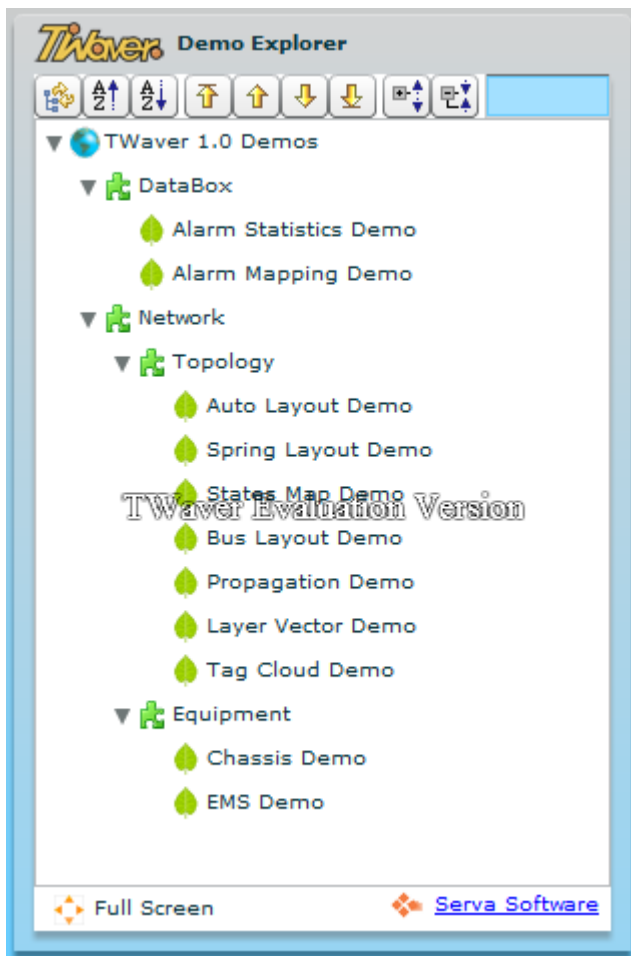


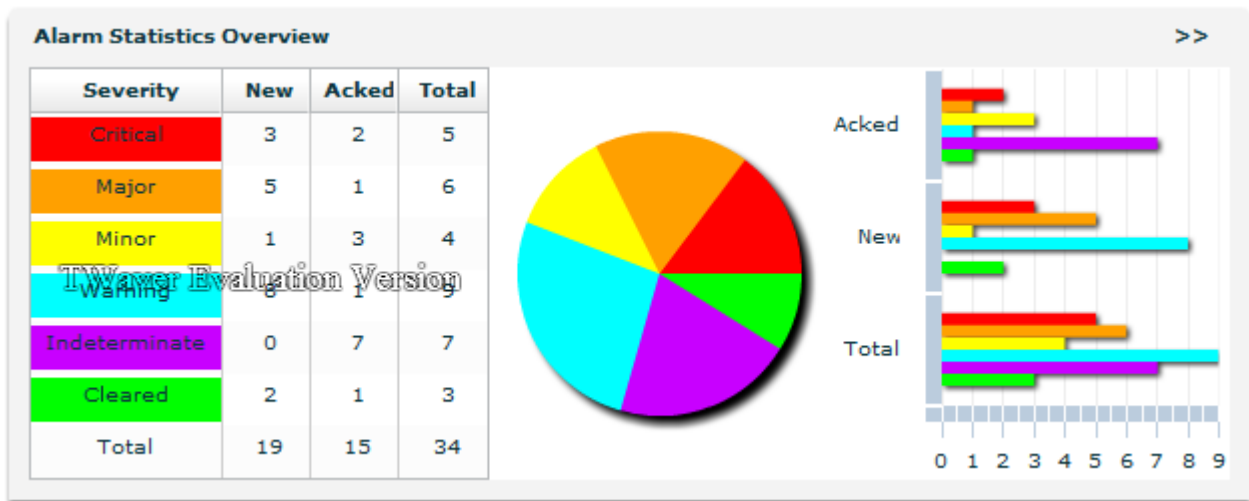
Table Component

Table component is a customized table which can display all properties of data.

| Mapping ID | Alarm Severity | IsAcked | IsCleared | Raised Time |
|------------|----------------|-------------------------------------|-------------------------------------|-----------------------------------|
| 1 | Indeterminate | <input type="checkbox"/> | <input type="checkbox"/> | Thu Aug 13 13:22:19 GMT+0800 2009 |
| 2 | Warning | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Thu Aug 13 13:22:19 GMT+0800 2009 |
| 3 | Minor | <input type="checkbox"/> | <input checked="" type="checkbox"/> | Thu Aug 13 13:22:19 GMT+0800 2009 |
| 4 | Major | <input type="checkbox"/> | <input type="checkbox"/> | Thu Aug 13 13:22:19 GMT+0800 2009 |
| 5 | Critical | <input checked="" type="checkbox"/> | <input type="checkbox"/> | Thu Aug 13 13:22:19 GMT+0800 2009 |
| | | | | |

AlarmOverview Component

AlarmOverview component provides a statistic and a display of alarm information with a series of graphics.



Getting Started

The following chapter is very important because it provides developers with core development information. It demonstrates how to build the programming environment, how to use TWaver Flex to create a new project, and how to initially utilize TWaver Flex components.

Chapter Content:

- [Convention](#)
- [TWaver Flex Quick Start](#)
- [Setup Environment](#)

Convention

There are name conventions for TWaver Flex and phrase comparison with TWaver Java and TWaver Silverlight/WPF. All these are to make you more easily understand this tutorial.

MVC - Model-View-Control model

Data Model: Correspond to Model of MVC. Such as IData, DataBox, ElementBox, AlarmBox, LayerBox, Element and etc in this document.

View: Correspond to View of MVC. Such as ElementUI, Network, Tree, Table and etc in this document.

Data Element: IData and its implemented class such as Data, Element, Node and etc.

Data Container: Data Element container, for example DataBox, ElementBox, AlarmBox, LayerBox.

The class name comparison between TWaver Java, TWaver Flex and TWaver Silverlight/WPF

| TWaver Flex | TWaver Silverlight/WPF | TWaver Java | Comment |
|-------------|------------------------|---------------|----------------------------|
| ElementBox | ElementBox | TDataBox | Network elements container |
| AlarmBox | AlarmBox | AlarmModel | Alarms container |
| LayerBox | LayerBox | LayerModel | Layers container |
| Network | Network | TNetwork | Topology component |
| Tree | Tree | TTree | Tree component |
| Table | NA | TElementTable | Table Component |

The element property comparison between TWaver Java, TWaver Flex and TWaver Silverlight/WPF

| TWaver Flex | TWaver Silverlight/WPF | TWaver Java | Comment |
|----------------|------------------------|------------------------|----------------|
| node.setStyle | node.SetStyle | node.putClientProperty | Style property |
| node.setClient | node.SetClient | node.putUserProperty | User property |

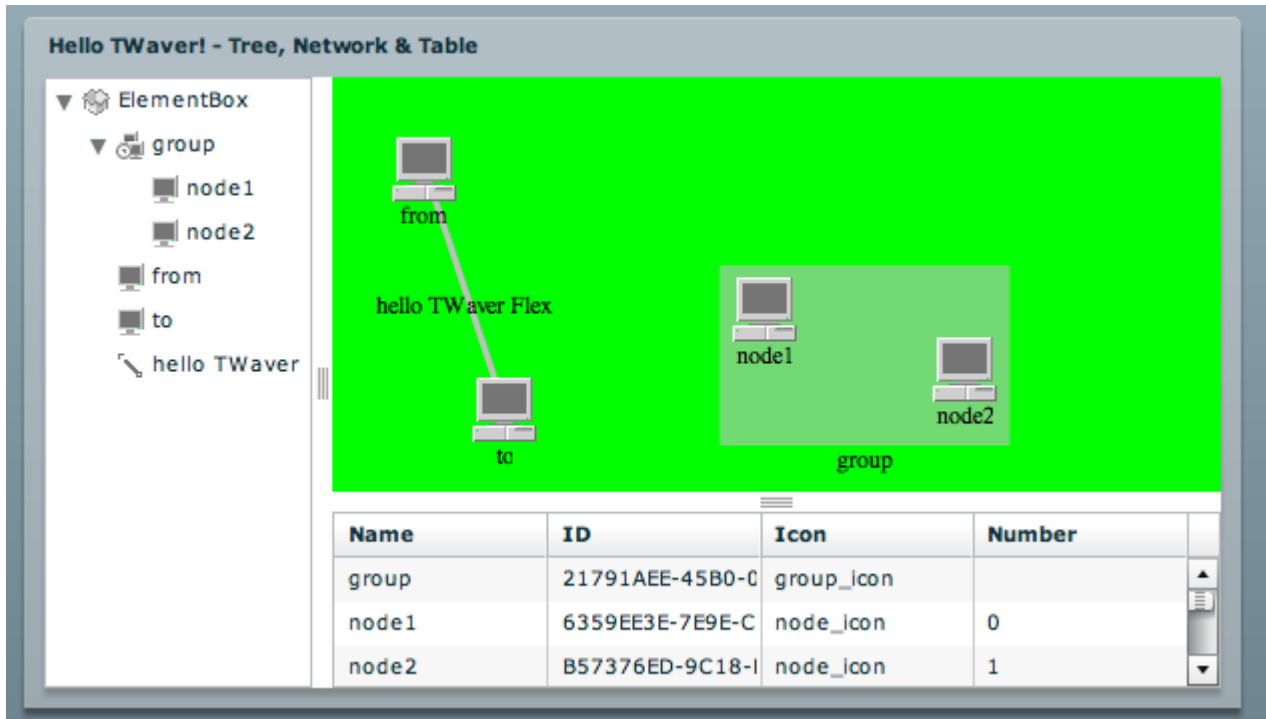
The filter comparison between TWaver Java, TWaver Flex and TWaver Silverlight/WPF

| TWaver Flex | TWaver Silverlight/WPF | TWaver Java | Comment |
|--------------------|------------------------|----------------------------|-----------------------------|
| alarmLabelFunction | AlarmLabelFunction | alarmLabelGenerator | Generator for alarm label |
| visibleFunction | VisibleFunction | visibleFilter | Visible filter |
| movableFunction | MovableFunction | movableFilter | Movable filter |
| editableFunction | EditableFunction | elementLabelEditableFilter | Editable filter |
| labelFunction | LabelFunction | elementLabelGenerator | Generator for element label |

| | | | |
|------------------------|------------------------|------------------------------|---------------------------------------|
| toolTipFunction | ToolTipFunction | elementToolTipTextGenerator | Generator for element tooltips |
| innerColorFunction | InnerColorFunction | elementBodyColorGenerator | Generator for body color of element |
| outerColorFunction | OuterColorFunction | elementOutlineColorGenerator | Generator for border color of element |
| selectColorFunction | SelectColorFunction | elementSelectColorGenerator | Generator for selected element |
| alarmFillColorFunction | AlarmFillColorFunction | alarmColorGenerator | Generator for alarm |
| alarmLabelFunction | AlarmLabelFunction | alarmLabelGenerator | Generator for alarm label |

TWaver Flex Quick Start

The best way to learn TWaver Flex is to make an example. Let's create an example application like below. This is a very simple application with a network view, tree and table components.



Now let's start step by step:

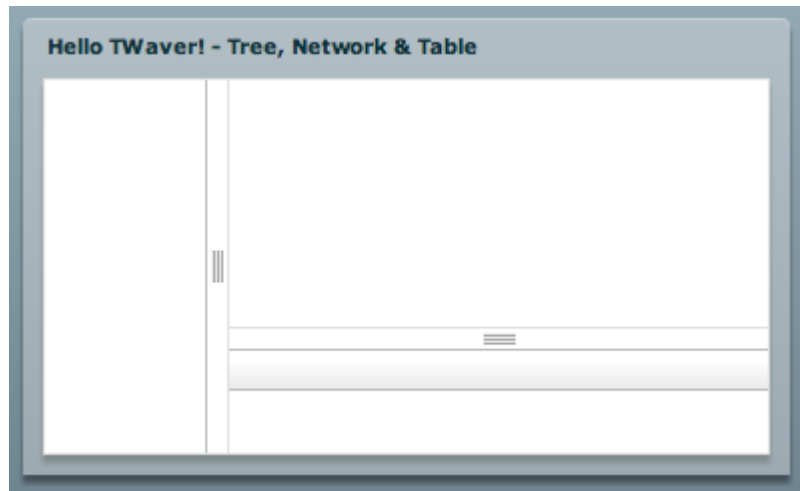
- First, create a new MXML Application called HelloTWaverFull.mxml, setup the namespace of TWaver:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex">
</mx:Application>
```

* (Depending on the layout) Add tree component to the left, topology to center, and the table below the topology:

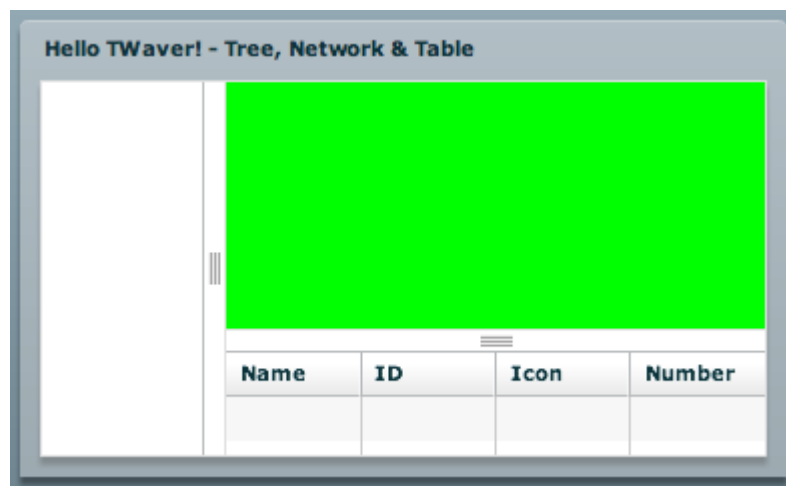
```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex">
<mx:Panel title="Hello TWaver! - Tree, Network & Table" width="100%" height="100%">
<mx:HDividedBox width="100%" height="100%">
<twaver:Tree id="tree" width="30%" height="100%"/>
<mx:VDividedBox width="100%" height="100%">
<twaver:Network id="network" width="100%" height="70%"/>
<twaver:Table width="100%" height="30%" id="table" editable="true">
</twaver:Table>
</mx:VDividedBox>
</mx:HDividedBox>
</mx:Panel>
```

```
</mx:Application>
```



- Setup background color of topology and add table columns:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex">
<mx:Panel title="Hello TWaver! - Tree, Network & Table" width="100%" height="100%">
<mx:HDividedBox width="100%" height="100%">
<twaver:Tree id="tree" width="30%" height="100%" />
<mx:VDividedBox width="100%" height="100%">
<twaver:Network id="network" backgroundColor="0x00ff00" width="100%" height="70%" />
<twaver:Table width="100%" height="30%" id="table" editable="true">
<twaver:columns>
<twaver:TableColumn dataField="name" headerText="Name" />
<twaver:TableColumn dataField="id" headerText="ID" />
<twaver:TableColumn dataField="icon" headerText="Icon" />
<twaver:TableColumn dataField="C:number" headerText="Number" />
</twaver:columns>
</twaver:Table>
</mx:VDividedBox>
</mx:HDividedBox>
</mx:Panel>
</mx:Application>
```



- Finally add listener of applicationComplete, fill in topology with network element:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex" applicationComplete="init()">
<mx:Script>
<![CDATA[
import twaver.*;
import twaver.network.Network;
private var box:ElementBox;
private var number:int;

private function init():void
{
    number=0;
    box=network.elementBox;
    tree.dataBox=box;
    table.dataBox=box;
    var group:Group=new Group();
    group.name="group";
    box.add(group);
    group.addChild(createTWaverNode("node1",200,100));
    group.addChild(createTWaverNode("node2",300,130));
    group.expanded=true;
    var from:Node=createTWaverNode("from",30,30);
    var to:Node=createTWaverNode("to",70,150);
    var link:Link=new Link(from,to);
    link.name="hello TWaver Flex";
    box.add(link);
}

private function createTWaverNode(name:String,x:int,y:int):Node
{
    var node:Node=new Node();
    node.name=name;
    node.setClient("number",number++);
    node.setLocation(x,y);
    box.add(node);
    return node;
}]]>
</mx:Script>
<mx:Panel title="Hello TWaver! - Tree, Network & Table" width="100%" height="100%">
<mx:HDividedBox width="100%" height="100%">
<twaver:Tree id="tree" width="30%" height="100%"/>
<mx:VDividedBox width="100%" height="100%">
<twaver:Network id="network" backgroundColor="0x00ff00" width="100%" height="70%"/>
<twaver:Table width="100%" height="30%" id="table" editable="true">
<twaver:columns>
<twaver:TableColumn dataField="name" headerText="Name"/>
<twaver:TableColumn dataField="id" headerText="ID"/>
<twaver:TableColumn dataField="icon" headerText="Icon"/>
<twaver:TableColumn dataField="C:number" headerText="Number"/>
</twaver:columns>
</twaver:Table>
</mx:VDividedBox>
</mx:HDividedBox>
</mx:Panel>
</mx:Application>
```

Displays below:

Hello TWaver! - Tree, Network & Table

▼ ElementBox

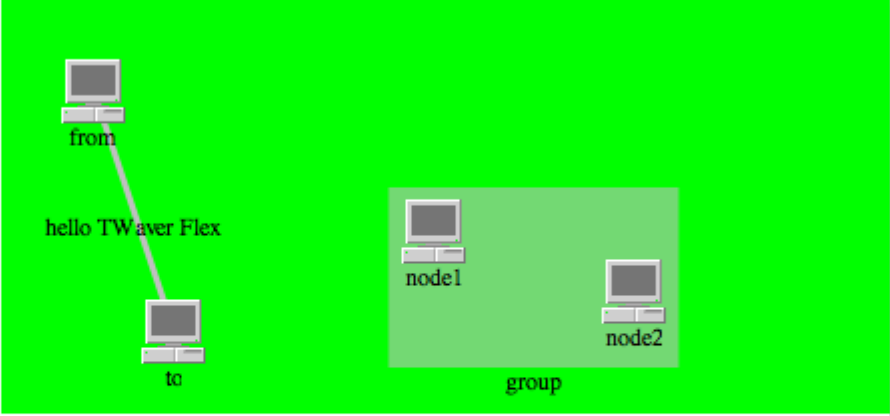
▼ group

- node1
- node2

from

to

hello TWaver



hello TWaver Flex

from

to

node1

node2

group

| Name | ID | Icon | Number |
|-------|-----------------|------------|--------|
| group | 21791AEE-45B0-0 | group_icon | |
| node1 | 6359EE3E-7E9E-C | node_icon | 0 |
| node2 | B57376ED-9C18-I | node_icon | 1 |

Setup Environment

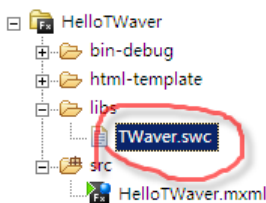
To use TWaver Flex, you need TWaver.swc library, this can be downloaded from Serva Software website www.servasoftware.com. Also you need an IDE. Flex Builder 3 or Flash Builder 4 is the best choice. You may like open source IDE like FlashDevelop 3+, this is ok. The following instruction will explain how to use TWaver.swc library in Flex Builder 3 and FlashDevelop 3.

• Flex Builder 3 As Development Tools

Flex Builder 3 is already accompanied by Flex 3 SDK and Debug Flash Player 9, users don't need to set these up.

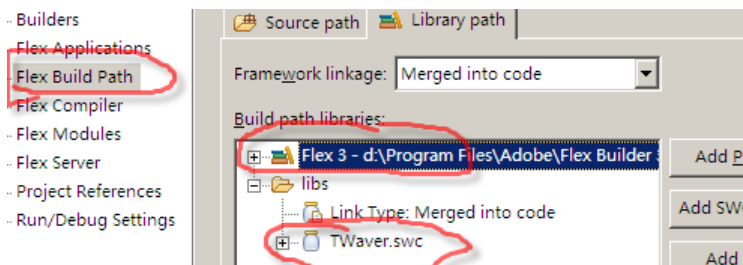
Create a New Project

Follow the step, File -> New -> MXML Application to create a new application : HelloTWaver. Put TWaver.swc file in libs directory. Flex Builder 3 will add all *.swc files under libs directory to compile class library automatically.



Add TWaver.swc to build path

Add TWaver.swc to build path as below, and users can add other library files or use other Flex SDK versions.



• FlashDevelop 3 As Development Tools

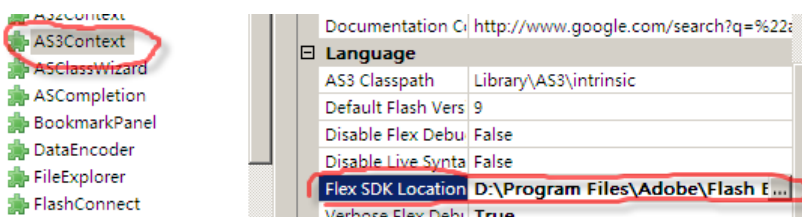
Setup Flex SDK and Debug Flash Player

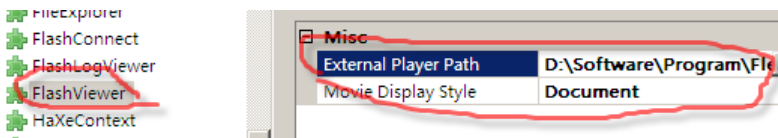
References:

<http://www.adobe.com/products/flex/>
<http://www.adobe.com/support/flashplayer/downloads.html>

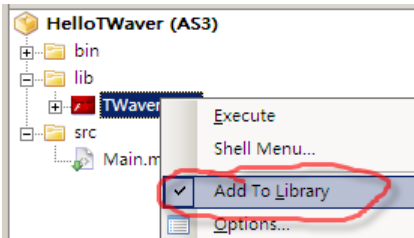
Add Flex SDK and Flash Player into FlashDevelop

FlashDevelop 3 has set up the path of Flex SDK and Debug Flash Player, setup value as below. Please refer to official website www.flashdevelop.org for detailed introductions.





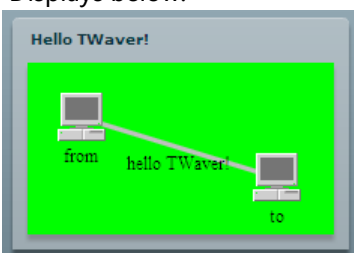
Create a Flex 3 Project



- Create first application

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:tw="http://www.servasoftware.com/2009/twaver/flex"
    applicationComplete="init()">
  <mx:Script>
    <![CDATA[
      import flash.geom.Point;
      import twaver.*;
      import twaver.network.Network;
      private function init():void{
        var box:ElementBox = network.elementBox;
        var from:Node = new Node();
        from.name = "from";
        from.location = new Point(20, 20);
        box.add(from);
        var to:Node = new Node();
        to.name = "to";
        to.location = new Point(150, 60);
        box.add(to);
        var link:Link = new Link(from, to);
        link.name = "hello TWaver!";
        box.add(link);
      }
    ]]>
  </mx:Script>
  <mx:Panel title="Hello TWaver!" width="100%" height="100%">
    <tw:Network id="network" backgroundColor="0x00ff00" width="100%" height="100%"/>
  </mx:Panel>
</mx:Application>
```

Displays below:



Basic Knowledge

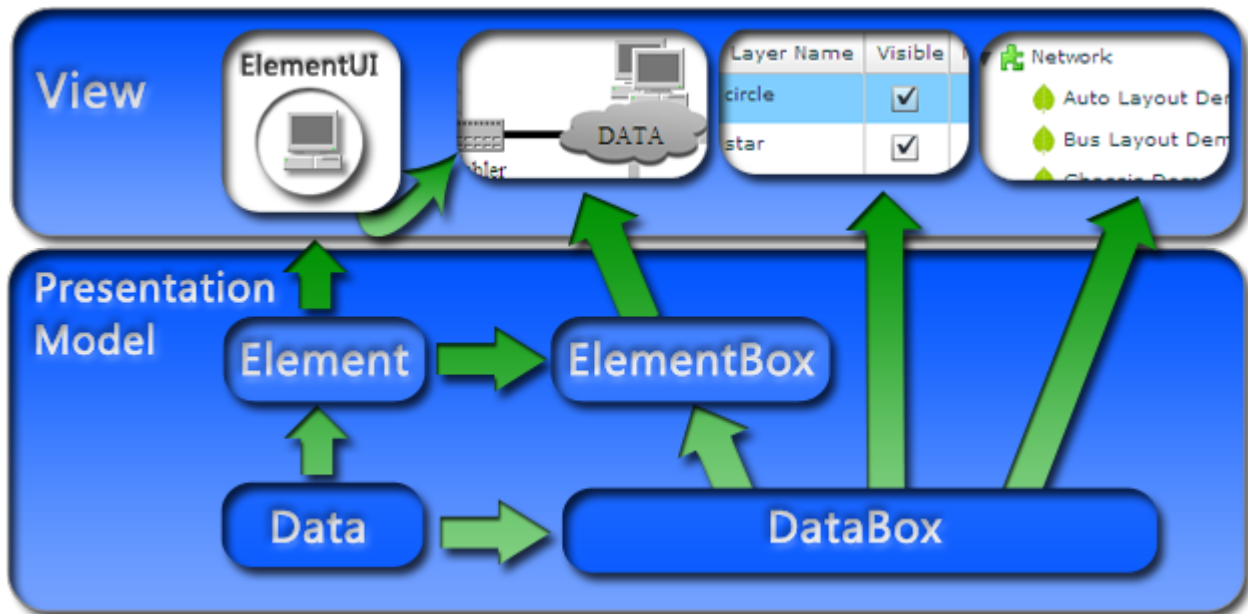
This chapter introduces the UI components and design model of TWaver Flex. Also included is a brief overview of the data model and the types of visual components. After reading this chapter you will know the principles of TWaver Flex.

- [The Design Model and Frame of TWaver Flex](#)
- [The Data Model of TWaver Flex](#)
 - [The Data Element of TWaver Flex](#)
 - [The Data Container of TWaver Flex](#)
- [The View Components of TWaver Flex](#)
 - [Network Introduction](#)
 - [Table Introduction](#)
 - [Tree Introduction](#)
- [Data Serialization](#)

The Design Model and Frame of TWaver Flex

TWaver Flex is based on MVC (Model-View-Controller) model. The data model of the client side is a nesting MV as a group. The basic data element is `twaver.Data`, and `twaver.DataBox` is the data container. Up to the application layer, `twaver.network.ui.ElementUI` will serve as the basic component, and `twaver.Network`, `twaver.Tree`, `twaver.Table` and etc will serve as the data container components. This is the frame of TWaver Flex.

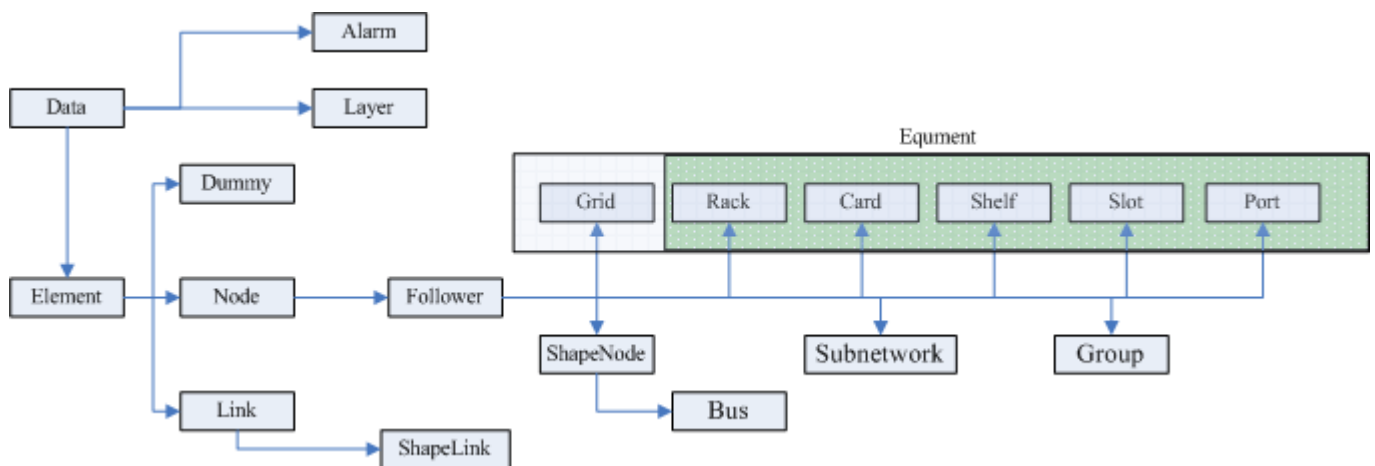
TWaver Design Model



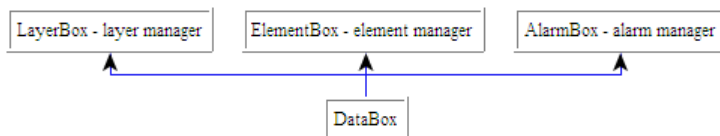
The Data Model of TWaver Flex

Base on two basic elements, which `twaver.IData` and `twaver.DataBox` correspond to be the basic data element and data container, TWaver Flex predefined series of business objects, network elements and data containers. For example alarm element (`twaver.IAlarm`) and alarm container (`twaver.AlarmBox`), view layer (`twaver.ILayer`) and view layer container (`twaver.LayerBox`), topology element (`twaver.IElement`) and topology container (`twaver.ElementBox`)...

The structure of data elements



The structure of data containers



Topology container (`twaver.ElementBox`) integrates other containers, provides abundance topology elements including Dummy, Node, Link, Bus, ShapeNode, ShapeLink, Follower, Rack, Shelf, Slot, Card, Port, Grid, Group, SubNetwork and etc, offers strong support for the design model and business function of webmaster interface developing.

- [The Data Element of TWaver Flex](#)
- [The Data Container of TWaver Flex](#)

The Data Element of TWaver Flex

TWaver Flex takes `twaver.IData` as the basic data element, extends series of data element with GUI and business logic including `IAAlarm`, `ILayer`, `IElement`...

- **twaver.IData**

`IData` is the basic interface of TWaver Flex data element, take `twaver.Data` as its implementation class. It defined its own properties such as `id`, `name`, `icon`, `toolTip`, `parent`, `children` and etc, support importing and exporting xml data which lay foundation of data serialization between each TWaver platform.

`Data` implements `flash.events.EventDispatcher` which gives its the event dispatching and listening function. It realizes event dispatching and listening by following methods:

```
flash.events.EventDispatcher#public dispatchEvent(event:Event):Boolean
flash.events.EventDispatcher#addEventListener(type:String, listener:Function,
    useCapture:Boolean=false, priority:int=0, useWeakReference:Boolean=false):void
```

`Data` serve `twaver.IData` as its interface, realizes the dispatching and listening of property change.

```
function dispatchPropertyChangeEvent(property:String, oldValue:Object,
    newValue:Object):Boolean;
function addPropertyChangeListener(listener:Function, priority:int = 0,
    useWeakReference:Boolean = false):void;
function removePropertyChangeListener(listener:Function):void;
```

More, `Data` owns other functions:

```
function get childrenCount():int;
function get hasChildren():Boolean;
function isDescendantOf(data:IData):Boolean;
function isParentOf(data:IData):Boolean;
function isRelatedTo(data:IData):Boolean;

function toXML(context:XMLContext, newInstance:IData):void;
function parseXML(context:XMLContext, xml:XML):void;

function setPropertyValue(property:String, value:Object):void;
function getPropertyValue(property:String):Object;
```

Let's introduce the implementation classes one by one:

- **twaver.ILayer**

`Layer`, which is the layer management component of TWaver, take `twaver.ILayer` as interface, has three special properties: `visible`, `editable`, `movable`. `LayerBox` is to maintain all relations of TWaver Flex layers, the default order is decided by father-children relationship and adding indexes. In topology, the view of each network is decided by layer ID of corresponding element.

• twaver.IAlarm

Alarm, which is the data model of equipment fault and network exception of webmaster system, take Alarm as its implementation class. The connection between Element and Alarm can response the alarm status of network. Alarm owns critical levels to express whether cleared, acknowledged and etc.

| 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 |

AlarmBox is to manage all alarm issues in TWaver. Alarm and Network can be mutual reference directly, but can be connected by AlarmBox. AlarmState can be referenced by Network to express the level and quality of new alarm events.

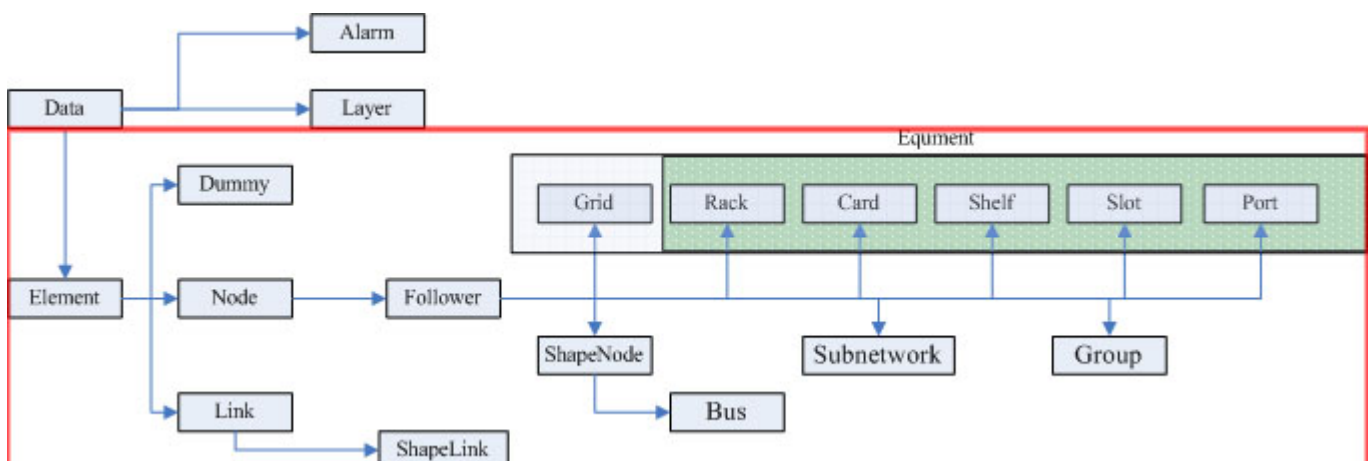
• twaver.IElement

IElement is the most important data element in TWaver which takes Element as its implementation class. Element is to be the object of topology such as Node, Link, SubNetwork, Bus, Card and etc.

TWaver predefined abundance kinds of Network such as Dummy, Node, Link, Bus, ShapeNode, ShapeLink, Follower, Rack, Shelf, Slot, Card, Port, Grid, Group, SubNetwork and etc. Each network correspond one ElementUI class, and the network present its style, these two consists of the MV model.

We can demonstrate the effectiveness and various of presentations by setting the properties and style of Network. At the same time, user can extends all these predefined Elements to cope with special requirement.

ElementBox is to maintain all data of Element. It can drive multiple views such as twaver.Network, twaver.Tree, twaver.Table.



Dummy

It is invisible in topology, but visible in Tree and Table. It is usually used to group logical things which no topology significance.

Node

It is the basic class of other nodes, stands for one node in topology.

Link

It is the basic class of other links, stands for link in topology.

Follower

It stands for follower, can be attached in another node. Once the host moves, Follower will follow up.

Bus

It implements ShapeNode, is one kind of layout node. It can do Bus Layout with other nodes in it.

ShapeNode

Its shape is determined by a series of control point. It present as abundant styles.

ShapeLink

It implements Link but different with Link. Its direction is determined by a series of control point, and it can customize special Connection Layout.

Grid

It is the basic class of Rack, Shelf, Slot, Card and Port, stands for equipment panel. It will display as grid in topology, and can change columns and rows.

Group

It stands for a group, contains child Network. It can be expended or combined, the location and scope of children will decide the location and scope of Group.

SubNetwork

SubNetwork means a lot for topology. In topology will not show all Networks once generally, but only show Networks in current SubNetwork. It can resolve large data volume problem by switching SubNetwork and data lazy loading.

Rack

It stands for rack in equipment panel.

Shelf

It stands for shelf in equipment panel.

Slot

It stands for slot in equipment panel.

Card

It stands for card in equipment panel.

Port

It stands for port in equipment panel.

The Data Container of TWaver Flex

Data Management Container is a container to maintain data. DataBox in TWaver Flex is Data Management Container, play an important role as Model in TWaver Flex MVC model. One DataBox can drive multiple views, and the data change of DataBox can be shown its related view components. DataBox of TWaver Flex supports views including twaver.controls.Tree, twaver.controls.Table. ElementBox of TWaver Flex has its own special view component twaver.network.Network.

- **DataBox**

DataBox implements EventDispatcher, adds listener to data change and container change. User can master all Element transformation by listening DataBox, and can handle the event by rewriting on***Changed function:

```
public function addDataBoxChangeListener(listener:Function, priority:int = 0
, useWeakReference:Boolean = false):void
public function removeDataBoxChangeListener(listener:Function):void
public function addDataPropertyChangeListener(listener:Function,
priority:int = 0, useWeakReference:Boolean = false):void
public function removeDataPropertyChangeListener(listener:Function):void
public function addPropertyChangeListener(listener:Function, priority:int = 0
, useWeakReference:Boolean = false):void
public function removePropertyChangeListener(listener:Function):void
public function addHierarchyChangeListener(listener:Function, priority:int = 0
, useWeakReference:Boolean = false):void
public function removeHierarchyChangeListener(listener:Function):void
protected function onDataPropertyChanged(data:IData, e:PropertyChangeEvent):void
protected function onClientChanged(styleProp:String, oldValue:*, newValue:*):
```

One container needs to have way to maintain data alternation. User can add/remove data element by following methods in DataBox:

```
public function add(data:IData, index:int = -1):void
public function remove(data:IData):void
public function removeByID(id:Object):void
public function clear():void
public function getDataByID(id:Object):IData
public function contains(data:IData):Boolean
```

There are iterate methods in DataBox as below:

```
public function forEach(callbackFunction:Function):void
public function forEachByDepthFirst(callbackFunction:Function, data:IData = null):void
private function depthFirst(callbackFunction:Function, data:IData):void
public function forEachByBreadthFirst(callbackFunction:Function, data:IData = null):void
```

DataBox maintains all layers of data Elements. The following methods is to do moving or inserting operation:

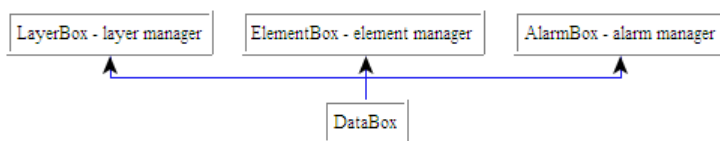
```
public function moveUp(data:IData):void
public function moveDown(data:IData):void
public function moveToTop(data:IData):void
public function moveToBottom(data:IData):void
```

```
public function moveTo(data:IData, newIndex:int):void
```

More, DataBox packed the selection mechanism of data Element, realized the SelectionModel of Data, offer the simple operation approach

```
public function get selectionModel():SelectionModel
public function moveSelectionUp(sm:SelectionModel = null):void
public function moveSelectionDown(sm:SelectionModel = null):void
public function moveSelectionToTop(sm:SelectionModel = null):void
public function moveSelectionToBottom(sm:SelectionModel = null):void
```

TWaver Flex defined LayerBox to do layer management, AlarmBox to do alarm management, ElementBox to do network management. And the LayerBox and AlarmBox service for ElementBox, used to maintain the layer and alarm information of network.



- LayerBox

LayerBox is the layer management container, need to be attached to one ElementBox. It defined methods to add/remove layers, and one default layer:

```
function LayerBox(elementBox:ElementBox)
public function get elementBox():ElementBox
public function get defaultLayer():ILayer
```

* h3. AlarmBox

AlarmBox is to maintain alarm information, also needs to be attach one ElementBox:

```
function AlarmBox(elementBox:ElementBox)
public function get elementBox():ElementBox
```

AlarmBox also defined the AlarmElementMapping to manage the matchup between network and alarm information. User can customize the relevance to implement requirement such as one alarm effect multi network:

```
public function get/set alarmElementMapping():IAlarmElementMapping
```

More, We still offer many properties:

```
//When network is removed from ElementBox, whether the related alarm need to be deleted
removeAlarmWhenElementIsRemoved
//Whether need to remove Alarm when the alarm level is Cleared
removeAlarmWhenAlarmIsCleared
```

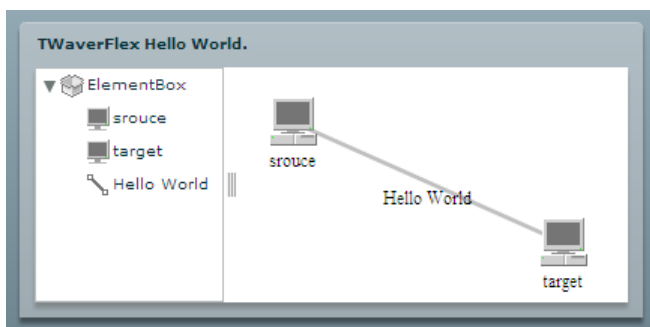
* h3. ElementBox

ElementBox includes AlarmBox, LayerBox, is the network management container. TWaver Flex defined proper view components for ElementBox to show the topology relationships.

ElementBox HelloWorld application:

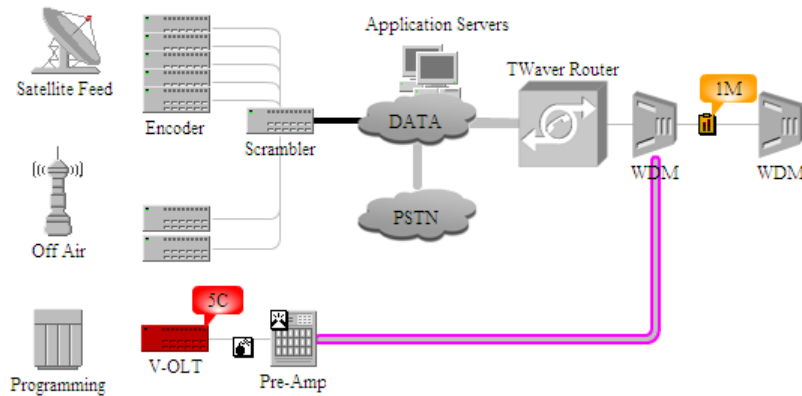
```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
  layout="vertical" pageTitle="TWaverFlex" creationComplete="init()">
  <mx:Script>
    <![CDATA[
      import twaver.Link;
      import twaver.Node;
      import twaver.ElementBox;
      private function init():void{
        var box:ElementBox=new ElementBox();
        var node1:Node=new Node();
        node1.name="source";
        node1.setLocation(20,20);
        box.add(node1);
        var node2:Node=new Node();
        node2.name="target";
        node2.setLocation(200,100);
        box.add(node2);
        var link:Link=new Link(node1,node2);
        link.name="Hello World";
        box.add(link);
        network.elementBox=box;
        tree.dataBox=box;
      }
    ]]>
  </mx:Script>
  <mx:Panel title="TWaverFlex Hello World." width="100%" height="100%">
    <mx:HDividedBox width="100%" height="100%">
      <twaver:Tree id='tree' width="30%" height="100%"/>
      <twaver:Network id="network" width="70%" height="100%"/>
    </mx:HDividedBox>
  </mx:Panel>
</mx:Application>
```

It runs as below:

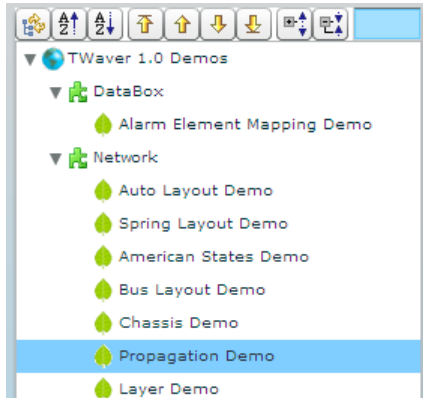


The View Components of TWaver Flex

TWaver Flex extends Flex components, defined components including Network, Table, Tree and etc driving by DataBox. Network is GUI components to show all topology relations, Table and Tree are based on DataGrid and Tree in Flex, All these components packed more properties and offer more convenient methods to let user use TWaver Flex.



| Gradient | Gradient Colo | Gradient Alpha | Outline Width | Outline Color | Outline Alpha |
|----------------|---------------|----------------|---------------|---------------|---------------|
| radial.northwe | FFFFFF | 1 | 2 | CCEECC | 1 |
| radial.northwe | FFFFFF | 1 | 0 | CCEECC | 1 |
| radial.northwe | FFFFFF | 1 | 2 | CCEECC | 1 |
| spread.east | FFFFFF | 1 | 0 | CCEECC | 1 |



- [Network Introduction](#)
- [Table Introduction](#)
- [Tree Introduction](#)

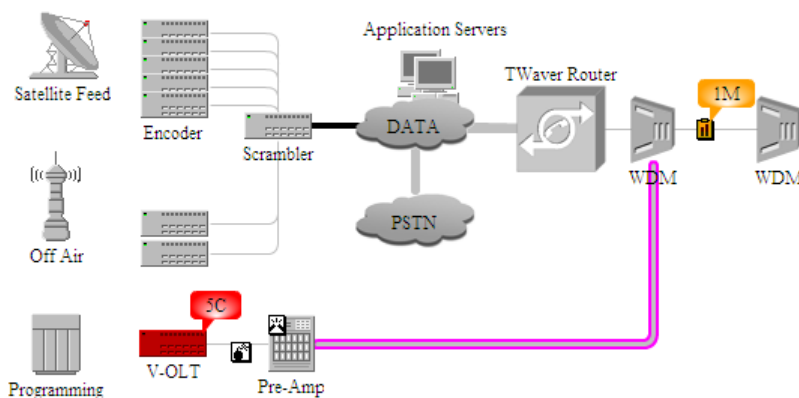
Network Introduction

• Network Overview

Network component is used to present network information in graphical way, has many functions such as element filter, SubNetwork switching, background supporting, zoom viewing, multi layers, attachment components showing and etc. It is the most intuitive and most shapely platform for network master system. More than that, Network still offers hundreds of properties and functions to let user to extend and configure, meet the special requirements.

Let's show one sample of Network to learn common features. The detail will be introduced in subsequence chapters.

We can make a topology picture as below easily by using Network:



• The Model and Hierarchy of Network Component

Network implements Canvas of flex, includes many containes to place different components.

The hierarchy of Network:

```

Network
rootCanvas // root panel
topCanvas // top panel
attachmentCanvas // attachment panel
layerCanvas // network view panel, to put ElementUI
layer n // the hierarchy of ElementUI is maintained by ElementBox.layerBox
layer ...
layer 0
bottomCanvas // bottom panel
backgroundCanvas// backgroud panel
    
```

Network in Topology

Each Network element is to correspond with one ElementUI view component, Network express each view component by operating network element data. The method at below is the way to get correspond component of network element.

```

public function getElementUI(element:IElement):ElementUI
    
```

Attachment Component

Network is not only related to ElementUI component, but also can be bounded with multi Attachment components such as network element tags, alarm bubbles. User can add new Attachment into network element to express itself more completely.

```
twaver.network.ui.Attachment
//Attachment constructor function
public function Attachment(elementUI:ElementUI, showInAttachmentCanvas:Boolean= false)

twaver.network.Network
//attachment canvas in network, Attachment components can be added into ElementUI component
//Attachment also can be detached with ElementUI component, can be added into attachment canvas directly
which will make sure attachment top show.
public function get attachmentCanvas():Canvas

twaver.network.ui.ElementUI
//ElementUI can get all attachments of network element
public function get attachments():ICollection
```

- Network Commonly Features

Switch Interactive Model

Switching interactive model in Network can make the interactive of different mouse events in different models come true. The most commonly use model is Default Model and Edit Model. User can handle mouse and key events by implementing InteractionHandler as interface, thus, user can customize own interactive model. All these InteractionHandler can be used alternately and draw up together.

```
//switch to Default Interaction Model
public function setDefaultInteractionHandlers(lazyMode:Boolean= false):void//switch to Edit Interaction Model
public function setEditInteractionHandlers(lazyMode:Boolean= false):void//switch to Create Interaction Model
public function setCreateNodeInteractionHandlers(nodeClass:Class =null):void//switch to Create Link Interaction
Model
public function setCreateLinkInteractionHandlers(linkClass:Class =null):void//switch to Create ShapNode Interaction
Model
public function setCreateShapeNodeInteractionHandlers(shapeNodeClass:Class =null):void
```

Filter

```
network.visibleFunction = function(node:IElement):Boolean{
  return node.childrenCount > 0;
};
```

Some Properties Listed

There are hundreds of properties of Network, let's list parts of them:

```
//set current SubNetwork
    public function get/setcurrentSubNetwork():ISubNetwork
    //up into higher level SubNetwork
    public function upSubNetwork(animate:Boolean= false):void     //mouse and keyboard add/remove
interaction listener
    public function addInteractionListener(listener:Function, priority:int= 0, useWeakReference:Boolean= false):void
    public function removeInteractionListener(listener:Function):void     //refresh view of network element
    public function invalidateElementUI(element:IElement):void //get view of network element
```

```
public function getElementUI(element:IElement):ElementUI

//zoom operation
public function get zoom():NumberpublicfunctionzoomIn():void
public function zoomOut():void
public function zoomReset():void
public function zoomOverview():void  //get container of components in topology
public function get rootCanvas():Canvas
public function get topCanvas():Canvas
public function get attachmentCanvas():Canvas
public function get layerCanvas():Canvas
public function get bottomCanvas():Canvas
public function get backgroundCanvas():Canvas
```


Table Introduction

twaver.controls.Table implements DataGrid of Flex, is bound with DataBox. It can be view compoment for any DataBox Model including LayerBox, AlarmBox and ElementBox.

twaver.controls.Table make columns configuration and operation more simple, and can express network element properties and style more convinient. More, twaver.controls.Table extends some Renderer and Editor, these make user more brief to create table, do data showcase.

twaver.controls.Table sample:

| Gradient | Gradient Color | Gradient Alpha | Outline Width | Outline Color | Outline Alpha |
|----------------|----------------|----------------|---------------|---------------|---------------|
| radial.northwe | FFFFFF | 1 | 2 | CCEECC | 1 |
| radial.northwe | FFFFFF | 1 | 0 | CCEECC | 1 |
| radial.northwe | FFFFFF | 1 | 2 | CCEECC | 1 |
| spread.east | FFFFFF | 1 | 0 | CCEECC | 1 |

The use of Table and DataGrid is the same:

```
<tw:Table id="table1" width="100%" height="100%" editable="true">
  <tw:columns>
    <twaver:TableColumn dataField="name" headerText="Name"/>
    <twaver:TableColumn dataField="x" headerText="X" />
    <twaver:TableColumn dataField="y" headerText="Y" />
    <twaver:TableColumn dataField="C:number" headerText="Number" />
  </tw:columns>
</tw:Table>
```

twaver.controls.Table made improvement on columns for Flex that is what we need to pay attention to. For example: dataField="C:number" stands for getting and modifying cell unit value by invoking item.getClient("number"), item.setClient("number",value), thus, twaver.controls.Table show properties and style of network element more convinient.

Filter:

```
table.visibleFunction = function(data:IData):Boolean{
  retron data.childrenCount > 0;
};
```

Let's give an example for customizing column and renderer separately:

Customizing Column:

```
package table
{
  import twaver.controls.TableColumn;
  import mx.core.ClassFactory;

  public class DataColumn extends TableColumn
  {
    public function DataColumn(columnName:String=null)
    {
      super(columnName);
      this.itemRenderer = new ClassFactory(DataRenderer);
      this.editable = false;
      this.dataField = "id";
      this.headerText = "Element";
    }
  }
}
```

```
}
}
```

Customizing renderer

```
package table
{
    import flash.display.BitmapData;
    import flash.geom.Matrix;

    import mx.containers.HBox;
    import mx.controls.TextInput;
    import mx.core.UIComponent;

    import twaver.IData;
    import twaver.Utills;
    import twaver.controls.Table;

    public class DataRenderer extends HBox
    {
        public static const size:Number = 18;
        private var image:UIComponent = new UIComponent();
        private var textInput:TextInput = new TextInput();

        public function DataRenderer()
        {
            textInput.percentWidth = 100;
            this.addChild(image);
            this.addChild(textInput);
            this.horizontalScrollPolicy = "off";
            this.verticalScrollPolicy = "off";
            this.textInput.mouseEnabled = false;
            this.textInput.mouseChildren = false;
            this.textInput.setStyle("borderStyle", "none");
            this.textInput.setStyle("backgroundAlpha", 0);
            this.setStyle("paddingLeft", 3);
            this.setStyle("paddingRight", 3);
            this.setStyle("horizontalGap", 2);
        }

        override public function set data(value:Object):void {
            super.data = value;
            var d:IData = null;
            if(this.data is IData){
                d = this.data as IData;
            }
            else if(this.data && this.owner is Table){
                d = Table(this.owner).dataBox.getDataByID(this.data.id);
            }
            image.graphics.clear();
            if(d != null){
                this.textInput.text = Utills.getQualifiedClassName(d);
                var bd:BitmapData = Utills.getImageAsset(d.icon).getBitmapData();
                var matrix:Matrix = new Matrix();
                matrix.scale(size/Number(bd.width), size/Number(bd.height));
                image.graphics.beginBitmapFill(bd, matrix);
                image.graphics.drawRect(0, 0, size, size);
                image.graphics.endFill();
                image.width = size;
                image.height = size;
            }
            else{
                this.textInput.text = label;
                image.width = 0;
            }
        }
    }
}
```

```

        image.height = 0;
    }

}
}
}

```

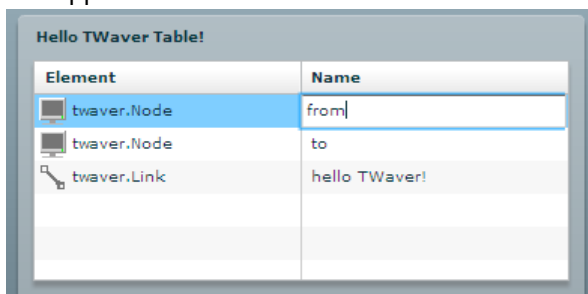
The following codes customized a table with two columns which one cloumn maintain the element in DataBox, another column is the name of corresponding element. And make the "name" column editable:

```

<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:tw="http://www.servasoftware.com/2009/twaver/flex"
    xmlns:demo="table.*"
    applicationComplete="init()">
    <mx:Script>
    <![CDATA[
        import twaver.*;
        private function init():void{
            var box:DataBox = table1.dataBox;
            var from:Node = new Node();
            from.name = "from";
            from.location = new Point(20, 20);
            box.add(from);
            var to:Node = new Node();
            to.name = "to";
            to.location = new Point(150, 60);
            box.add(to);
            var link:Link = new Link(from,to);
            link.name = "hello TWaver!";
            box.add(link);
        }
    ]]>
    </mx:Script>
    <mx:Panel title="Hello TWaver Table!" width="100%" height="100%">
        <tw:Table id="table1" width="100%" height="100%" editable="true">
            <tw:columns>
                <demo:DataColumn/>
                <twaver:TableColumn dataField="name" editable="true" headerText="Name"/>
            </tw:columns>
        </tw:Table>
    </mx:Panel>
</mx:Application>

```

The application run as below:



Tree Introduction

twaver.controls.Tree implements Tree component of Flex, can be bounded with any DataBox containers such as LayerBox, AlarmBox, ElementBox and etc. The hierarchy of Tree view is decided by the father-child relationships in DataBox container. Tree view own itself selection model, also can use the one of DataBox, in this way, the view and the DataBox container can achieve a very good effect in steps.

twaver.controls.Tree is pretty easy to use, once related to DataBox container, it can express the hierarchy of this container.

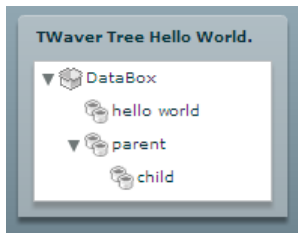
Filter:

```
tree.visibleFunction = function(data:IData):Boolean{
    return data.childrenCount > 0;
};
```

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
layout="vertical" pageTitle="TWaverFlex" creationComplete="init()">
<mx:Script>
<![CDATA[
import twaver.Link;
import twaver.Data;
import twaver.DataBox;
private function init():void{
    var box:DataBox = new DataBox();
    var node:Data = new Data();
    node.name = "hello world";
    box.add(node);
    var parentNode:Data = new Data();
    parentNode.name = "parent";
    box.add(parentNode);
    var child:Data = new Data();
    child.name = "child";
    child.parent = parentNode;
    box.add(child);
    tree.dataBox = box;
    tree.callLater(function():void{
        tree.expandChildrenOf(tree.rootTreeData, true);
    });
}
]]>
</mx:Script>
<mx:Panel title="TWaver Tree Hello World." width="100%" height="100%">
<twaver:Tree id="tree" width="100%" height="100%"/>
</mx:Panel>
</mx:Application>
```

The application run as below:



Data Serialization

TWaver Flex supports importing and exporting data container which facilitate data saving and transmission. TWaver Flex defined the serialize/deserialize methods by XML Serializer for DataBox to accomplish this job.

Construct a XML

```
var xmlSerializer:XMLSerializer = new XMLSerializer(box);
```

Serialize and Deserialize

```
public function serialize():String
public function deserialize(xmlString:String, rootParent:IData = null):void
```

XML Serializer also can appoint properties to output or not. Let's give an example for not outputting Element ID:

```
xmlSerializer.settings.registerProperty("id",null);
```

The test application:

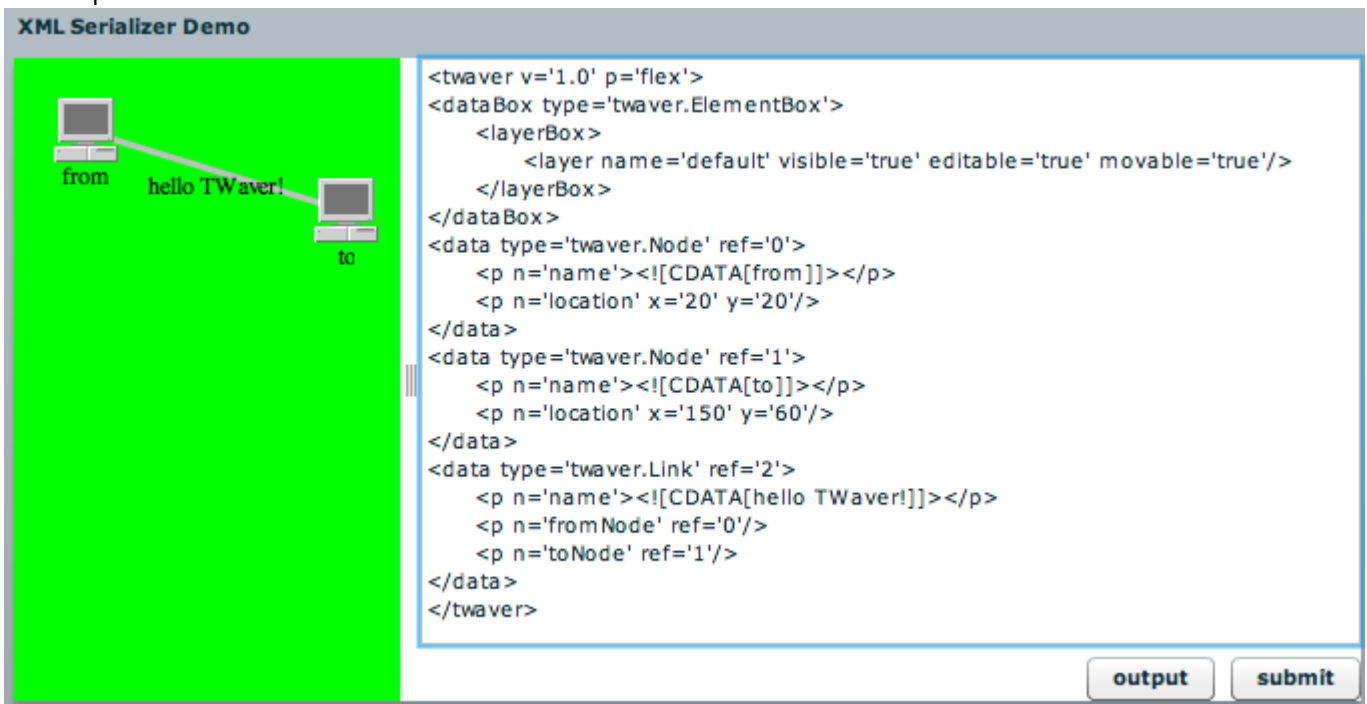
```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
    applicationComplete="init()">
    <mx:Script>
        <![CDATA[
            import twaver.*;
            import twaver.network.Network;
            private var box:ElementBox;
            private var xmlSerializer:XMLSerializer;
            private function init():void{
                box = network.elementBox;
                xmlSerializer=new XMLSerializer(box);
                //not output element "id"
                xmlSerializer.settings.registerProperty("id",null);
                var from:Node = new Node();
                from.name = "from";
                from.location = new Point(20, 20);
                box.add(from);
                var to:Node = new Node();
                to.name = "to";
                to.location = new Point(150, 60);
                box.add(to);
                var link:Link = new Link(from,to);
                link.name = "hello TWaver!";
                box.add(link);
            }

            protected function output(event:MouseEvent):void
            {
                text.text=xmlSerializer.serialize();
            }
        ]]>
    </mx:Script>
</mx:Application>
```

```
protected function submit(event:MouseEvent):void
{
    box.clear();
    xmlSerializer.deserialize(text.text);
}

]]>
</mx:Script>
<mx:Panel title="XML Serializer Demo" width="100%" height="100%">
    <mx:HDividedBox width="100%" height="100%">
        <twaver:Network id="network" backgroundColor="0x00ff00" width="50%" height="100%">
            </twaver:Network>
            <mx:VBox width="50%" height="100%">
                <mx:TextArea width="100%" height="100%" id="text"/>
                <mx:HBox width="100%" horizontalAlign="right">
                    <mx:Button label="output" click="output(event)"/>
                    <mx:Button label="submit" click="submit(event)"/>
                </mx:HBox>
            </mx:VBox>
        </mx:HDividedBox>
    </mx:Panel>
</mx:Application>
```

The output interface:



Data Element

Data Element is the basic element of Data model. It is applied to describe graphic and business of network element, or It is just the data added in network. All Data Element of TWaver Flex implement interface `twaver.IData`. TWaver Flex pre-defined three kinds of data to meet with different requirements that `twaver.IElement` is used for describing network element, `twaver.IAlarm` is for describing alarm element and `twaver.ILayer` is for describing layers in topology. thereinto, `twaver.IElement` extends several kinds of network element to express abundant characteristics of network, including basic network element, links, bus, equipment and etc.

This chapter introduce the peculiarities, usage and extended application of these network elements and other data elements.

- [Basic Data Element](#)
- [Alarm Data Element](#)
- [Layer Data Element](#)
- [Topology Element](#)
 - [twaver.Node](#)
 - [twaver.Link](#)
 - [Links Binding](#)
 - [Links Type](#)
 - [twaver.Follower](#)
 - [twaver.ISubNetwork](#)
 - [twaver.Group](#)
 - [twaver.ShapeNode](#)
 - [twaver.Grid](#)

Basic Data Element

IData is the basic interface of TWaver Flex Data Element, and it takes Data as its implementation class. Data defined basic properties such as id, name, icon, toolTip, parent and children and etc, packed event dispatching methods, supports importing and exporting xml data which lay foundation of data serialization between each TWaver platform.

Data implements flash.events.EventDispatcher which gives its the event dispatching and listening function. It realizes event dispatching and listening by following methods:

```
flash.events.EventDispatcher#dispatchEvent(event:Event):Boolean
flash.events.EventDispatcher#addEventListener(type:String, listener:Function,
useCapture:Boolean=false, priority:int=0, useWeakReference:Boolean=false):void
```

Data serve twaver.IData as its interface, realizes the dispatching and listening of property chang.

```
function dispatchPropertyChangeEvent(property:String, oldValue:Object, newValue:Object):Boolean;
function addPropertyChangeListener(listener:Function, priority:int = 0, useWeakReference:Boolean = false):void;
function removePropertyChangeListener(listener:Function):void;
```

Data defined parent-child relations, make each element enable adding children and setting parent.

```
function get childrenCount():int;function get hasChildren():Boolean;
function isDescendantOf(data:IData):Boolean;
function isParentOf(data:IData):Boolean;
function isRelatedTo(data:IData):Boolean;
```

More, Data defined some basic properties including id, name, icon, toolTip and etc.

Thereinto, In Data container ID is the unique identification of data element. It can't be duplicated. TWaver Flex will set an unique identification for element when constructing Data, of course, user also can set value for ID, making sure the id can't be duplicated.

```
public function Data(id:Object=null)
public function get id():Object
public function get/set name():String
public function get/set icon():String
public function get/set toolTip():String
```

User can put up other properties for element by implementing setPropertyValue(...) method which is similar to HashMap in Java.

```
function setPropertyValue(property:String, value:Object):void;
function getPropertyValue(property:String):Object;
```

Data Element owns importing and exporting functions which give convinient to exchange and transform data.

```
public function serializeXML(serializer:XMLSerializer, newInstance:IData):void
```

```
public function deserializeXML(serializer:XMLSerializer, xml:XML):void
```

Alarm Data Element

Each alarm defined by TWaver Alarm own level which is used for reacting degree of urgency. AlarmBox maintain the alarm element, and associate alarm with the network element in topology. Network element doesn't store specific alarm but store current alarm states information.

Alarm

Alarm is the data model of equipment fault and network exception in webmaster system, connected with Element to express alarm information of network element. Alarm pre-defined alarm levels, alarm cleared or not, alarm acknowledged or not and ID of alarm network element, More, user can add other properties by implementing setPropertyValue(...) method.

The properties of Alarm:

```
public function Alarm(alarmID:Object = null, elementID:Object = null, alarmSeverity:AlarmSeverity = null,
isAked:Boolean = false, isCleared:Boolean = false)
public function get elementID():Object
public function get/set isAked():Boolean
public function get/set isCleared():Boolean
public function get/set alarmSeverity():AlarmSeverity
```

Alarm Levels

Alarm level is used for reacting degree of urgency. TWaver Flex pre-defined six alarm levels, the larger of default value, the more critical this alarm is.

```
public static var CRITICAL:AlarmSeverity = register(500, "Critical", "C", 0xFF0000);
public static var MAJOR:AlarmSeverity = register(400, "Major", "M", 0xFFA000);
public static var MINOR:AlarmSeverity = register(300, "Minor", "m", 0xFFFF00);
public static var WARNING:AlarmSeverity = register(200, "Warning", "W", 0x00FFFF);
public static var INDETERMINATE:AlarmSeverity = register(100, "Indeterminate", "N", 0xC800FF);
public static var CLEARED:AlarmSeverity = register(0, "Cleared", "R", 0x00FF00);
```

| 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 |

All levels of Alarm are static variables, user can register or unregister alarm levels in overall situation. in addition, TWaver Flex offer method to clear all alarm levels(Pay attention to use clearing all levels method, it will effect whole program).

```
public static function register(value:int, name:String, nickName:String,
color:uint, displayName:String = null):AlarmSeverity
public static function unregister(name:String):AlarmSeverity
public static function clear():void
```

There are methods to listen change of alarm levels in overall situation such as registering, unregistering and clearing alarm levels.

```
public static function addAlarmSeverityChangeListener(listener:Function, priority:int = 0,
useWeakReference:Boolean = false):void
public static function removeAlarmSeverityChangeListener(listener:Function):void
```

Alarm State

A project can appear thousands of alarm at one time in practical terms. Obviously network element connected with alarm directly is very heavy in the alarm storm, So TWaver Flex detached network element and alarm element. Network element save the alarm state only including how much alarm it has, what is the highest level and etc. The detail information of alarm is stored in AlarmBox. How to response alarm storm and how to use AlarmBox will be introduced in following chapter.

Alarm state is defined by twaver.AlarmState, is to express the level and quantity of new alarm issued.

Alarm state properties including highest level of acknowledged alarms, highest level of new alarms, highest level of all alarms, the highest but one of new alarms, highest level of self alarm, transmission alarm level and quantity of each alarm level.

```
public function get highestAcknowledgedAlarmSeverity():AlarmSeverity
public function get highestNewAlarmSeverity():AlarmSeverity
public function get hasLessSevereNewAlarms():Boolean
public function get highestOverallAlarmSeverity():AlarmSeverity
public function get highestNativeAlarmSeverity():AlarmSeverity
public function get/set propagateSeverity():AlarmSeverity
public function getAcknowledgedAlarmCount(severity:AlarmSeverity = null):int
public function getAlarmCount(severity:AlarmSeverity = null):int
public function getNewAlarmCount(severity:AlarmSeverity = null):int
public function setNewAlarmCount(severity:AlarmSeverity, count:uint):void
```

The methods to modify alarm state: acknowledge alarm, clear alarm, add/decrease acknowledged alarm, remove alarms and etc.

```
public function increaseAcknowledgedAlarm(severity:AlarmSeverity, increment:uint=1):void
public function increaseNewAlarm(severity:AlarmSeverity, increment:uint=1):void
public function decreaseAcknowledgedAlarm(severity:AlarmSeverity, decrement:uint=1):void
public function decreaseNewAlarm(severity:AlarmSeverity, decrement:uint=1):void
public function acknowledgeAlarm(severity:AlarmSeverity):void
public function acknowledgeAllAlarms(severity:AlarmSeverity = null):void
public function removeAllNewAlarms(severity:AlarmSeverity = null):void
public function setAcknowledgedAlarmCount(severity:AlarmSeverity, count:uint):void
public function removeAllAcknowledgedAlarms(severity:AlarmSeverity = null):void
public function clear():void
```

Other functions:

```
public function isEmpty():Boolean
public function get/set isEnabledPropagation():Boolean
```

The Usage of Alarm

User need to pay attention to alarm adding and removing, it should be operated via AlarmBox which is consistent with adding and removing Element via ElementBox.

For example:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
    applicationComplete="init()">
<mx:Script>
<![CDATA[
    import twaver.*;
    import twaver.network.Network;

    private function init():void{
        var box:ElementBox = network.elementBox;

        var node:Node=new Node();
        node.setLocation(50,50);
        box.add(node);

        addAlarm("alarm 1",node.id,AlarmSeverity.CRITICAL,box.alarmBox);
    }

    //Generally we add alarm via AlarmBox which is consistent with adding/removing Element via ElementBox
    private function addAlarm(alarmID:Object,elementID:Object,
        alarmSeverity:AlarmSeverity,alarmBox:AlarmBox):void{
        var alarm:Alarm=new Alarm(alarmID,elementID,alarmSeverity);
        alarmBox.add(alarm);
    }
    ]]>
</mx:Script>
<mx:Panel title="Hello TWaver! - Alarm" width="100%" height="100%">
    <twaver:Network id="network" width="100%" height="100%"/>
</mx:Panel>
</mx:Application>
```



Layer Data Element

ILayer is used for describing layer information of network element. Layer implements interface ILayer which own three special properties including visible, editable, movable.

```
public function get/set visible():Boolean
public function get/set movable():Boolean
public function get/set editable():Boolean
```

The hierarchy of TWaver Flex is maintained by LayerBox, the default order is decided by father-child relations and the join sequence. Each element connect with certain layer by setting layer ID, thus, all network elements in topology will be shown in hierarchy.

Let's explain how to use layer and its three properties: the up-down move of layer will be explained more in LayerBox.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
  applicationComplete="init()">
  <mx:Script>
    <![CDATA[
      import twaver.*;
      import twaver.network.Network;

      private var box:ElementBox;
      private var layerBox:LayerBox;
      private function init():void{
        box=network.elementBox;
        layerBox=box.layerBox;
        tree.dataBox=layerBox;

        network.setEditInteractionHandlers();

        var layer1:Layer=new Layer("unmovable","unmovable layer");
        layer1.movable=false;
        var layer2:Layer=new Layer("uneditable","uneditable layer");
        layer2.editable=false;
        var layer3:Layer=new Layer("invisible","invisible layer");
        layer3.visible=false;

        layerBox.add(layer1);
        layerBox.add(layer2);
        layerBox.add(layer3,0);

        createNode(layer1,Consts.SHAPE_CIRCLE,10,40,100,100,0xff0000);
        createNode(layer2,Consts.SHAPE_DIAMOND,30,60,100,100,0x00ff00);
        createNode(layer3,Consts.SHAPE_RECTANGLE,50,80,100,100,0x0000ff);
        createNode(layerBox.defaultLayer,Consts.SHAPE_RECTANGLE,70,20,150,150,0x808080);

      }

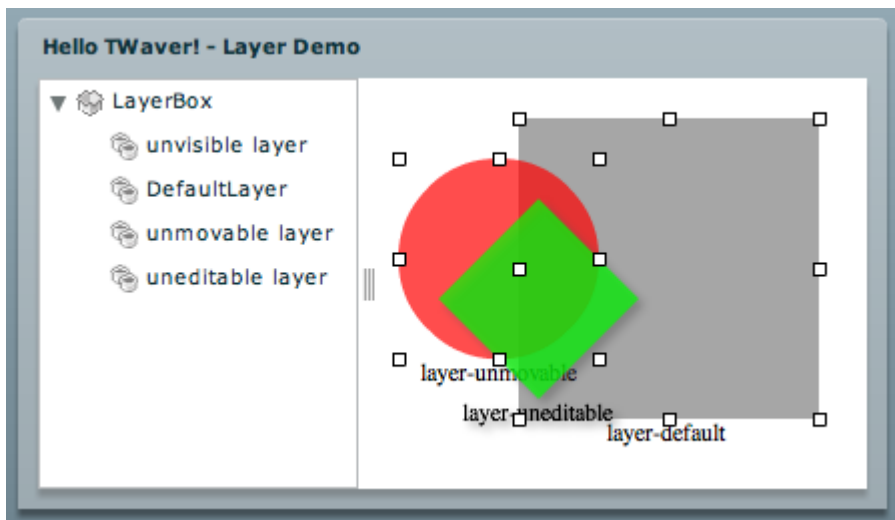
      private function createNode(layer:ILayer,shape:String,x:int,y:int,width:int,height:int,fillColor:uint):Node{
        var node:Node=new Node();
        node.layerID=layer.id;
        node.name="layer-" + layer.id;
        node.setStyle(Styles.CONTENT_TYPE, Consts.CONTENT_TYPE_VECTOR);
        node.setStyle(Styles.VECTOR_FILL_ALPHA,0.7);
        node.setStyle(Styles.VECTOR_SHAPE,shape);
        node.setSize(width,height);
```

```

        node.setLocation(x,y);
        node.setStyle(Styles.VECTOR_FILL_COLOR,fillColor);
        box.add(node);
        return node;
    }

    ]]>
</mx:Script>
<mx:Panel title="Hello TWaver! - Layer Demo" width="100%" height="100%">
    <mx:HDividedBox width="100%" height="100%">
        <twaver:Tree id="tree" width="30%" height="100%" />
        <twaver:Network id="network" width="70%" height="100%" />
    </mx:HDividedBox>
</mx:Panel>
</mx:Application>

```



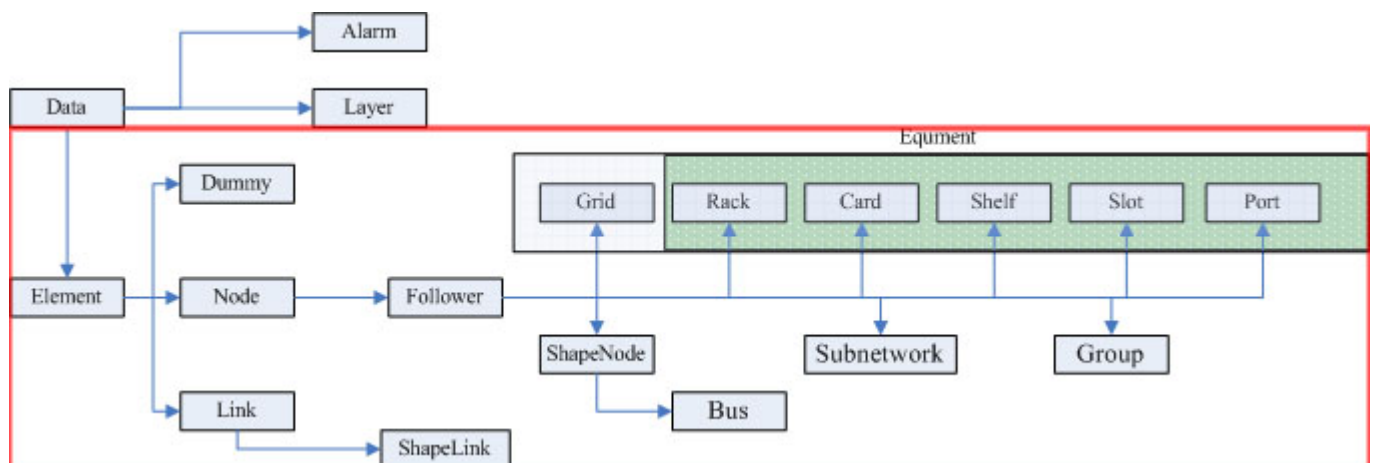
Topology Element

Interface IElement is to define the network element in topology, It is the most important data element in TWaver Flex. Topology element is for topology, It will be in Dummy, Node and Link.

Thereinto the dummy is not visible in topology, but visible in the tree component. Generally dummy will be parent node of other nodes to classify and be in category. For example, put all links network element under one dummy, that mean its the link catagory.

Node is the most commonly used network element. It stands for entity object including nodes, group, subnetwork, equipment and etc.

Link stands for connection relations between nodes. ShipLink extends Link, is used for representing irregular trend links.



IElement implements interface IData, extends properties such as alarmState, layerID, elementUIClass and etc to represent alarm state, layer ID, view type of network element correspondingly.

```
function get layerID():Object;
function set layerID(layerID:Object):void;
function get alarmState():AlarmState;
function get elementUIClass():Class;
```

Thereinto elementUIClass is the ElementUI class or extends ElementUI. Different network element correspond with certain ElementUI, for example, twaver.Node correspond with twaver.network.ui.NodeUI, twaver.Grid to twaver.network.ui.GridUI. ElementUI implements UIComponent. It is Flex component, is the view component of network element in topology, these two points construct a data view separation model. The detail will be introduced in view component chapter.

IElement implements interface IStyle, defined get/setStyle() methods to set network element style including color, border, background, alignment and etc. Generally different network element has different style properties, Please reference to stylesheet for detail.

```
function setStyle(styleProp:String, newValue:*) :IStyle;
function getStyle(styleProp:String, returnDefaultIfNull:Boolean = true):*;
```

Set label color to be red for network element:

```
var node:Node=new Node();
```



```
node.setStyle(Styles.LABEL_COLOR,0xff0000);
node.name="Node A";
```

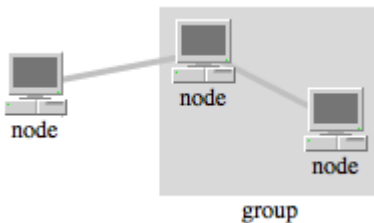
Run interface as follow:



Otherwise, Element defined isAdjustedToBottom():Boolean method to represent whether view in bottom, the default value is false.

```
function isAdjustedToBottom():Boolean;
```

twaver.Group overwrite this method, and the return value is true which mean element will be in bottom. Thus, it will not cover the child node but locate at the bottom of this node:



- [twaver.Node](#)
- [twaver.Link](#)
- [twaver.Follower](#)
- [twaver.ISubNetwork](#)
- [twaver.Group](#)
- [twaver.ShapeNode](#)
- [twaver.Grid](#)

twaver.Node

twaver.Node implements twaver.Element, stands for node in topology. Generally node represent entity object. It can be the endpoint of link, and its location and size can be fixed by x, y, width and height properties. It also has picture property.

```
[Embed(source="resource/images/portUpImage.png")]
public static const portUpImage:Class;
private function init():void{
    Utils.registerImageByClass("portUpImage",portUpImage);
    ...
}
```

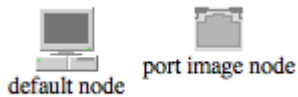
twaver.Node set picture:

```
public function get/setimage():String
```

For example:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
    applicationComplete="init()">
    <mx:Script>
        <![CDATA[
            import twaver.*;
            import twaver.network.Network;

            [Embed(source="resource/images/portUpImage.png")]
            public static const portUpImage:Class;
            private function init():void{
                Utils.registerImageByClass("portUpImage",portUpImage);
                var box:ElementBox = network.elementBox;
                var node:Node=new Node();
                node.name="default node";
                node.setLocation(20,20);
                box.add(node);
                var imageNode:Node=new Node();
                imageNode.image="portUpImage";
                imageNode.setLocation(100,20);
                imageNode.name="port image node";
                box.add(imageNode);
            }
        ]]>
    </mx:Script>
    <mx:Panel title="Hello TWaver!" width="100%" height="100%">
        <twaver:Network id="network" width="100%" height="100%"/>
    </mx:Panel>
</mx:Application>
```



Related properties and methods for the location and size of node

```
public function get x():Number
public function get y():Number
public function get location():Point
public function set location(location:Point):void
public function setLocation(x:Number, y:Number):void
public function setCenterLocation(x:Number, y:Number):void
public function get centerLocation():Point
public function set centerLocation(point:Point):void
public function translate(dx:Number, dy:Number):void
public function get width():Number
public function set width(width:Number):void
public function get height():Number
public function set height(height:Number):void
public function setSize(width:Number, height:Number):void
public function get size():Size
public function get rect():Rectangle
```

Get the connected links methods: if there is no links for this node, it will return null.

```
public function get loopedLinks():ICollection
public function get links():ICollection
public function get agentLinks():ICollection
public function get fromLinks():ICollection
public function get toLinks():ICollection
public function get hasAgentLinks():Boolean
public function get fromAgentLinks():ICollection
public function get toAgentLinks():ICollection
```

In addition, Node can add followers which can be moved with node together. This will be introduced in following chapters.

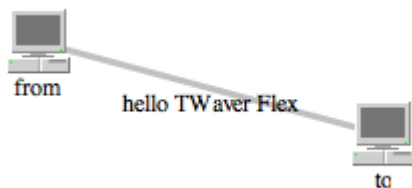
```
public function get followers():ICollection
```

twaver.Link

Link generally stands for connection between nodes with start point and end point. TWaver supports loopback that the start point and the end point will be the same one.

Start Node, End Node and Link

```
public function get/set fromNode():Node
public function get/set toNode():Node
```



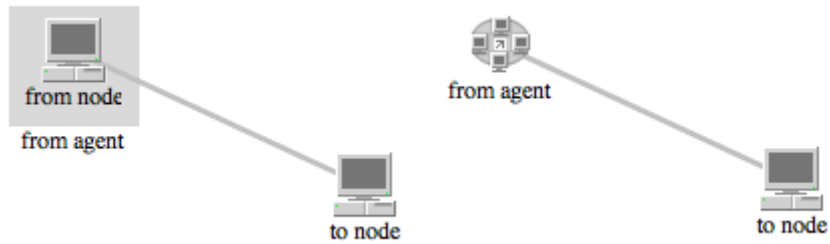
Start Node Agent and End Node Agent

The node connecting with link directly named fromNode, toNode in TWaver Flex. If the start or end node is in one Group, It will look like one group connect with one node when the group is in combination state, In this situation the group will be agent node.

```
public function get fromAgent():Node
public function get toAgent():Node
```

For example:

```
var node1:Node=new Node();
node1.setLocation(20,20);
node1.name="from node";
var node2:Node=new Node();
node2.setLocation(150,50);
node2.name="to node";
var group:Group=new Group();
group.addChild(node1);
group.name="from agent";
var link6:Link=new Link(node1,node2);
box.add(node1);
box.add(node2);
box.add(group);
box.add(link6);
```

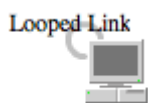


Self-Loop, The Start and End Point of This Link Share the Same Node

```
public function isLooped():Boolean
```

For example:

```
var node:Node=new Node();
box.add(node);
link=new Link(node,node);
link.name="Looped Link";
link.setStyle(Styles.LABEL_POSITION,Consts.POSITION_TOP);
box.add(link);
```



The Expend and Binding of Link

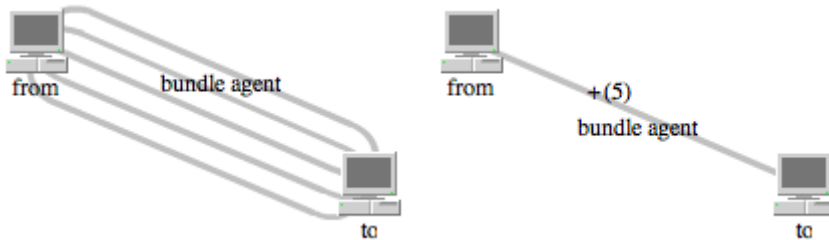
When there are multi links between nodes, TWaver supports expending and binding links. Double click one link can realize the states switchover by default. The visible link when links binding together is named bundle agent, the default agent will be the first link. User can set all these default setting.

```
public function get bundleLinks():BundleLinks
public function get bundleCount():int
public function get bundleIndex():int
public function reverseBundleExpanded():Boolean
public function isBundleAgent():Boolean
```

For example:

```
var from:Node = new Node();
from.name = "from";
from.location = new Point(20, 20);
box.add(from);
var to:Node = new Node();
to.name = "to";
to.location = new Point(150, 60);
box.add(to);
var link:Link = new Link(from,to);
link.name = "bundle agent";
box.add(link);
```

```
var link2:Link=new Link(from,to);
box.add(link2);
var link3:Link=new Link(from,to);
box.add(link3);
var link4:Link=new Link(from,to);
box.add(link4);
var link5:Link=new Link(from,to);
box.add(link5);
```



- [Links Binding](#)
- [Links Type](#)

Links Binding

The chapter of twaver.Link introduce links binding and expending in brief. We will introduce functions such as the gap between extended links, binging in group and self-loop binding and etc.

Normal Links Binding(not self-loop)

TWaver defined six binding properties:

Styles.LINK_BUNDLE_ID : The identification of bingding, links will be in same group with same ID
 Styles.LINK_BUNDLE_INDEPENDENT : Whether the bundle is independent, whether bundle independently when has multi link groups
 Styles.LINK_BUNDLE_GAP : The gap between links
 Styles.LINK_BUNDLE_OFFSET : The link offset from endpoint
 Styles.LINK_BUNDLE_ENABLE : Whether join the binding or not
 Styles.LINK_BUNDLE_EXPANDED : Whether extended links, **false** stands **for** binding

For example: set two boudle links with ID "1" and "2"

```

<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
  applicationComplete="init()">
  <mx:Script>
    <![CDATA[
      import twaver.*;
      import twaver.network.Network;

      private var box:ElementBox;
      private function init():void{
        box = network.elementBox;

        var from:Node = new Node();
        from.name = "from";
        from.location = new Point(20, 20);
        box.add(from);
        var to:Node = new Node();
        to.name = "to";
        to.location = new Point(150, 60);
        box.add(to);

        createLink(from,to,"g1_1",1);
        createLink(from,to,"g1_2",1);
        createLink(from,to,"g1_3",1);

        createLink(from,to,"g2_1",2,0xff0000);
        createLink(from,to,"g2_2",2,0xff0000);
        createLink(from,to,"g2_3",2,0xff0000);
      }

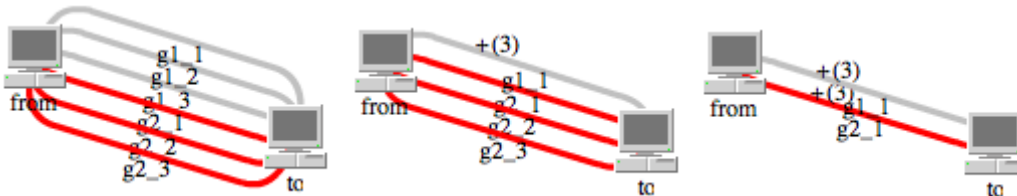
      private function createLink(from:Node,to:Node,name:String,groupID:int=-1,
        color:int=-1,type:String=null,groupIndependent:Boolean=false,
        gap:Number=-1,offset:Number=-1,bundleEnable:Boolean=true):Link{
        var link:Link=new Link(from,to);
        link.name=name;
        if(type){
          link.setStyle(Styles.LINK_TYPE,type);
        }
        if(color>=0){
  
```

```

        link.setStyle(Styles.LINK_COLOR,color);
    }
    if(groupID>=0){
        link.setStyle(Styles.LINK_BUNDLE_ID,groupID);
    }
    if(gap>0){
        link.setStyle(Styles.LINK_BUNDLE_GAP,gap);
    }
    if(offset>0){
        link.setStyle(Styles.LINK_BUNDLE_OFFSET,offset);
    }
    link.setStyle(Styles.LINK_BUNDLE_INDEPENDENT,groupIndependent);
    link.setStyle(Styles.LINK_BUNDLE_ENABLE,bundleEnable);
    box.add(link);
    return link;
}
]]>
</mx:Script>
<mx:Panel title="Hello TWaver! - Link Bundler Demo" width="100%" height="100%">
    <twaver:Network id="network" width="100%" height="100%" />
</mx:Panel>
</mx:Application>

```

Run application as follows:



Let's set one bundle binding independently, and set the link type to be `Consts.LINK_TYPE_EXTEND_BOTTOM`

```

private function init():void{
    box = network.elementBox;

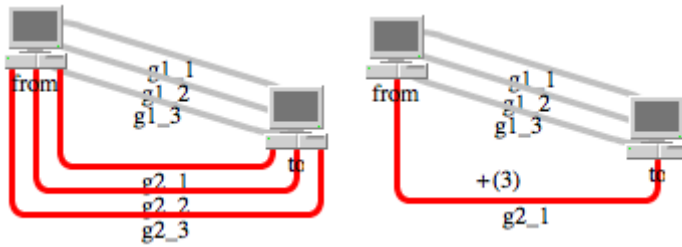
    var from:Node = new Node();
    from.name = "from";
    from.location = new Point(20, 20);
    box.add(from);
    var to:Node = new Node();
    to.name = "to";
    to.location = new Point(150, 60);
    box.add(to);

    createLink(from,to,"g1_1",1);
    createLink(from,to,"g1_2",1);
    createLink(from,to,"g1_3",1);

    createLink(from,to,"g2_1",2,0xff0000,Consts.LINK_TYPE_EXTEND_BOTTOM,true);
    createLink(from,to,"g2_2",2,0xff0000,Consts.LINK_TYPE_EXTEND_BOTTOM,true);
    createLink(from,to,"g2_3",2,0xff0000,Consts.LINK_TYPE_EXTEND_BOTTOM,true);
}

```

Run as follows:



Let's have a try for parameters LINK_BUNDLE_GAP and LINK_BUNDLE_OFFSET. In addition, let's add one link without binding by setting LINK_BUNDLE_ENABLE parameter.

```
private function init():void{
    box = network.elementBox;

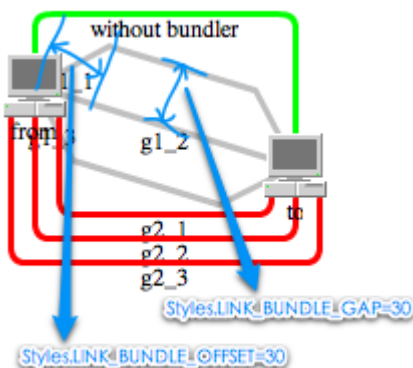
    var from:Node = new Node();
    from.name = "from";
    from.location = new Point(20, 20);
    box.add(from);
    var to:Node = new Node();
    to.name = "to";
    to.location = new Point(150, 60);
    box.add(to);

    var gap:Number=30;
    var offset:Number=30;
    createLink(from,to,"g1_1",1,0,Consts.LINK_TYPE_TRIANGLE,false,gap,offset);
    createLink(from,to,"g1_2",1,0,Consts.LINK_TYPE_TRIANGLE,false,gap,offset);
    createLink(from,to,"g1_3",1,0,Consts.LINK_TYPE_TRIANGLE,false,gap,offset);

    createLink(from,to,"g2_1",2,0xff0000,Consts.LINK_TYPE_EXTEND_BOTTOM,true);
    createLink(from,to,"g2_2",2,0xff0000,Consts.LINK_TYPE_EXTEND_BOTTOM,true);
    createLink(from,to,"g2_3",2,0xff0000,Consts.LINK_TYPE_EXTEND_BOTTOM,true);

    createLink(from,to,"without bundler",-1,0x00ff00,Consts.LINK_TYPE_EXTEND_TOP,false,0,0,false);
}
```

Run as follows:



The example as below show us how to set binding state for links with API

```
private function init():void{
    box = network.elementBox;

    var from:Node = new Node();
    from.name = "from";
    from.location = new Point(20, 20);
}
```

```

box.add(from);
var to:Node = new Node();
to.name = "to";
to.location = new Point(150, 60);
box.add(to);

var gap:Number=30;
var offset:Number=30;
createLink(from,to,"g1_1",1,0,Consts.LINK_TYPE_TRIANGLE,false,gap,offset);
createLink(from,to,"g1_2",1,0,Consts.LINK_TYPE_TRIANGLE,false,gap,offset);
createLink(from,to,"g1_3",1,0,Consts.LINK_TYPE_TRIANGLE,false,gap,offset);

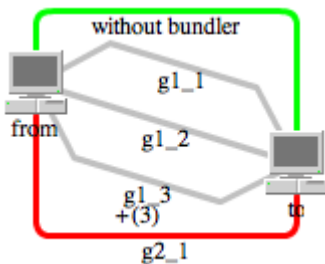
var link1:Link=createLink(from,to,"g2_1",2,0xff0000,Consts.LINK_TYPE_EXTEND_BOTTOM,true);
var link2:Link=createLink(from,to,"g2_2",2,0xff0000,Consts.LINK_TYPE_EXTEND_BOTTOM,true);
var link3:Link=createLink(from,to,"g2_3",2,0xff0000,Consts.LINK_TYPE_EXTEND_BOTTOM,true);

createLink(from,to,"without bundler",-1,0x00ff00,Consts.LINK_TYPE_EXTEND_TOP,false,0,0,false);

network.callLater(function():void{
    link1.setStyle(Styles.LINK_BUNDLE_EXPANDED,false);
    link2.setStyle(Styles.LINK_BUNDLE_EXPANDED,false);
    link3.setStyle(Styles.LINK_BUNDLE_EXPANDED,false);
});
}

```

Run as follow:



Self-Loop Binding

TWaver Flex supports self-loop link to express the native connection relations of network element. It controls by a set of parameters on its own.

Styles.LINK_LOOPED_GAP : the gap of self-loop, 12 is the **default** value
 Styles.LINK_LOOPED_DIRECTION : the direction of links including east,south,west,north and etc eight values.
 twaver.Consts.DIRECTION_*
 Styles.LINK_LOOPED_TYPE : the type of self-loop including circle and rectangle, twaver.Consts.LINK_LOOPED_TYPE_*

The example as below creat three self-loop links, the gap is 20, and the type is twaver.Consts.LINK_LOOPED_TYPE_RECTANGLE

```

private function init():void{
    box = network.elementBox;

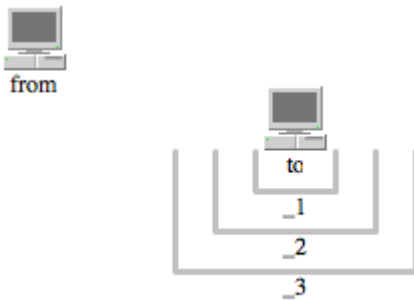
    var from:Node = new Node();
    from.name = "from";
    from.location = new Point(20, 20);
    box.add(from);
    var to:Node = new Node();
    to.name = "to";

```

```
to.location = new Point(150, 60);
box.add(to);

var i:int=3;
while(i>0){
    var link:Link=createLink(to,to,"_"+i);
    link.setStyle(Styles.LINK_LOOPED_GAP,20);
    link.setStyle(Styles.LINK_LOOPED_DIRECTION,Consts.DIRECTION_SOUTH);
    link.setStyle(Styles.LINK_LOOPED_TYPE,Consts.LINK_LOOPED_TYPE_RECTANGLE);
    i--;
}
}
```

Run as follows:



Links Type

The links direction of TWaver Flex has two ways:

One of is twaver.Link which decided the links direction by seting parameters, another is twaver.ShapeLink which decided by a series of control points.

Let's introduce the link types what TWaver has pre-defined:

```
public static const LINK_TYPE_ARC:String = "arc";
public static const LINK_TYPE_TRIANGLE:String = "triangle";
public static const LINK_TYPE_PARALLEL:String = "parallel";

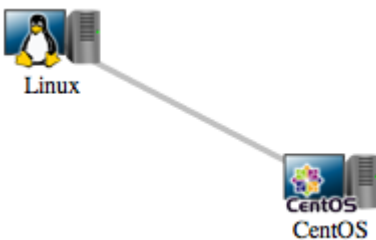
public static const LINK_TYPE_FLEXIONAL:String="flexional";
public static const LINK_TYPE_FLEXIONAL_HORIZONTAL:String="flexional.horizontal";
public static const LINK_TYPE_FLEXIONAL_VERTICAL:String="flexional.vertical";

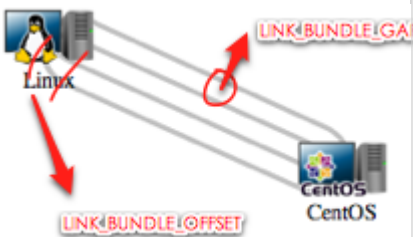
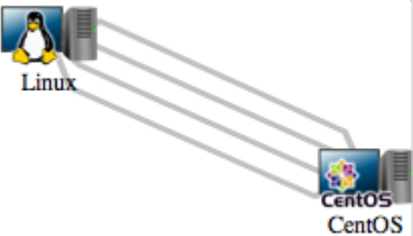
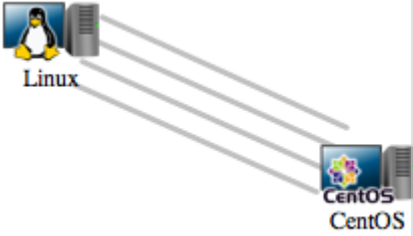

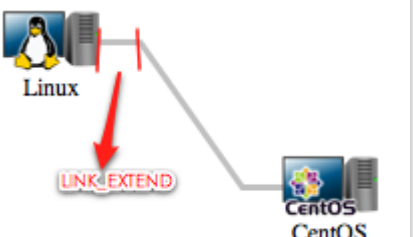

public static const LINK_TYPE_ORTHOGONAL:String="orthogonal";
public static const LINK_TYPE_ORTHOGONAL_HORIZONTAL:String = "orthogonal.horizontal";
public static const LINK_TYPE_ORTHOGONAL_VERTICAL:String = "orthogonal.vertical";
public static const LINK_TYPE_HORIZONTAL_VERTICAL:String = "orthogonal.H.V";
public static const LINK_TYPE_VERTICAL_HORIZONTAL:String = "orthogonal.V.H";
public static const LINK_TYPE_EXTEND_TOP:String = "extend.top";
public static const LINK_TYPE_EXTEND_LEFT:String = "extend.left";
public static const LINK_TYPE_EXTEND_BOTTOM:String = "extend.bottom";
public static const LINK_TYPE_EXTEND_RIGHT:String = "extend.right";
```

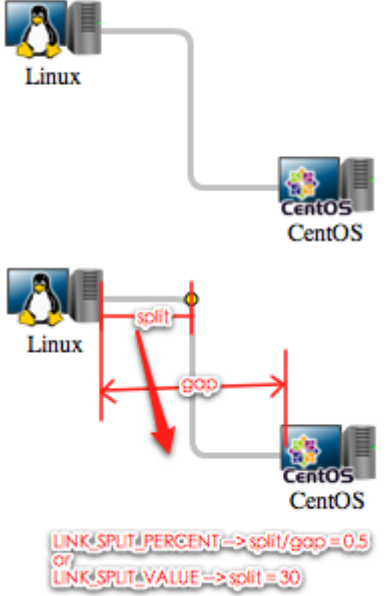
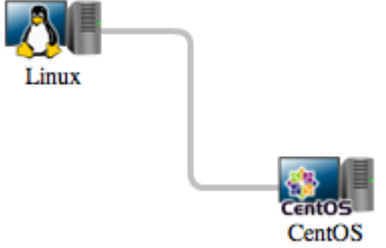


Set link types

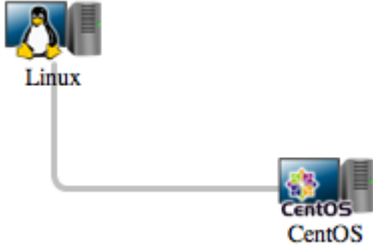
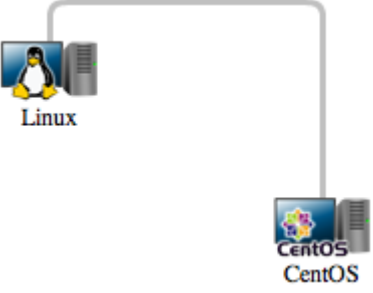
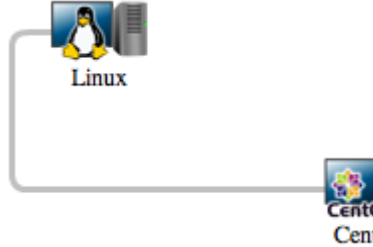
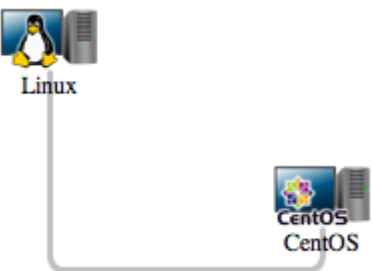

```
link.setStyle(Styles.LINK_TYPE,Consts.LINK_TYPE_ORTHOGONAL);
```

The table below list all kinds of link types and corresponding control parameters

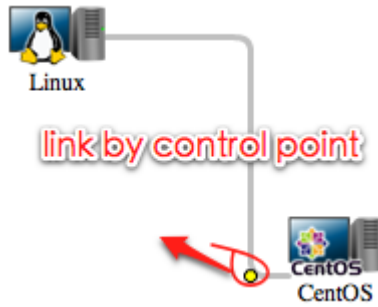
| Link Type | View | Control Parameters |
|--|---|---|
| <p>Basic Type is to connect start and end node directly.</p> <p>When there are more than one links between two nodes, TWaver classify three types by links expanding effects:</p> <p>LINK_TYPE_ARC</p> <p>LINK_TYPE_TRIANGLE</p> <p>LINK_TYPE_PARALLEL</p> <p>These types only works when there are multi links between two nodes.</p> |  | <p>LINK_FROM_POSITION Default Consts.POSITION_CENTER</p> <p>LINK_FROM_XOFFSET Default 0</p> <p>LINK_FROM_YOFFSET Default 0</p> <p>LINK_TO_POSITION Default Consts.POSITION_CENTER</p> <p>LINK_TO_XOFFSET Default 0</p> <p>LINK_TO_YOFFSET Default 0</p> |

| | | |
|--|--|---|
| LINK_TYPE_ARC The knee point present as arc when links extend. |  | LINK_BUNDLE_OFFSET The knee point offset, default value is 20 LINK_BUNDLE_GAP The gap between links, default value is 12 |
| LINK_TYPE_TRIANGLE The knee point at right angle when links extend. |  | The same as above |
| LINK_TYPE_PARALLEL Links present as parallel lines when links extend. |  | The same as above |
| LINK_TYPE_FLEXIONAL This kind of link has three directions: Automatic direction: Value as horizontal direction when gap is bigger in X-axis. Horizontal direction: LINK_TYPE_FLEXIONAL_HORIZONTAL Vertical direction: LINK_TYPE_FLEXIONAL_VERTICAL |  | LINK_FROM_AT_EDGE Links to edge of start point Default value is true LINK_TO_AT_EDGE Links to edge of end point Default value is true LINK_EXTEND Default value of extended gap is 20 |
| LINK_TYPE_FLEXIONAL_HORIZONTAL |  | The same as above |
| LINK_TYPE_FLEXIONAL_VERTICAL |  | The same as above |

| Orthogonal Line Links Type | | |
|--|---|---|
| <p>LINK_TYPE_ORTHOGONAL Orthogonal Line This kind of link has three directions: Automatic direction: Value as horizontal direction when gap is bigger in X-axis. Horizontal direction: LINK_TYPE_ORTHOGONAL_HORIZONTAL Vertical direction: LINK_TYPE_ORTHOGONAL_VERTICAL</p> |  | <p>LINK_FROM_AT_EDGE Links to edge of start point Default value is true LINK_TO_AT_EDGE Links to edge of end point Default value is true LINK_SPLIT_BY_PERCENT Whether split by percent Default value is true LINK_SPLIT_PERCENT The percent gap of split to start point Default value is 0.5 Set this property if split by percent LINK_SPLIT_VALUE The split offset to start point Default value is 20 Set this property if split by offset</p> |
| LINK_TYPE_ORTHOGONAL_HORIZONTAL |  | The same as above |
| LINK_TYPE_ORTHOGONAL_VERTICAL |  | The same as above |
| LINK_TYPE_HORIZONTAL_VERTICAL |  | <p>LINK_FROM_AT_EDGE Links to edge of start point Default value is true LINK_TO_AT_EDGE Links to edge of end point Default value is true</p> |

| | | |
|--|---|---|
| <p>LINK_TYPE_VERTICAL_HORIZONTAL From start point, align vertical directionat first then horizontal</p> |  | <p>The same as above</p> |
| <p>LINK_TYPE_EXTEND_TOP Extend upwards The extended value is split point to topmost</p> |  | <p>LINK_EXTEND The extended value, Default value is 20</p> |
| <p>LINK_TYPE_EXTEND_LEFT Extend left The extended value is split point to left-most</p> |  | <p>The same as above</p> |
| <p>LINK_TYPE_EXTEND_BOTTOM Extend downwards The extended value is split point to bottom-most</p> |  | <p>The same as above</p> |
| <p>LINK_TYPE_EXTEND_RIGHT Extend right The extended value is split point to right-most</p> |  | <p>The same as above</p> |

All orthogonal lines above are control by control points and decide their direction.
If these control points are set value at first, TWaver will graphic the split point by these control point priority.



LINK_CONTROL_POINT
Control point
This control point decided the split point of link
The direction of links is decided by link type
Automaticdirection:
LINK_TYPE_ORTHOGONAL
Horizontal direction:
LINK_TYPE_ORTHOGONAL_HORIZONTAL
Vertical direction:
LINK_TYPE_ORTHOGONAL_VERTICAL

twaver.Follower

The follower attach itself to host node. Generally it is used for equipment panel such as port rely on card.

Set Host Node

One node can be attached more than one followers

```
//get host node
public function get host():Node
//Whether attach itself to certain node
public function isHostOn(node:Node):Boolean
```

Looped Host

Moreover, followers can be attached to each other which means they can be host and follower for each other.

```
//Whether looped host on each other
public function isLoopedHostOn(follower:Follower):Boolean
```

For example:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
  applicationComplete="init()">
  <mx:Script>
    <![CDATA[
      import twaver.*;
      import twaver.network.Network;

      private function init():void{
        var box:ElementBox = network.elementBox;
        var node:Node=new Node();
        node.setLocation(20,20);
        node.name="Host";
        box.add(node);

        var follower:Follower=new Follower();
        follower.setLocation(50,50);
        follower.host=node;
        follower.name="Follower";
        box.add(follower);
      }
    ]]>
  </mx:Script>
  <mx:Panel title="Hello TWaver! - Follower" width="100%" height="100%">
    <twaver:Network id="network" width="100%" height="100%"/>
  </mx:Panel>
</mx:Application>
```



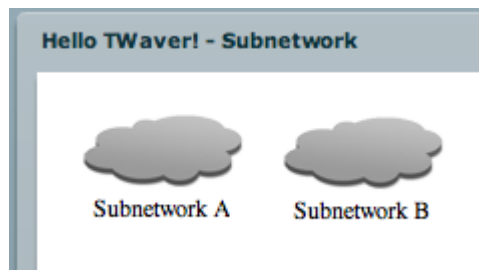
twaver.ISubNetwork

twaver.ISubNetwork is the interface of SubNetwork, stands for part of Net. TWaver Flex can switch SubNetwork in topology, and the components of interface are the network elements of current SubNetwork.

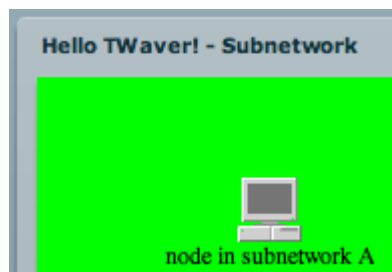
twaver.SubNetwork is the implement class of twaver.ISubNetwork. In topology double click can enter into SubNetwork, double click background can go back upper SubNetwork. The Null value of current SubNetwork stands for it is the highest SubNetwork.

Switch SubNetwork in Network:
twaver.network.Network

```
//Set current SubNetwork
public function get/set currentSubNetwork():ISubNetwork
//Enter into SubNetwork in flash
public function setCurrentSubNetwork(currentSubNetwork:ISubNetwork, animate:Boolean = false):void
//Go back to upper SubNetwork
public function upSubNetwork(animate:Boolean = false):void
```



Double click to go into SubNetwork:



For example:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
  applicationComplete="init()">
  <mx:Script>
    <![CDATA[
      import twaver.*;
      import twaver.network.Network;

      private function init():void{
        var box:ElementBox = network.elementBox;

        var node:Node=new Node();
        node.name="node in subnetwork A";
        node.setLocation(100,50);
```

```

var subNetworkA:SubNetwork=new SubNetwork();
subNetworkA.setStyle(Styles.BACKGROUND_TYPE,Consts.BACKGROUND_TYPE_VECTOR);
subNetworkA.setStyle(Styles.BACKGROUND_VECTOR_FILL_COLOR,0x00ff00);
subNetworkA.addChild(node);
subNetworkA.name="Subnetwork A";
subNetworkA.setLocation(20,20);

var node2:Node=new Node();
node2.name="node in subnetwork B";
node2.setLocation(100,50);
var subNetworkB:SubNetwork=new SubNetwork();
subNetworkB.setStyle(Styles.BACKGROUND_TYPE,Consts.BACKGROUND_TYPE_VECTOR);
subNetworkB.setStyle(Styles.BACKGROUND_VECTOR_FILL_COLOR,0x0000ff);
subNetworkB.addChild(node2);
subNetworkB.name="Subnetwork B";
subNetworkB.setLocation(120,20);

box.add(node);
box.add(subNetworkA);
box.add(node2);
box.add(subNetworkB);
}
]]>
</mx:Script>
<mx:Panel title="Hello TWaver! - Subnetwork" width="100%" height="100%">
  <twaver:Network id="network" width="100%" height="100%"/>
</mx:Panel>
</mx:Application>

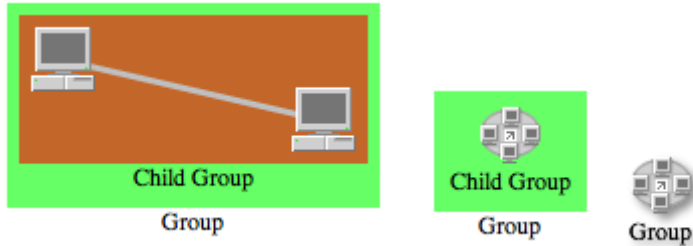
```

twaver.Group

Generally Group contains multi child-network element, display group and its child-network when extends group, show icon of group network element when combine group.

The effect drawing of group extending and combining show as below:

Double click can extend or combine group by default.



For example:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
    applicationComplete="init()">
<mx:Script>
<![CDATA[
    import twaver.*;
    import twaver.network.Network;

    private function init():void{
        var box:ElementBox = network.elementBox;
        var node:Node=new Node();
        node.setLocation(20,20);
        var node2:Node=new Node();
        node2.setLocation(150,50);
        var link:Link=new Link(node,node2);

        var childGroup:Group=new Group();
        childGroup.addChild(node);
        childGroup.addChild(node2);
        childGroup.addChild(link);
        childGroup.setStyle(Styles.GROUP_FILL_COLOR,0xff0000);
        childGroup.name="Child Group";
        childGroup.expanded=true;

        var group:Group=new Group();
        group.addChild(childGroup);
        group.setStyle(Styles.GROUP_FILL_COLOR,0x00ff00);
        group.name="Group";
        group.expanded=true;

        box.add(node);
        box.add(node2);
        box.add(link);
        box.add(childGroup);
        box.add(group);
    }
    ]]>
</mx:Script>
<mx:Panel title="Hello TWaver! - Follower" width="100%" height="100%">
    <twaver:Network id="network" width="100%" height="100%"/>
</mx:Panel>
</mx:Application>
```

```
</mx:Panel>  
</mx:Application>
```

twaver.ShapeNode

The location and figure of twaver.ShapeNode are decided by a series of control points. It is commonly used for representing irregular graphics such as coverage area, map block and etc.

Related methods to add/remove control point:

```
public function get/set points():ICollection
public function addPoint(point:Point):void
public function addPointAt(point:Point, index:int):void
public function setPointAt(point:Point, index:int):void
public function removePoint(point:Point):void
```

In addition, TWaver can offer methods to draw segments which will express ShapNode more enrichment such as curve (QuadTo), interruption (moveTo) and etc.

TWaver has three drawing methods: moveTo, lineTo, QuadTo
twaver.Consts

```
public static const SEGMENT_MOVETO:String = "moveTo";
public static const SEGMENT_LINETO:String = "lineTo";
public static const SEGMENT_QUADTO:String = "QuadTo";
```

Set segments
twaver.ShapeNode

```
public function get/set segments():ICollection
```

Let's give an example: Create five control points at first, then set drawing method for each segment. Move the first control to draw curve, then draw until the end.

```
var shapeNode2:ShapeNode=new ShapeNode();
shapeNode2.addPoint(new Point(30,10));
shapeNode2.addPoint(new Point(80,10));
shapeNode2.addPoint(new Point(100,90));
shapeNode2.addPoint(new Point(10,90));
shapeNode2.addPoint(new Point(30,10));
shapeNode2.setStyle(Styles.VECTOR_FILL_COLOR,0x00ff00);
shapeNode2.setLocation(150,10);

var segments:ICollection=new Collection();
segments.addItem(Consts.SEGMENT_MOVETO);
segments.addItem(Consts.SEGMENT_QUADTO);
segments.addItem(Consts.SEGMENT_QUADTO);
shapeNode2.segments=segments;
shapeNode2.name="ShapeNode with Segments";
box.add(shapeNode2);
```



ShapeNode



ShapeNode with Segments



Note: One curve segment needs two control points

The whole program of this application:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
    applicationComplete="init()">
  <mx:Script>
    <![CDATA[
      import twaver.*;
      import twaver.network.Network;

      private function init():void{
        var box:ElementBox = network.elementBox;

        var shapeNode:ShapeNode=new ShapeNode();
        shapeNode.addPoint(new Point(30,10));
        shapeNode.addPoint(new Point(80,10));
        shapeNode.addPoint(new Point(100,90));
        shapeNode.addPoint(new Point(10,90));
        shapeNode.addPoint(new Point(30,10));
        shapeNode.setStyle(Styles.VECTOR_FILL_COLOR,0xff0000);
        shapeNode.name="ShapeNode";
        box.add(shapeNode);

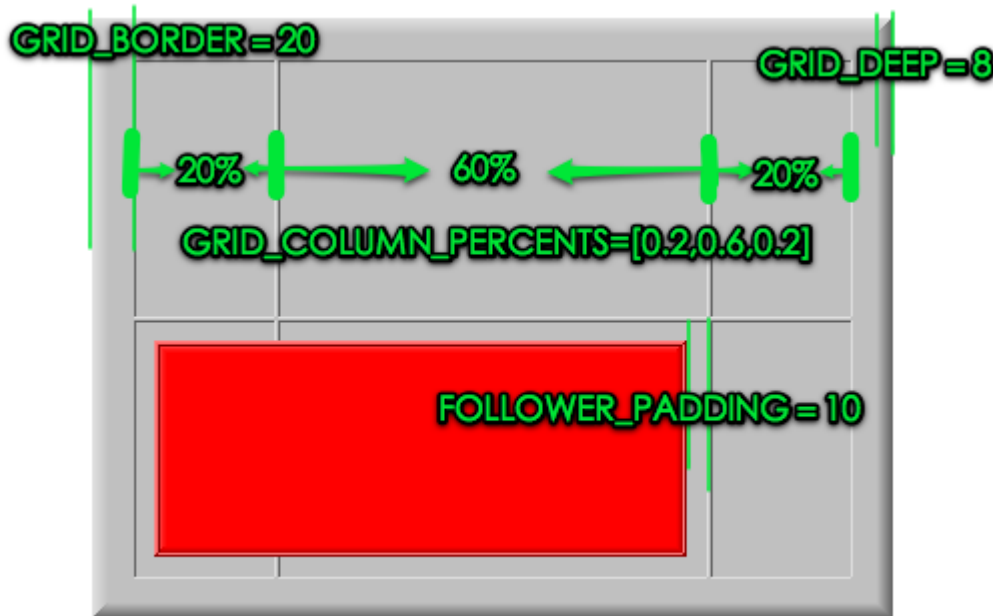
        var shapeNode2:ShapeNode=new ShapeNode();
        shapeNode2.addPoint(new Point(30,10));
        shapeNode2.addPoint(new Point(80,10));
        shapeNode2.addPoint(new Point(100,90));
        shapeNode2.addPoint(new Point(10,90));
        shapeNode2.addPoint(new Point(30,10));
        shapeNode2.setStyle(Styles.VECTOR_FILL_COLOR,0x00ff00);
        shapeNode2.setLocation(150,10);

        var segments:ICollection=new Collection();
        segments.addItem(Consts.SEGMENT_MOVETO);
        segments.addItem(Consts.SEGMENT_QUADTO);
        segments.addItem(Consts.SEGMENT_QUADTO);
        shapeNode2.segments=segments;
        shapeNode2.name="ShapeNode with Segments";
        box.add(shapeNode2);
      }
    ]]>
  </mx:Script>
  <mx:Panel title="Hello TWaver! - ShapeNode" width="100%" height="100%">
    <twaver:Network id="network" width="100%" height="100%"/>
  </mx:Panel>
</mx:Application>
```


twaver.Grid

twaver.Grid network element present as grid in topology, implements layout similar with TableLayout of java Swing. Thus, it realize positional placement by appointing column and row, the ranks of span.

The following example is a grid with three columns and two rows, and a child-network element in red color located at second row, first and two column.



For example:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
  applicationComplete="init()">
  <mx:Script>
    <![CDATA[
      import twaver.*;
      import twaver.network.Network;

      private function init():void{
        var box:ElementBox = network.elementBox;

        var grid:Grid=new Grid();
        grid.setLocation(20, 20);
        grid.setSize(400, 300);
        grid.setStyle(Styles.GRID_BORDER, 20);
        grid.setStyle(Styles.GRID_ROW_COUNT, 2);
        grid.setStyle(Styles.GRID_COLUMN_COUNT, 3);
        grid.setStyle(Styles.GRID_COLUMN_PERCENTS,[0.2,0.6,0.2]);
        grid.setStyle(Styles.GRID_DEEP, 8);
        box.add(grid);

        var cell:Grid=new Grid();
        cell.setStyle(Styles.FOLLOWER_COLUMN_INDEX,0);
        cell.setStyle(Styles.FOLLOWER_ROW_INDEX,1);
        cell.setStyle(Styles.FOLLOWER_COLUMN_SPAN,2);
        cell.setStyle(Styles.GRID_FILL_COLOR,0xff0000);
        cell.setStyle(Styles.FOLLOWER_PADDING,10);
        cell.setStyle(Styles.GRID_DEEP,5);
      }
    ]]>
  </mx:Script>
</mx:Application>
```

```
        cell.host=grid;
        grid.addChild(cell);
        box.add(cell);
    }
}]>
</mx:Script>
<mx:Panel title="Hello TWaver! - Grid" width="100%" height="100%">
    <twaver:Network id="network" width="100%" height="100%"/>
</mx:Panel>
</mx:Application>
```

Data Container

Based on introduction of data container classify and basic application in [Getting Started](#) chapter, we will introduce the application of data container in detail. Here includes various events of data container, quick-finder mechanism and selection model appliance.

- [Events of Data Container](#)
- [Quick Finder Mechanism](#)
- [Selection Model Appliance](#)

Events of Data Container

Data container is not only the collection of network elements, but also is the container of layers and the manager of selection model. Different changes of elements dispatch different events. For example, element plus or minus dispatch `DataBoxChangeEvent`, property changes of element dispatch `PropertyChangeEvent`, layer changes of element dispatch `HierarchyChangeEvent`, and selection changes will be maintained by `SelectionModel`. Know more about [SelectionModel](#).

Property Changes of Data Container -PropertyChange

The property of `DataBox` self can be changed such as name, client property. Event style property can be changed for `ElementBox`.

```
databox.name = "my databox";
databox.setClient(propertyName, value);
elementBox.setStyle(styleName, value);
...
```

The changes of data container property will dispatch `PropertyChangeEvent`. Add listener as following codes:

```
//databox property change listener
databox.addPropertyChangeListener(function(e:PropertyChangeEvent):void{ });
```

Element Plus or Minus - DataBoxChange

As the name suggests, the quantity of elements in container changes will dispatch `DataBoxChangeEvent`. Adding new elements is correspond to `DataBoxChangeEvent.ADD`. Removing elements is correspond to `DataBoxChangeEvent.REMOVE`. And clearing elements is correspond to `DataBoxChangeEvent.CLEAR`.

```
//databox change listener
databox.addDataBoxChangeListener(function(e:DataBoxChangeEvent):void{ });
```

`DataBoxChangeEvent` properties:

`kind:String` - three event types

- `DataBoxChangeEvent.ADD`
- `DataBoxChangeEvent.REMOVE`
- `DataBoxChangeEvent.CLEAR`

`data:IData` - target element. For example: `var dataA:Data = new Data()`, adding `dataA` into `dataBox`, the `dataA` property of correspondent event is `data`.

`datas:Array` - target element collection. For example: removing elements, the elements property of correspondent event is `datas`.

```
databox.addDataBoxChangeListener(function(e:DataBoxChangeEvent):void{
    var kind:String = e.kind;
    if(kind == DataBoxChangeEvent.ADD){
        var data:IData = e.data;
        trace("add a new data [" + data + "]");
    }
});
```

Property Changes of Element- DataPropertyChange

TWaver forward element changes by data container. In this way, we don't need to add listener on each element but only on data container.



The difference between DataBoxChange and DataPropertyChange is the target object. The DataBoxChange is point to databox self properties, and the DataPropertyChange is point to element properties.

```
//data property change listener
databox.addDataPropertyChangeListener(function(e:PropertyChangeEvent):void{ });
For example: add listener for name changes of all elements as following codes.
databox.addDataPropertyChangeListener(function(e:PropertyChangeEvent):void{
    if(e.property == "name"){
        trace("data name changed - old name: " + e.oldValue + ", new name: " + e.newValue);
    }
});
```

Layer Changes of Element - HiberarchyChange

Data container organizes the hierarchy by parent-child relationships of elements. Such as the tree view responds to the hierarchy of databox. You can change the hierarchy relationships by modifying parent-child relationships, or calling related functions. Data container trigger HiberarchyChange event when hierarchy is modified.

```
//hierarchy change listener
databox.addHierarchyChangeListener(function(e:HiberarchyChangeEvent):void{ });
```

The two important messages of HiberarchyChangeEvent are oldIndex and newIndex. The oldindex stands for original position and the newIndex is for new one.

Sample:

The following codes demonstrate all events and results.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
    xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute" creationComplete="init();">
<mx:Script>
<![CDATA[
    import mx.events.PropertyChangeEvent;

    import twaver.Data;
    import twaver.DataBox;
    import twaver.DataBoxChangeEvent;
    import twaver.HiberarchyChangeEvent;

    private var databox:DataBox;

    private function init():void{
        databox = new DataBox();

        //databox property change listener
        databox.addPropertyChangeListener(function(e:PropertyChangeEvent):void{
            log(e);
        });
```

```

//databox change listener
databox.addDataBoxChangeListener(function(e:DataBoxChangeEvent):void{
    log(e);
});

//data property change listener
databox.addDataPropertyChangeListener(function(e:PropertyChangeEvent):void{
    if(e.property == "name"){
        trace("data name changed - old name: " + e.oldValue + ", new name: " + e.newValue);
    }
    log(e);
});

//hierarchy change listener
databox.addHierarchyChangeListener(function(e:HierarchyChangeEvent):void{
    log(e);
});

databox.name = "Box A";

var dataA:Data = createData("A");
var dataB:Data = createData("B");
dataA.name = "AA";
dataB.setClient("NO", 1);
databox.clear();

var dataC:Data = createData("C");
var dataD:Data = createData("D");
var dataE:Data = createData("E");
var dataF:Data = createData("F");

databox.remove(dataC);

databox.moveToTop(dataE);

databox.moveToBottom(dataD);

textarea.text += "Root datas: " + databox.roots.toArray().toString();
}

private function createData(name:String):Data{
    var data:Data = new Data();
    data.name = name;
    databox.add(data);
    return data;
}

private function log(e:*) :void{
    var text:String = "";
    if(e is DataBoxChangeEvent){
        text += "DataBoxChangeEvent - kind: " + (e as DataBoxChangeEvent).kind + ", data: " +
            (e as DataBoxChangeEvent).data + ", datas: " + (e as DataBoxChangeEvent).datas;
    }else if(e is HierarchyChangeEvent){
        text += "HierarchyChangeEvent - data: " + (e as HierarchyChangeEvent).data + ", oldIndex: " +
            (e as HierarchyChangeEvent).oldIndex + ", newIndex: " + (e as HierarchyChangeEvent).newIndex;
    }else if(e is PropertyChangeEvent){
        text += "PropertyChangeEvent - kind: " + (e as PropertyChangeEvent).kind + ", property: " +
            (e as PropertyChangeEvent).property + ", oldValue: " +
            (e as PropertyChangeEvent).oldValue + ", newValue: " + (e as PropertyChangeEvent).newValue;
    }
    textarea.text += text + "\n";
}

]]>
</mx:Script>
<mx:TextArea id="textarea" height="100%" width="100%" />

```

```
</mx:Application>
```

Output results:

PropertyChangeEvent - kind: update, property: name, oldValue: DataBox, newValue: Box A
DataBoxChangeEvent - kind: add, data: A, datas: null
DataBoxChangeEvent - kind: add, data: B, datas: null
PropertyChangeEvent - kind: update, property: name, oldValue: A, newValue: AA
PropertyChangeEvent - kind: update, property: C:NO, oldValue: null, newValue: 1
DataBoxChangeEvent - kind: clear, data: null, datas: AA,B
DataBoxChangeEvent - kind: add, data: C, datas: null
DataBoxChangeEvent - kind: add, data: D, datas: null
DataBoxChangeEvent - kind: add, data: E, datas: null
DataBoxChangeEvent - kind: add, data: F, datas: null
DataBoxChangeEvent - kind: remove, data: C, datas: null
HiberarchyChangeEvent - data: E, oldIndex: 1, newIndex: 0
HiberarchyChangeEvent - data: D, oldIndex: 1, newIndex: 2
Root datas: E,F,D

Quick Finder Mechanism

QuickFinder is used to find out the kind of elements with same property quickly, such as alarms with same security level and etc.

Let's explain how to use QuickFinder, tacking elements with certain name for an example:

At first, create an QuickFinder bundled with name property. The first property is the data container that will be the target of QuickFinder, and the second one is the property you are looking for.

```
var nameFinder:QuickFinder = new QuickFinder(databox, "name");
```

Secondly, find out all elements named "groupA_1".

```
var datas:Array = nameFinder.find("groupA_1");
```

Introduction of QuickFinder in detail:

Construct QuickFinder

```
public function QuickFinder(dataBox:DataBox, propertyName:String,
    propertyType:String = Consts.PROPERTY_TYPE_ACCESSOR,
    valueFunction:Function = null, filterFunction:Function = null)
```

dataBox : data container, the target of QuickFinder.

propertyName : property name.

propertyType : property type, the default one is Consts.PROPERTY_TYPE_ACCESSOR stands for being able to getting this property directly, such as data.name.

Three property types:

Consts.PROPERTY_TYPE_ACCESSOR : getting property directly by datapropertyName, just like the javaBean property of java.

Consts.PROPERTY_TYPE_CLIENT : getting property by data.getClient(propertyName), client property.

Consts.PROPERTY_TYPE_STYLE : getting property by data.getStyle(propertyName), style property.

valueFunction : the function to get results. The default way is by the three methods as above. You can custom the function valueFunction yourself.

filterFunction : filter function is to filter elements again based on above custom parameters.

function(data:IData):Boolean{}

Returning true or false to indicate whether participate in finding.

How to Find

QuickFinder provides two finding functions:

```
public function findFirst(value:Object):IData
public function find(value:Object):Array
```

findFirst : return the first found out element.

find : return all found out elements.

Sample

Construct one property QuickFinder and one client property QuickFinder.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute" creationComplete="init(event)" >
<mx:Script>
<![CDATA[
import mx.events.FlexEvent;

import twaver.Data;
import twaver.DataBox;
import twaver.IData;
import twaver.QuickFinder;

private var databox:DataBox;

protected function init(event:FlexEvent):void
{
databox = new DataBox();

var data:IData;
//fill datas
for(var i:int = 0; i < 20; i++){
data = new twaver.Data();
data.name = "groupA_" + i%3;
data.setClient("NO", i%5);
databox.add(data);
}

for(i = 0; i < 20; i++){
data = new Data();
data.name = "groupB_" + i%3;
data.setClient("NO", i%5);
databox.add(data);
}

var nameFilder:QuickFinder = new QuickFinder(databox, "name");
var noFilder:QuickFinder = new QuickFinder(databox, "NO", twaver.Consts.PROPERTY_TYPE_CLIENT);

data = nameFilder.findFirst("groupA_1");
log("nameFilder.findFirst(\"groupA_1\")", data);

var datas:Array = nameFilder.find("groupA_1");
log("nameFilder.find(\"groupA_1\")", datas);

data = noFilder.findFirst(1);
log("noFilder.findFirst(1)", data);

datas = noFilder.find(1);
log("noFilder.find(1)", datas);

}

private function getDataDescription(data:IData):String{
return "name:" + data.name + ", NO:" + data.getClient("NO");
}

private function log(title:String, data:*.):void{
textarea.text += title + ":\n";
if(!data){
```

```

        return;
    }
    if(data is Data){
        textarea.text += " " + getDataDescription(data as Data) + "\n";
    }else if(data is Array){
        (data as Array).forEach(function(item:*, index:int, array:Array):void{
            textarea.text += " " + getDataDescription(item as Data) + "\n";
        });
    }
}

]]>
</mx:Script>
<mx:TextArea id="textarea" width="100%" height="100%" />
</mx:Application>

```

Output results:

```

nameFilder.findFirst("groupA_1"):
name:groupA_1, NO:1
nameFilder.find("groupA_1"):
name:groupA_1, NO:1
name:groupA_1, NO:4
name:groupA_1, NO:2
name:groupA_1, NO:0
name:groupA_1, NO:3
name:groupA_1, NO:1
name:groupA_1, NO:4
noFilder.findFirst(1):
name:groupA_1, NO:1
noFilder.find(1):
name:groupA_1, NO:1
name:groupA_0, NO:1
name:groupA_2, NO:1
name:groupA_1, NO:1
name:groupB_1, NO:1
name:groupB_0, NO:1
name:groupB_2, NO:1
name:groupB_1, NO:1

```

Selection Model Appliance

SelectionModel is to maintain the selection information of elements in DataBox. All adding, clearing selection for elements are realized by SelectionModel.Make "data" element in databox selected by calling following API:

```
databox.selectionModel.appendSelection(data);
```

View components own their SelectionModel, and bundle with their DataBox. For example: make element selected in topology by calling following API:

```
network.selectionModel.appendSelection(element);
```

Construct SelectionModel

You can get default SelectionModel of DataBox by databox.selectionModel. The SelectionModel of view components is the same as DataBox by default. Invoke it by following code: view.selectionModel, for example, network.selectionModel and tree.selectionModel.

User can setup your SelectionModel both for DataBox and view components. Construct a custom SelectionModel at first.

```
var mySelectionModel: SelectionModel = new SelectionModel(databox);
```

Then bundle SelectionModel with databox/view components in constructor function.

```
databox.selectionModel = mySelectionModel;
network.selectionModel = mySelectionModel;
...
```

The SelectionModel of view components is the one of databox by default. In this way, TWaver can make sure synchronize for all view components selection. Constructing SelectionModel for view components separately is to deal with the independent selection issue.

How to Make Element Selected and How to Remove Selection

The following functions of SelectionModel can realize selection appending, element selected, all elements selected, the selection of all elements removed and etc.

```
//append selected element, the parameter can be both one single element and elements collection
public function appendSelection(datas:Object):void

//setting selection for elements is different with appending selected element.
//This function will remove all original selection at first.
public function setSelection(datas:Object):void

//select all elements in databox
public function selectAll():void

//remove selected element, the parameter can be both one single element and elements collection
public function removeSelection(datas:Object):void
```

```
//clear all selected elements
public function clearSelection():void
```

Get Selection Information of Elements

You can get the selection information in SelectionModel. Such as all selected elements, the quantity of selected elements, the selection status of elements and etc.

```
//get all selected elements. Note: the return result is passed by
//reference. So you shouldn't do any changes to this collection.
public function get selection():ICollection

//get new collection from SelectionModel. Note: the return result is the
//new constructed collection which is different with above method.
//matchFunction: the parameter is IData and the return result is
//true/false. The false result stands for excluding this element.
public function toSelection(matchFunction:Function = null):ICollection

//the quantity of selected elements
public function get count():int

//the selection status of element
public function contains(data:IData):Boolean

//get the last element in selected elements collection
public function get lastData():IData

//get the first element in selected elements collection
public function get firstData():IData

//whether the element is selected
public function isSelectable(data:IData):Boolean
```

Selection Change Event

SelectionModel maintains selection state of all elements. The status changes can dispatch respondent SelectionChangeEvent. For example: calling method to append element selection would dispatch APPEND SelectionChangeEvent.

TWaver defined class named twaver.SelectionChangeEvent to describe selection change event.

SelectionChangeEvent extends Event class, and its event type is twaver.Consts.EVENT_SELECTION_CHANGE.

```
public function SelectionChangeEvent(kind:String, datas:Array = null, bubbles:Boolean = false, cancelable:Boolean
= false){
    super(Consts.EVENT_SELECTION_CHANGE, bubbles, cancelable);
    this.kind = kind;
    this.datas = datas;
}
```

SelectionChangeEvent includes two properties. One stands for selection event type, another is for event data.

```
public var kind:String;
public var datas:Array;
```

There are five types of SelectionChangeEvent: append, set, remove, all, clear. They are respond to five functions: appendSelection, setSelection, removeSelection, selectAll, clearSelection.

```
//append
public static const APPEND:String = "append";
//set
public static const SET:String = "set";
//remove
public static const REMOVE:String = "remove";
//all
public static const ALL:String = "all";
//clear
public static const CLEAR:String = "clear";
```

Selection Model

SelectionModel provides three models: multiple-selection, single-selection and unselectable which are in common use for UI. For example: Flex DataGrid contains multiple-selection model and single-selection model. At the same time, SelectionModel support the data layer of view components SelectionModel, taking multiple selection as default.

```
SelectionModel.NONE_SELECTION
SelectionModel.SINGLE_SELECTION
SelectionModel.MULTIPLE_SELECTION
```

Change the selection model:

```
selectionModel.selectionMode = SelectionModel.NONE_SELECTION
```



Note: when the SelectionModel changes from multiple-selection to single-selection or unselectable, TWaver will invoke clear method to clear all selection status.

Sample

Let's show how to do element selection, how to change element selection and how to switch the selection model.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
    xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute" creationComplete="init(event);">
    <mx:Script>
        <![CDATA[
            import mx.events.FlexEvent;

            import twaver.Data;
            import twaver.DataBox;
            import twaver.IData;
            import twaver.SelectionChangeEvent;
            import twaver.SelectionModel;
            import twaver.network.Network;

            private var databox:DataBox;
            private var selectionModel:SelectionModel;
```

```
protected function init(event:FlexEvent):void
{
    databox = new DataBox();
    selectionModel = databox.selectionModel;

    for(var i:int = 0; i<10; i++){
        var data:Data = new Data(i);
        data.name = "data_" + i;
        databox.add(data);
    }

    selectionModel.addSelectionChangeListener(function(e:SelectionChangeEvent):void{
        logSelectionChangeEvent(e);
    });

    //selectionChangeEvents: append/remove/clear
    textarea.text += "\nselection change event: append/remove/clear\n";
    selectionModel.appendSelection(databox.getDataByID(0));
    selectionModel.appendSelection(databox.getDataByID(1));
    selectionModel.removeSelection(databox.getDataByID(0));
    selectionModel.setSelection([databox.getDataByID(2),databox.getDataByID(6)]);
    selectionModel.appendSelection([databox.getDataByID(2),databox.getDataByID(3),databox.getDataByID(4)]);

    //single selection mode
    textarea.text += "\nsetting single selection mode, remove all selected elements at first,
        then append no more than one element at the same time \n";
    selectionModel.selectionMode = SelectionModel.SINGLE_SELECTION;
    //single selection mode, select only last data
    selectionModel.appendSelection([databox.getDataByID(2),databox.getDataByID(3),databox.getDataByID(4)]);
    textarea.text += selectionModel.count;

    //none selection mode, unable select data any more
    textarea.text += "\nnone selection mode, remove all selected elements at first,
        then unable select data any more\n";
    selectionModel.selectionMode = SelectionModel.NONE_SELECTION;
    selectionModel.appendSelection([databox.getDataByID(2),databox.getDataByID(3),databox.getDataByID(4)]);

    selectionModel.selectionMode = SelectionModel.MULTIPLE_SELECTION;
    //setting filterFunction , filter data 0;
    textarea.text += "\nsetting filterFunction , filter data 0\n";
    selectionModel.filterFunction = function(data:IData):Boolean{
        return data.id != 0;
    };
    selectionModel.appendSelection([databox.getDataByID(2),databox.getDataByID(3),databox.getDataByID(4)]);

    var network:Network = new Network();
    network.selectionModel
}

private function logSelectionChangeEvent(e:SelectionChangeEvent):void{
    textarea.text += "Kind: " + e.kind + ", datas: " + e.datas.toString() + "\n";
}
]]>
</mx:Script>
<mx:TextArea id="textarea" height="100%" width="100%" />
</mx:Application>
```

Output results:

```
selection change event: append/remove/clear
Kind: append, datas: data_0
Kind: append, datas: data_1
Kind: remove, datas: data_0
Kind: set, datas: data_1,data_2,data_6
```

Kind: append, datas: data_3,data_4

setting single selection mode, remove all selected elements at first, then append no more than one element at the same time

Kind: clear, datas: data_2,data_6,data_3,data_4

Kind: append, datas: data_4

1

none selection mode, remove all selected elements at first, then unable select data any more

Kind: clear, datas: data_4

setting filterFunction , filter data 0

Kind: append, datas: data_2,data_3,data_4

Network Component


This chapter we introduce the design and usage of topology components including hierarchy of topology, setting background, filter and swithing interactive models:

- [Network Hierarchy](#)
- [Network Adding Backgroud](#)
- [Network Filter](#)
- [Network Function for Style Rule](#)
- [Network GUI Interaction](#)

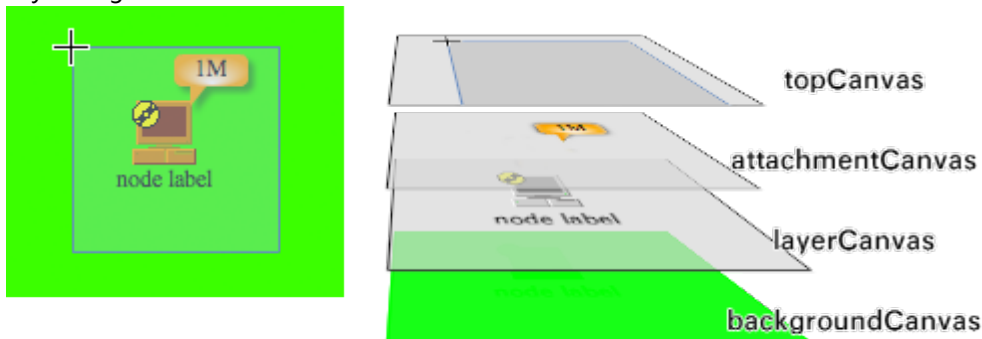
Network Hierarchy

All components in topology are put into rootCanvas. The following is its hierarchy:

- rootCanvas - - main canvas, contains all topology components
- topCanvas - - top canvas, is to draw interaction frame, adjust the size of drag block
- attachmentCanvas - - top components placement canvas
- layerCanvas - - the view of all components canvas
 - layer n
 - layer ...
 - default layer
- bottomCanvas - - bottom canvas, is used for drawing shade on background
- backgroundCanvas - - background canvas including color and picture features

 Attachment will be attached with ElementUI by default, If the value of showInAttachemtnCanvas is true, Attachment will be palced in attachmentCanvas layer

Layer diagram:



User can get any canvas in Network, then add compoments as requirement:

```
public function get rootCanvas():Canvas
public function get topCanvas():Canvas
public function get attachmentCanvas():Canvas
public function get layerCanvas():Canvas
public function get bottomCanvas():Canvas
public function get backgroundCanvas():Canvas
```

Let's add button in layer as above to deep impression:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
  applicationComplete="init()">
  <mx:Script>
    <![CDATA[
      import mx.controls.Button;

      import twaver.*;
      import twaver.network.Network;

      [Embed(source="resource/images/linkStatus2.png")]
      public static const linkStatus2:Class;
      private function init():void{
```

```

Utils.registerImageClass("linkStatus2",linkStatus2);
var box:ElementBox = network.elementBox;

var node:Node=new Node();
node.setLocation(80,30);
node.name="node label";
node.alarmState.increaseNewAlarm(AlarmSeverity.MAJOR);
node.setStyle(Styles.ICON_NAME,"linkStatus2");
box.add(node);

var buttonInRootCanvas:Button=new Button();
buttonInRootCanvas.label="button in rootCanvas";
buttonInRootCanvas.x=100;
buttonInRootCanvas.y=40;
buttonInRootCanvas.height=100;
network.rootCanvas.addChild(buttonInRootCanvas);

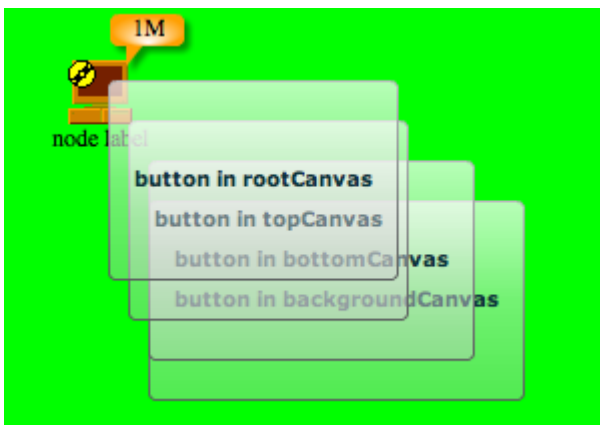
var buttonInTopCanvas:Button=new Button();
buttonInTopCanvas.label="button in topCanvas";
buttonInTopCanvas.x=110;
buttonInTopCanvas.y=60;
buttonInTopCanvas.height=100;
network.topCanvas.addChild(buttonInTopCanvas);

var buttonInBottomCanvas:Button=new Button();
buttonInBottomCanvas.label="button in bottomCanvas";
buttonInBottomCanvas.x=120;
buttonInBottomCanvas.y=80;
buttonInBottomCanvas.height=100;
network.bottomCanvas.addChild(buttonInBottomCanvas);

var buttonInBackgroundCanvas:Button=new Button();
buttonInBackgroundCanvas.label="button in backgroundCanvas";
buttonInBackgroundCanvas.x=120;
buttonInBackgroundCanvas.y=100;
buttonInBackgroundCanvas.height=100;
network.backgroundCanvas.addChild(buttonInBackgroundCanvas);

box.setStyle(Styles.BACKGROUND_TYPE, Consts.BACKGROUND_TYPE_VECTOR);
box.setStyle(Styles.BACKGROUND_VECTOR_FILL_COLOR,0x00ff00);
}
]]>
</mx:Script>
<mx:Panel title="Hello TWaver! - Network Layer Test" width="100%" height="100%">
  <twaver:Network id="network" width="100%" height="100%"/>
</mx:Panel>
</mx:Application>

```



Network Adding Background

TWaver Flex supports multi style background including raster formatted picture, color, color gradient, basic shape (such as rectangle, start and etc), combination of picture and shape and etc.

The background property of TWaver Flex is set in Model layer. Both ElementBox and network element in topology can set background. Due to topology can only represent network elements of current subnetwork, the background of Network refer to the background of ElementBox and Subnetwork, the former is the top subnetwork background and the latter is each subnetwork background. The background property is set in Model layer to offer convenient to serialize. TWaver take background as parts of data, it shouldn't be lost.

Setting data element style can affect TWaver Flex background. First user need to set background style such as color, picture, basic shape, combination of picture and shape and etc, then set properties of background such as name, shape style, color, gradient type, gradient color, gap, alignment and etc.

Background Style

twaver.Consts

```
//No background
public static const BACKGROUND_TYPE_NONE:String = "none";
//Picture background
public static const BACKGROUND_TYPE_IMAGE:String = "image";
//Vector background(shape, color, color gradient)
public static const BACKGROUND_TYPE_VECTOR:String = "vector";
//Picture on the top, vector at the bottom
public static const BACKGROUND_TYPE_IMAGE_VECTOR:String = "image.vector";
//Vector on the top, picture at the bottom
public static const BACKGROUND_TYPE_VECTOR_IMAGE:String = "vector.image";
```

Background Stylesheet

twaver.Styles

```
public static const BACKGROUND_TYPE:String = "background.type";
public static const BACKGROUND_IMAGE:String = "background.image";
public static const BACKGROUND_IMAGE_SCOPE:String = "background.image.scope";
public static const BACKGROUND_IMAGE_STRETCH:String = "background.image.stretch";
public static const BACKGROUND_IMAGE_PADDING:String = "background.image.padding";
public static const BACKGROUND_IMAGE_PADDING_LEFT:String = "background.image.padding.left";
public static const BACKGROUND_IMAGE_PADDING_RIGHT:String = "background.image.padding.right";
public static const BACKGROUND_IMAGE_PADDING_TOP:String = "background.image.padding.top";
public static const BACKGROUND_IMAGE_PADDING_BOTTOM:String = "background.image.padding.bottom";
public static const BACKGROUND_IMAGE_SHAPE:String = "background.image.shape";
public static const BACKGROUND_IMAGE_OUTLINE_WIDTH:String = "background.image.outline.width";
public static const BACKGROUND_IMAGE_OUTLINE_COLOR:String = "background.image.outline.color";
public static const BACKGROUND_IMAGE_OUTLINE_ALPHA:String = "background.image.outline.alpha";
public static const BACKGROUND_IMAGE_JOINT_STYLE:String = "background.image.joint.style";
public static const BACKGROUND_IMAGE_CAPS_STYLE:String = "background.image.caps.style";
public static const BACKGROUND_IMAGE_SCALE_MODE:String = "background.image.scale.mode";
public static const BACKGROUND_IMAGE_PIXEL_HINTING:String = "background.image.pixel.hinting";
public static const BACKGROUND_VECTOR_FILL:String = "background.vector.fill";
public static const BACKGROUND_VECTOR_FILL_COLOR:String = "background.vector.fill.color";
public static const BACKGROUND_VECTOR_FILL_ALPHA:String = "background.vector.fill.alpha";
public static const BACKGROUND_VECTOR_SCOPE:String = "background.vector.scope";
public static const BACKGROUND_VECTOR_SHAPE:String = "background.vector.shape";
public static const BACKGROUND_VECTOR_OUTLINE_WIDTH:String = "background.vector.outline.width";
public static const BACKGROUND_VECTOR_OUTLINE_COLOR:String = "background.vector.outline.color";
```

```
public static const BACKGROUND_VECTOR_OUTLINE_ALPHA:String = "background.vector.outline.alpha";
public static const BACKGROUND_VECTOR_3D_DEEP:String = "background.vector.3d.deep";
public static const BACKGROUND_VECTOR_JOINT_STYLE:String = "background.vector.joint.style";
public static const BACKGROUND_VECTOR_CAPS_STYLE:String = "background.vector.caps.style";
public static const BACKGROUND_VECTOR_SCALE_MODE:String = "background.vector.scale.mode";
public static const BACKGROUND_VECTOR_PIXEL_HINTING:String = "background.vector.pixel.hinting";
public static const BACKGROUND_VECTOR_GRADIENT:String = "background.vector.gradient";
public static const BACKGROUND_VECTOR_GRADIENT_COLOR:String = "background.vector.gradient.color";
public static const BACKGROUND_VECTOR_GRADIENT_ALPHA:String = "background.vector.gradient.alpha";
public static const BACKGROUND_VECTOR_PADDING:String = "background.vector.padding";
public static const BACKGROUND_VECTOR_PADDING_LEFT:String = "background.vector.padding.left";
public static const BACKGROUND_VECTOR_PADDING_RIGHT:String = "background.vector.padding.right";
public static const BACKGROUND_VECTOR_PADDING_TOP:String = "background.vector.padding.top";
public static const BACKGROUND_VECTOR_PADDING_BOTTOM:String = "background.vector.padding.bottom";
```

For example:

Let's show how to set background with picture, vector gradient and pure color

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:tw="http://www.servasoftware.com/2009/twaver/flex"
    applicationComplete="init()">
<mx:Script>
<![CDATA[
    import twaver.*;
    import twaver.network.Network;

    [Embed(source="resource/images/background.jpg")]
    public static const background:Class;
    private function init():void{
        //Register picture, All picture in TWaver need to be registered before use
        Utils.registerImageClass("background",background);
        var box:ElementBox = network.elementBox;
        // Set box background type to be picture type
        box.setStyle(Styles.BACKGROUND_TYPE, Consts.BACKGROUND_TYPE_IMAGE);
        //Set background picture
        box.setStyle(Styles.BACKGROUND_IMAGE,"background");

        var subnetworkA:SubNetwork=addSubnetwork(box,"subnetwork A");
        subnetworkA.location=new Point(100,100);
        //Set subnetwork background type to be vector type
        subnetworkA.setStyle(Styles.BACKGROUND_TYPE, Consts.BACKGROUND_TYPE_VECTOR);
        //Set filling color and gradient color
        subnetworkA.setStyle(Styles.BACKGROUND_VECTOR_FILL_COLOR, 0x9CB5D8);
        subnetworkA.setStyle(Styles.BACKGROUND_VECTOR_GRADIENT_COLOR, 0xFFFFFF);
        //Set gradient style to be vertical spread
        subnetworkA.setStyle(Styles.BACKGROUND_VECTOR_GRADIENT, Consts.GRAIDENT_SPREAD_VERTICAL);

        var subnetworkB:SubNetwork=addSubnetwork(box,"subnetwork B");
        subnetworkB.location=new Point(300,100);
        //Set background type to be vector
        subnetworkB.setStyle(Styles.BACKGROUND_TYPE, Consts.BACKGROUND_TYPE_VECTOR);
        //Set background color
        subnetworkB.setStyle(Styles.BACKGROUND_VECTOR_FILL_COLOR, 0x0000ff);

        var link:Link=new Link(subnetworkA,subnetworkB);
        box.add(link);
    }

    private function addSubnetwork(box:ElementBox,name:String):SubNetwork{
        var subnetwork:SubNetwork=new SubNetwork();
```

```

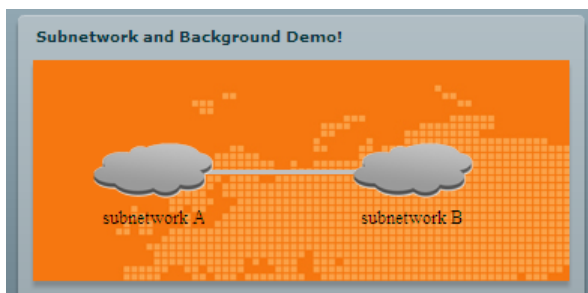
subnetwork.name=name;
box.add(subnetwork);
var from:Node=new Node();
from.name = "from";
from.location = new Point(20, 20);
box.add(from);
subnetwork.addChild(from);
var to:Node = new Node();
to.name = "to";
to.location = new Point(150, 60);
box.add(to);
subnetwork.addChild(to);
var link:Link = new Link(from,to);
link.name = "hello TWaver!";
box.add(link);
subnetwork.addChild(link);
return subnetwork;
}

]]>
</mx:Script>
<mx:Panel title="Subnetwork and Background Demo!" width="100%" height="100%">
  <tw:Network id="network" backgroundColor="0x00ff00" width="100%" height="100%">
    </tw:Network>
  </mx:Panel>
</mx:Application>

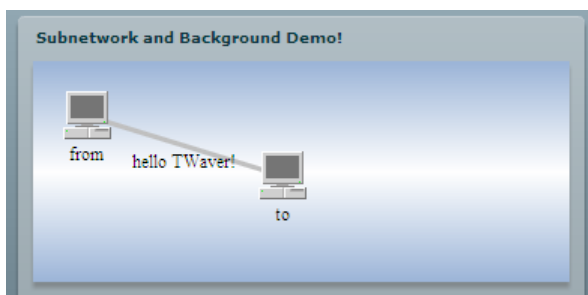
```

Results:

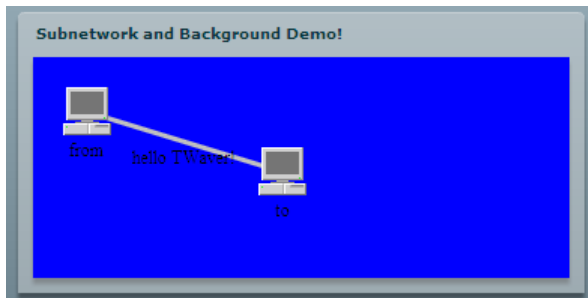
Picture background:



Gradient color background:



Pure color background:



Network Filter

Continuing the style of TWaver Java, TWaver Flex offers a series of filter including visible filter, movable filter, editable filter and etc. TWaver Flex realize one data modle, different view information and different operation model by setting filter. Generally user can get different interactions and views by setting permission and setting network element type.

Network filter includes:

```
//visible filter
public function get/set visibleFunction():Function
//movable filter
public function get/set movableFunction():Function
//editable filter
public function get/set editableFunction():Function
```

Let's give an example to show filter use. Pay attention that parameter type is Element, and the return value is Boolean. The example show how to set nodes has children to be visible:

```
network.visibleFunction = function(node:IElement):Boolean{
    retrun node.childrenCount > 0;
};
```

Network Function for Style Rule

TWaver Flex set component style by rule-making functions including tree view, color rendering of node in topology, border, bubble and etc. This chapter will introduce related regulations in topology.

Color rendering, border color, alarm color, alarm text, bundle line text, toolbar prompt text of network element in topology is realized by rule-making functions. User also can customize rule-making functions to meet requirements.

Style Rule of Network:

elementUIFunction : Function to create view for network element, to describe how network element to create view. Default it will create ElementUI by elementUIClass property of network element.

labelFunction : Label function for network element text, is to set label for network element.

toolTipFunction : The hint of toolbar

innerColorFunction : Rendering color function inner network element

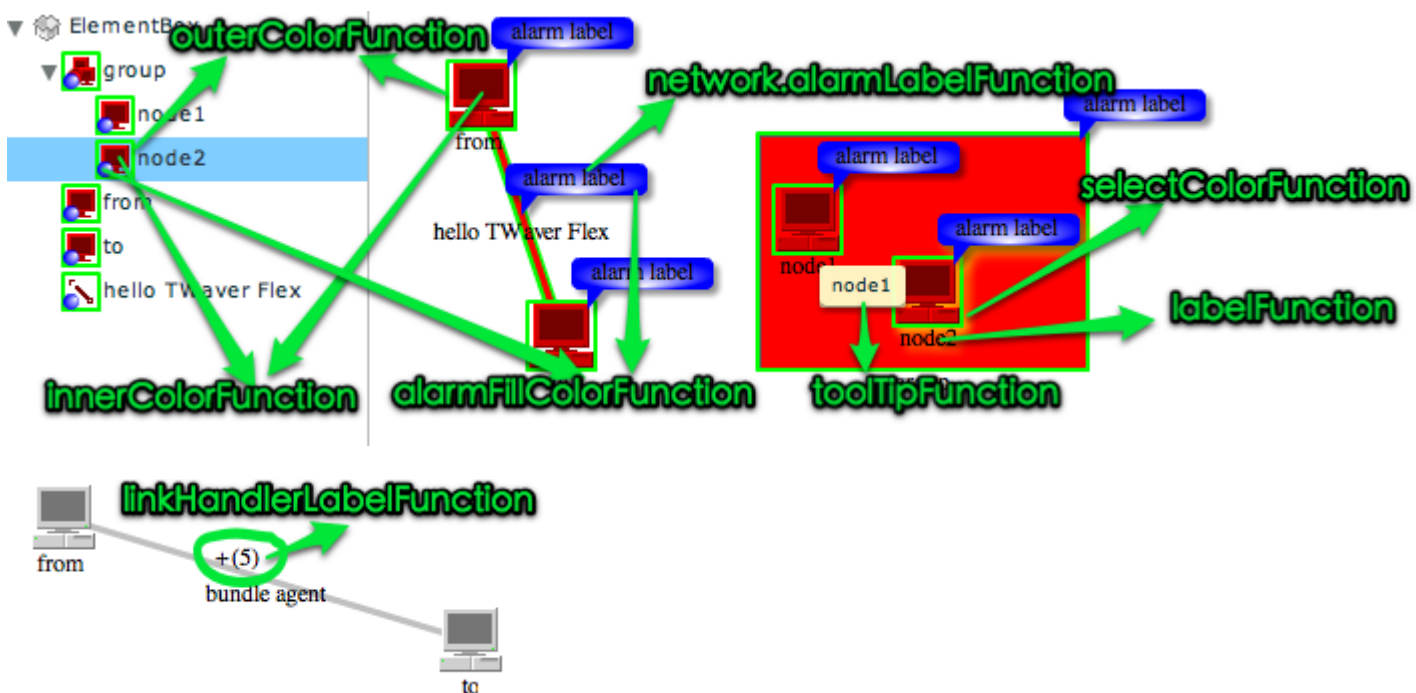
outerColorFunction : Border color of network element

selectColorFunction : Selected color of network element

alarmFillColorFunction : Alarm bubble color

alarmLabelFunction : Alarm bubble text

linkHandlerLabelFunction : Bundle link text



Default

The default definition rule of twaver.Defaults, list as below:

```
//twaver.Defaults
```



```
//elementUIFunction
public static var ELEMENTUI_FUNCTION:Function = function(network:Network, element:IElement):ElementUI{
    var clazz:Class = element.elementUIClass;
    if(clazz != null){
        return new clazz(network, element);
    }
    return null;
}

//labelFunction
public static var TREE_LABEL_FUNCTION:Function = function(data:IData):String{
    if(data is IElement){
        var label:String = IElement(data).getStyle(Styles.TREE_LABEL, false);
        if(label != null){
            return label;
        }
    }
    return data.name;
};

//toolTipFunction
public static var TOOLTIP_FUNCTION:Function = function(data:IData):String{
    return data.toolTip;
};

//innerColorFunction
public static var INNER_COLOR_FUNCTION:Function = function(data:IData):Object{
    if(data is IElement){
        var element:IElement = IElement(data);
        var severity:AlarmSeverity = element.alarmState.highestNativeAlarmSeverity;
        if(severity != null){
            return severity.color;
        }
        return element.getStyle(Styles.INNER_COLOR, false);
    }
    return null;
};

//outerColorFunction
public static var OUTER_COLOR_FUNCTION:Function = function(data:IData):Object{
    if(data is IElement){
        var element:IElement = IElement(data);
        var severity:AlarmSeverity = element.alarmState.propagateSeverity;
        if(severity != null){
            return severity.color;
        }
        return element.getStyle(Styles.OUTER_COLOR, false);
    }
    return null;
};

//selectColorFunction : selected color of network element
public static var SELECT_COLOR_FUNCTION:Function = function(data:IData):Object{
    if(data is IElement){
        return IElement(data).getStyle(Styles.SELECT_COLOR);
    }
    return Consts.COLOR_DARK;
};

//alarmFillColorFunction : alarm bubble color
public static var ALARM_FILL_COLOR_FUNCTION:Function = function(data:IData):Object{
    if(data is IElement){
        var severity:AlarmSeverity = IElement(data).alarmState.highestNewAlarmSeverity;
        if(severity != null){
            return severity.color;
        }
    }
}
```

```

    }
    return null;
};

//alarmLabelFunction
public static var ALARM_LABEL_FUNCTION:Function = function(element:IElement):Object{
    var severity:AlarmSeverity = element.alarmState.highestNewAlarmSeverity;
    if(severity != null){
        var label:String = element.alarmState.getNewAlarmCount(severity) + severity.nickName;
        if(element.alarmState.hasLessSevereNewAlarms){
            label += "+";
        }
        return label;
    }
    return null;
};

//linkHandlerLabelFunction
public static var LINK_HANDLER_LABEL_FUNCTION:Function = function(link:Link):Object{
    if(link.isBundleAgent){
        return "(" + link.bundleCount + ")";
    }
    return null;
}

```

Rule Making

```

<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
    applicationComplete="init()">
<mx:Script>
<![CDATA[
    import twaver.*;
    import twaver.network.Network;

    private var box:ElementBox;
    private var number:int;
    private function init():void{
        number=0;
        box=network.elementBox;
        tree.dataBox=box;

        var group:Group=new Group();
        group.name="group";
        box.add(group);
        group.addChild(createTWaverNode("node1",200,100));
        group.addChild(createTWaverNode("node2",300,130));
        group.expanded=true;

        var from:Node=createTWaverNode("from",30,30);
        var to:Node=createTWaverNode("to",70,150);
        var link:Link=new Link(from,to);
        link.name="hello TWaver Flex";
        box.add(link);

        //Color rendering of tree node
        tree.innerColorFunction=function(data:IData):Object{
            return 0xff0000;
        };

        //Border color of tree node
    ]]>

```

```

tree.outerColorFunction= function(data:IData):Object{
    return 0x00ff00;
};

//The left-bottom ball's color, blank if return value is null
tree.alarmFillColorFunction = function(data:IData):Object{
    if(data is Element&&!(data as Element).alarmState.isEmpty()){
        return 0x0000ff;
    }
    return null;
};

//network is under the same rule
//innerColorFunction - rendering color of node
//outerColorFunction - border color
//alarmFillColorFunction - alarm bubble color, blank if alarm text is null
//alarmLabelFunction - alarm text
//Body -rendering color
network.innerColorFunction=function(data:IData):Object{
    return 0xff0000;
};

//border color of node
network.outerColorFunction= function(data:IData):Object{
    return 0x00ff00;
};

//alarm bubble color, blank if alarmLabel return null or color is null
network.alarmFillColorFunction = function(data:IData):Object{
    if(data is Element&&!(data as Element).alarmState.isEmpty()){
        return 0x0000ff;
    }
    return null;
};

network.alarmLabelFunction=function(element:IElement):Object{
    if(!element.alarmState.isEmpty()){
        return "alarmLabel";
    }
    return null;
};

network.selectColorFunction=function(element:IElement):Object{
    return 0xffff00;
}
}

private function createTWaverNode(name:String,x:int,y:int):Node{
    var node:Node=new Node();
    node.name=name;
    node.toolTip=name;
    node.alarmState.increaseNewAlarm(AlarmSeverity.MAJOR);
    node.setClient("number",number++);
    node.setLocation(x,y);
    box.add(node);
    return node;
}

]]>
</mx:Script>
<mx:Panel title="Hello TWaver! - Tree&Network Style Functions" width="100%" height="100%">
<mx:HDividedBox width="100%" height="100%">
    <twaver:Tree id="tree" width="30%" height="100%" />
    <twaver:Network id="network" width="70%" height="100%" />
</mx:HDividedBox>
</mx:Panel>

```

```
</mx:Application>
```

Network GUI Interaction

InteractionHandler

Mouse and keyboard interaction in topology composed by a set of InteractionHandler. And each InteractionHandler contains independent event listening and uninstalling operation. InteractionHandler defined as below:

```
package twaver.network.interaction{
    public interface InteractionHandler{
        function installListeners():void;
        function uninstallListeners():void;
    }
}
```

Customize Interaction Model in Topology

Generally we add related mouse/keyboard listener by method installListeners(), uninstall these listener by method uninstallListeners().

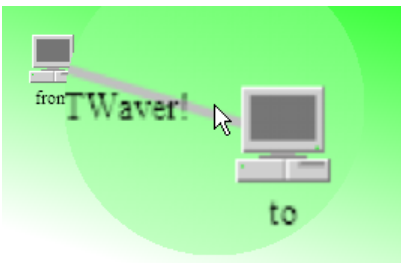
Customizing topology interaction needs to specify a set of InteractionHandler. TWaver will realize all methods uninstallListeners() in original interactionHandlers then invoke all methods installListeners() in new interactionHandlers.

Set Interaction Model for Topology

```
public function set interactionHandlers(interactionHandlers:ICollection):void
```

Set magnifier interaction:

```
network.interactionHandlers = new Collection([
    new SelectInteractionHandler(network),
    new EditInteractionHandler(network),
    new MoveInteractionHandler(network),
    new DefaultInteractionHandler(network),
    new MapFilterInteractionHandler(network, Consts.MAP_FILTER_MAGNIFY)
]);
```

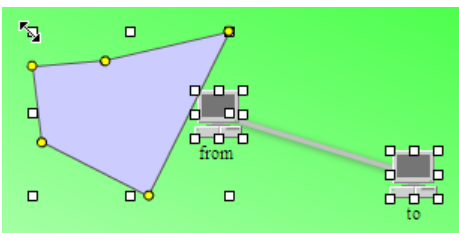


Commonly Used Interaction Model

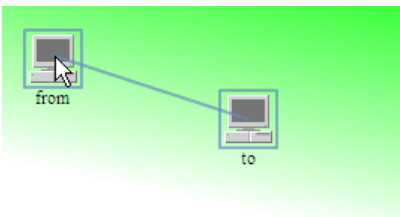
Network pre-defined several commonly used interaction models including default, editable, creating links, creating shaplink and polygonal network element models. Moreover, magnifier interaction and fish-eye interaction also are easy to implement.

```
public function setDefaultInteractionHandlers(lazyMode:Boolean = false):void
public function setEditInteractionHandlers(lazyMode:Boolean = false):void
public function setCreateLinkInteractionHandlers(linkClass:Class = null):void
public function setCreateShapeLinkInteractionHandlers(shapeLinkClass:Class = null):void
public function setCreateShapeNodeInteractionHandlers(shapeNodeClass:Class = null):void
```

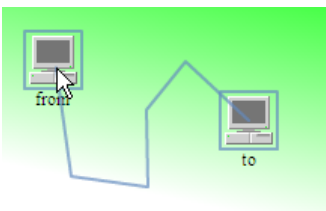
Editable Model



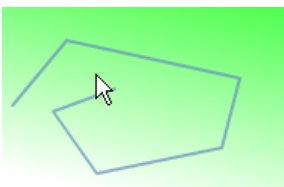
Creating Link Interaction Model



Creating ShapeLink Interaction Model



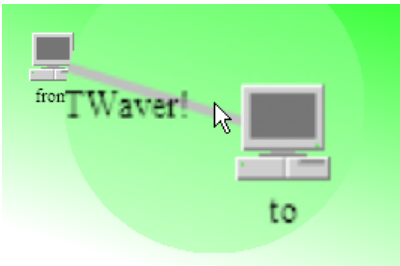
Creating Polygon Interaction Model



Magnifier Interaction:

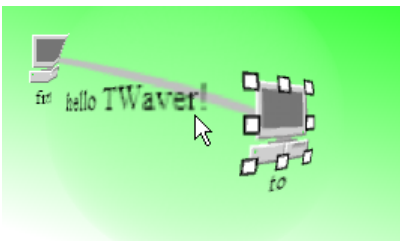
```
network.interactionHandlers = new Collection([
    new SelectInteractionHandler(network),
    new EditInteractionHandler(network),
    new MoveInteractionHandler(network),
```

```
new DefaultInteractionHandler(network),
new MapFilterInteractionHandler(network, Consts.MAP_FILTER_MAGNIFY)
]);
```



Fish-eye Interaction:

```
network.interactionHandlers = new Collection([
new SelectInteractionHandler(network),
new EditInteractionHandler(network),
new MoveInteractionHandler(network),
new DefaultInteractionHandler(network),
new MapFilterInteractionHandler(network)
]);
```



The following example realize the switch between any interactio models, and implements creating network element by dragging button:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:tw="http://www.servasoftware.com/2009/twaver/flex"
  applicationComplete="init()">
  <mx:Script>
    <![CDATA[
      import mx.core.*;
      import mx.events.*;
      import mx.managers.*;

      import twaver.*;
      import twaver.network.Network;
      import twaver.network.interaction.*;
      private function init():void{
        var box:ElementBox = network.elementBox;
        box.setStyle(Styles.BACKGROUND_TYPE, Consts.BACKGROUND_TYPE_VECTOR);
        box.setStyle(Styles.BACKGROUND_VECTOR_FILL_COLOR, 0x00ff00);
        box.setStyle(Styles.BACKGROUND_VECTOR_GRADIENT_COLOR, 0xffffffff);
        box.setStyle(Styles.BACKGROUND_VECTOR_GRADIENT, Consts.GRADIENT_LINEAR_SOUTHWEST);

        var from:Node = new Node();
        from.name = "from";
        from.location = new Point(20, 20);
```

```

        box.add(from);
        var to:Node = new Node();
        to.name = "to";
        to.location = new Point(150, 60);
        box.add(to);
        var link:Link = new Link(from,to);
        link.name = "hello TWaver!";
        box.add(link);
        network.setEditInteractionHandlers(true);

        var format:String="twaver";
        network.addEventListener(DragEvent.DRAG_ENTER,function(evt:DragEvent ):void{
            if( evt.dragSource.hasFormat( format ) )
            {
                DragManager.acceptDragDrop(network);
            }
        });
        network.addEventListener(DragEvent.DRAG_DROP,function ( evt:DragEvent ):void{
            if(evt.dragInitiator==createNodeButton){
                var centerLocation:Point=network.getLogicalPoint(evt as MouseEvent);
                var node:Node=new Node();
                node.centerLocation=centerLocation;
                var parentNode:ISubNetwork=network.currentSubNetwork;
                node.parent=parentNode;
                network.elementBox.add(node);
            }
        });
        createNodeButton.addEventListener(MouseEvent.CLICK,function (evt:MouseEvent ):void
        {
            var dragSource:DragSource = new DragSource();
            dragSource.addData( createNodeButton, format );
            DragManager.doDrag( createNodeButton, dragSource, evt );
        });
    }

    protected function button1_clickHandler(event:MouseEvent):void
    {
        network.setDefaultInteractionHandlers();
    }

    protected function button2_clickHandler(event:MouseEvent):void
    {
        network.setEditInteractionHandlers();
    }

    protected function button3_clickHandler(event:MouseEvent):void
    {
        network.interactionHandlers = new Collection([
            new SelectInteractionHandler(network),
            new EditInteractionHandler(network),
            new MoveInteractionHandler(network),
            new DefaultInteractionHandler(network),
            new MapFilterInteractionHandler(network),
        ]);
    }

    protected function button4_clickHandler(event:MouseEvent):void
    {
        network.interactionHandlers = new Collection([
            new SelectInteractionHandler(network),
            new EditInteractionHandler(network),
            new MoveInteractionHandler(network),
            new DefaultInteractionHandler(network),
            new MapFilterInteractionHandler(network, Consts.MAP_FILTER_MAGNIFY),
        ]);
    }
}

```



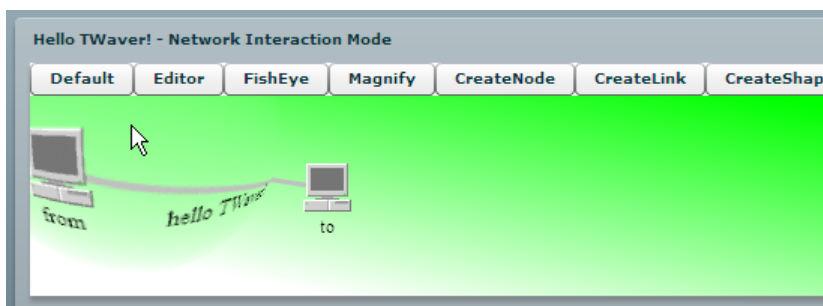
```

protected function button5_clickHandler(event:MouseEvent):void
{
    network.setCreateLinkInteractionHandlers();
}

protected function button6_clickHandler(event:MouseEvent):void
{
    network.setCreateShapeLinkInteractionHandlers(ShapeLink);
}

protected function button7_clickHandler(event:MouseEvent):void
{
    network.setCreateShapeNodeInteractionHandlers();
}
]]>
</mx:Script>
<mx:Panel title="Hello TWaver! - Network Interaction Mode" width="100%" height="100%" verticalGap="0">
<mx:HBox width="100%" horizontalGap="0" >
<mx:Button label="Default" click="button1_clickHandler(event)"/>
<mx:Button label="Editor" click="button2_clickHandler(event)"/>
<mx:Button label="FishEye" click="button3_clickHandler(event)"/>
<mx:Button label="Magnify" click="button4_clickHandler(event)"/>
<mx:Button label="CreateNode" id="createNodeButton"/>
<mx:Button label="CreateLink" click="button5_clickHandler(event)"/>
<mx:Button label="CreateShapeLink" click="button6_clickHandler(event)"/>
<mx:Button label="CreateShapeNode" click="button7_clickHandler(event)"/>
</mx:HBox>
<tw:Network id="network" width="100%" height="100%">
</tw:Network>
</mx:Panel>
</mx:Application>

```



Appendix

- [Network Element Stylesheet](#)

Network Element Stylesheet

All view properties of network element in TWaver Flex is realized by set style property. This chapter will list all properties of all kinds of network element including assignment range and instructions.

```
varlink:Link =newLink(from,to); link.name = "hello TWaver!";
//style : Styles.LINK_TYPE
//property : Consts.LINK_TYPE_FLEXIONAL
//link type is LINK_TYPE_FLEXIONAL
link.setStyle(Styles.LINK_TYPE,Consts.LINK_TYPE_FLEXIONAL);
```

Alarm Bubble

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|--|--|
| ALARM_ALPHA | number, default value is 1 0-1 | transparency |
| ALARM_BOLD | boolean, default value is false | whether use BOLD font for alarm bubble text |
| ALARM_COLOR | number | font color |
| ALARM_CONTENT_XSCALE | number, default value is 1 | text zoon in x-axis Note: use it when words embed |
| ALARM_CONTENT_YSCALE | number, default value is 1 | text zoon in y-axis Note: use it when words embed |
| ALARM_CORNER_RADIUS | number, default value is 8 | radius of alarm bubble |
| ALARM_DIRECTION | string, default value is aboveright Consts.ATTACHMENT_DIRECTION_* | the direction of alarm bubble, default value is right-top |
| ALARM_EMBED | boolean, default value is false | whether embed font |
| ALARM_FILL_ALPHA | number, default value is 1 0-1 | fill color transparency |
| ALARM_FONT | string | font |
| ALARM_GRADIENT | string, default value is radial.south Consts.GRADIENT_* | fill gradient type |
| ALARM_GRADIENT_ALPHA | number, default value is 1 0-1 | fill gradient color transparency |
| ALARM_GRADIENT_COLOR | number, default value is white | fill gradient color |
| ALARM_ITALIC | boolean, default value is false | font italic |
| ALARM_OUTLINE_ALPHA | number, default value is 1 0-1 | boarder color transparency |
| ALARM_OUTLINE_COLOR | number, default value is 0 | boarder color |

| | | |
|-----------------------|---|---|
| ALARM_OUTLINE_WIDTH | number, default value is -1 | boarder width, blank when value is smaller than 0 |
| ALARM_PADDING | number, default value is 0 | padding amount |
| ALARM_PADDING_BOTTOM | number, default value is 0 | padding bottom amount |
| ALARM_PADDING_LEFT | number, default value is 0 | padding left amount |
| ALARM_PADDING_RIGHT | number, default value is 0 | padding right amount |
| ALARM_PADDING_TOP | number, default value is 0 | padding top amount |
| ALARM_POINTER_LENGTH | number, default value is 10 | pointer length |
| ALARM_POINTER_WIDTH | number, default value is 8 | pointer width |
| ALARM_POSITION | string, default value is hotspot Consts.POSITION_* | alarm bubble position |
| ALARM_SHADOW_ALPHA | number, default value is 0.4 0-1 | shadow transparency alarm bubble |
| ALARM_SHADOW_ANGLE | number, default value is 45 | alarm shadow angle |
| ALARM_SHADOW_COLOR | number, default value is black | color of alarm shadow |
| ALARM_SHADOW_DISTANCE | number, default value is 4 | alarm shadow distance |
| ALARM_SIZE | number, default value is 12 | font size |
| ALARM_UNDERLINE | boolean, default value is false | text underline |
| ALARM_XOFFSET | number, default value is 0 | offset in x-axis |
| ALARM_YOFFSET | number, default value is 0 | offset in y-axis |

ARROW ([Link](#), [ShapeNode](#))

| | | |
|----------------------------|-----------------------------------|---|
| ARROW_FROM | boolean, default value is false | whether draw arrow for start point |
| ARROW_FROM_ALPHA | number, default value is 1 0-1 | the transparency of start point arrow |
| ARROW_FROM_COLOR | int, default value is 0 | the color of start point arrow |
| ARROW_FROM_FILL | boolean, default value is true | whether filling the start point arrow |
| ARROW_FROM_HEIGHT | number, default value is 9 | filling color |
| ARROW_FROM_OUTLINE_ALPHA | number, default value is 1 0-1 | fill gradient color |
| ARROW_FROM_OUTLINE_COLOR | int, default value is 0 | boarder color |
| ARROW_FROM_OUTLINE_PATTERN | array.number | parameter [X, Y] of arrow boarder dashed line |
| ARROW_FROM_OUTLINE_WIDTH | number, default value is 0 | boarder width |

| | | |
|--------------------------|---|---|
| ARROW_FROM_SHAPE | string, default value is arrow.standard Consts.ARROW_* | arrow shape |
| ARROW_FROM_WIDTH | number, default value is 12 | arrow width |
| ARROW_FROM_XOFFSET | number, default value is 0 | offset in connect direction |
| ARROW_FROM_YOFFSET | number, default value is 0 | offset in vertical direction |
| ARROW_TO | boolean, default value is false | whether draw arrow for end point |
| ARROW_TO_ALPHA | number, default value is 1 0-1 | transparency of end point arrow |
| ARROW_TO_COLOR | int, default value is 0 | color of end point arrow |
| ARROW_TO_FILL | boolean, default value is true | filling of end point arrow |
| ARROW_TO_HEIGHT | number, default value is 9 | height of end point arrow |
| ARROW_TO_OUTLINE_ALPHA | number, default value is 1 0-1 | boarder transparency of end point arrow |
| ARROW_TO_OUTLINE_COLOR | int, default value is 0 | boarder color of end point arrow |
| ARROW_TO_OUTLINE_PATTERN | array.number | parameter [X, Y] of arrow boarder dashed line |
| ARROW_TO_OUTLINE_WIDTH | number, default value is 0 | boarder width of end point arrow |
| ARROW_TO_SHAPE | string, default value is arrow.standard Consts.ARROW_* | shape of end point arrow |
| ARROW_TO_WIDTH | number, default value is 12 | width of end point arrow |
| ARROW_TO_XOFFSET | number, default value is 0 | offset of end point arrow in connect direction |
| ARROW_TO_YOFFSET | number, default value is 0 | offset of end point arrow in vertical direction |

BACKGROUND (SubNetwork, ElementBox)

| Style (twaver.Styles#) | Evaluation | Note |
|-----------------------------|---|---|
| BACKGROUND_IMAGE | string | background picture |
| BACKGROUND_IMAGE_CAPS_STYLE | string, default value is round Consts.CAPS_STYLE_* | endpoint style of background boarder Reference: flash.display.Graphics# lineStyle(thickness:Number = NaN, color:uint = 0, alpha:Number = 1.0, pixelHinting:Boolean = false, scaleMode:String = "normal", caps:String = null, joints:String = null, miterLimit:Number = 3) |

| | | |
|---------------------------------|--|---|
| BACKGROUND_IMAGE_JOINT_STYLE | string, default value is round Consts.JOINT_STYLE_* | turing style of background boarder Reference: flash.display.Graphics#lineStyle(thickness:Number = NaN, color:uint = 0, alpha:Number = 1.0, pixelHinting:Boolean = false, scaleMode:String = "normal", caps:String = null, joints:String = null, miterLimit:Number = 3) |
| BACKGROUND_IMAGE_OUTLINE_ALPHA | number, default value is 1 0-1 | transparency of background picture |
| BACKGROUND_IMAGE_OUTLINE_COLOR | number, default value is 0 | boarder color of background picture |
| BACKGROUND_IMAGE_OUTLINE_WIDTH | number, default value is -1 | boarder width of background picture blank when value is smaller than 0 |
| BACKGROUND_IMAGE_PADDING | number, default value is 0 | padding amount |
| BACKGROUND_IMAGE_PADDING_BOTTOM | number, default value is 0 | bottom padding amount |
| BACKGROUND_IMAGE_PADDING_LEFT | number, default value is 0 | left padding amount |
| BACKGROUND_IMAGE_PADDING_RIGHT | number, default value is 0 | right padding amount |
| BACKGROUND_IMAGE_PADDING_TOP | number, default value is 0 | top padding amount |
| BACKGROUND_IMAGE_PIXEL_HINTING | boolean, default value is true | whether use pixel fine-tuning for background picture boarder Reference: flash.display.Graphics#lineStyle(thickness:Number = NaN, color:uint = 0, alpha:Number = 1.0, pixelHinting:Boolean = false, scaleMode:String = "normal", caps:String = null, joints:String = null, miterLimit:Number = 3) |
| BACKGROUND_IMAGE_SCALE_MODE | string, default value is normal | zoom style of boarder line Reference: flash.display.Graphics#lineStyle(thickness:Number = NaN, color:uint = 0, alpha:Number = 1.0, pixelHinting:Boolean = false, scaleMode:String = "normal", caps:String = null, joints:String = null, miterLimit:Number = 3) |
| BACKGROUND_IMAGE_SCOPE | string, default value is viewsize Consts.SCOPE_* | scope of background picture Consts.SCOPE_ROOTCANVAS - rootCanvas scope Consts.SCOPE_VIEWPORT - viewPort scope Consts.SCOPE_VIEWSIZE - viewSize scope |

| | | |
|----------------------------------|---|---|
| BACKGROUND_IMAGE_SHAPE | string, default value is rectangle Consts.SHAPE_* | the shape of background |
| BACKGROUND_IMAGE_STRETCH | string, default value is north.west Consts.STRETCH_* | alignment of background |
| BACKGROUND_TYPE | string, default value is none Consts.BACKGROUND_TYPE_* | background type |
| BACKGROUND_VECTOR_CAPS_STYLE | string, default value is round Consts.CAPS_STYLE_* | endpoint style of background boarder Reference: flash.display.Graphics#lineStyle(thickness:Number = NaN, color:uint = 0, alpha:Number = 1.0, pixelHinting:Boolean = false, scaleMode:String = "normal", caps:String = null, joints:String = null, miterLimit:Number = 3) |
| BACKGROUND_VECTOR_DEEP | number, default value is 0 | depth of background |
| BACKGROUND_VECTOR_FILL | boolean, default value is true | filling background picture |
| BACKGROUND_VECTOR_FILL_ALPHA | number, default value is 1 0-1 | fill gradient color for background |
| BACKGROUND_VECTOR_FILL_COLOR | number, default value is 0xCCCCFF | fill color |
| BACKGROUND_VECTOR_GRADIENT | string, default value is none Consts.GRADIENT_* | gradient type |
| BACKGROUND_VECTOR_GRADIENT_ALPHA | number, default value is 1 0-1 | transparency of gradient color |
| BACKGROUND_VECTOR_GRADIENT_COLOR | number, default value is white | gradient color |
| BACKGROUND_VECTOR_JOINT_STYLE | string, default value is round | joint point style Reference: flash.display.Graphics#lineStyle(thickness:Number = NaN, color:uint = 0, alpha:Number = 1.0, pixelHinting:Boolean = false, scaleMode:String = "normal", caps:String = null, joints:String = null, miterLimit:Number = 3) |
| BACKGROUND_VECTOR_OUTLINE_ALPHA | number, default value is 1 0-1 | transparency of background border |
| BACKGROUND_VECTOR_OUTLINE_COLOR | number, default value is 0x5B5B5B | border color |
| BACKGROUND_VECTOR_OUTLINE_WIDTH | number, default value is 0 | border width |
| BACKGROUND_VECTOR_PADDING | number, default value is 0 | padding amount |
| BACKGROUND_VECTOR_PADDING_BOTTOM | number, default value is 0 | bottom padding amount |
| BACKGROUND_VECTOR_PADDING_LEFT | number, default value is 0 | left padding amount |
| BACKGROUND_VECTOR_PADDING_RIGHT | number, default value is 0 | right padding amount |

| | | |
|---------------------------------|--|--|
| BACKGROUND_VECTOR_PADDING_TOP | number, default value is 0 | top padding amount |
| BACKGROUND_VECTOR_PIXEL_HINTING | Boolean, default value is true | whether use pixel fine-tuning Reference: flash.display.Graphics#lineStyle(thickness:Number = NaN, color:uint = 0, alpha:Number = 1.0, pixelHinting:Boolean = false, scaleMode:String = "normal", caps:String = null, joints:String = null, miterLimit:Number = 3) |
| BACKGROUND_VECTOR_SCALE_MODE | String, default value is normal | zoom style of border line Reference: flash.display.Graphics#lineStyle(thickness:Number = NaN, color:uint = 0, alpha:Number = 1.0, pixelHinting:Boolean = false, scaleMode:String = "normal", caps:String = null, joints:String = null, miterLimit:Number = 3) |
| BACKGROUND_VECTOR_SCOPE | string, default value is viewsize | background scope Consts.SCOPE_ROOTCANVAS - rootCanvas scope Consts.SCOPE_VIEWPORT - viewport scope Consts.SCOPE_VIEWSIZE - viewSize scope |
| BACKGROUND_VECTOR_SHAPE | string, default value is rectangle Consts.SHAPE_* | background shape |

BODY

| Style (twaver.Styles#) | Evaluation | Note |
|----------------------------|--|--------------------------------------|
| BODY_ALPHA | number, default value is 1 0-1 | tranparency of network element |
| BODY_BEVEL | boolean, default value is false | whether use body bevel |
| BODY_BEVEL_ANGLE | number, default value is 45 | angle of body bevel |
| BODY_BEVEL_BLURX | number, default value is 4.0 | index of blurry in x-axis direction |
| BODY_BEVEL_BLURY | number, default value is 4.0 | index of blurry in y-axis direction |
| BODY_BEVEL_DISTANCE | number, default value is 3 | body bevel distance |
| BODY_BEVEL_HIGHLIGHT_ALPHA | number, default value is 1 | transparency of body bevel highlight |
| BODY_BEVEL_HIGHLIGHT_COLOR | number, default value is white | color of body bevel highlight |
| BODY_BEVEL_KNOCKOUT | Boolean, defalut value is false | wether deduct body bevel |
| BODY_BEVEL_QUALITY | int, default value is Consts.QUALITY_HIGH | quality of body bevel |

| | | |
|-------------------------|--|-----------------------------------|
| BODY_BEVEL_SHADOW_ALPHA | number, default value is 1 | transparency of body bevel shadow |
| BODY_BEVEL_SHADOW_COLOR | number, default value is black | color of body bevel shadow |
| BODY_BEVEL_STRENGTH | number, default value is 1 | body bevel strength |
| BODY_BEVEL_TYPE | string, default value is Consts.BEVEL_TYPE_INNER | body bevel type |
| BODY_INTERACTIVE | Boolean, default value is false | body interactive |

BUS

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|--|---------------------------|
| BUS_STYLE | string, default value is nearby Consts.BUS_STYLE_* | BUS network element style |

CONTENT

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|--|------------------------------|
| CONTENT_TYPE | string, default value is default Consts.CONTENT_TYPE_* | network element content type |

FOLLOWER

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|----------------------------|---------------------------|
| FOLLOWER_COLUMN_INDEX | number, default value is 0 | column index for follower |
| FOLLOWER_COLUMN_SPAN | number, default value is 1 | column span for follower |
| FOLLOWER_PADDING | number, default value is 0 | padding amount |
| FOLLOWER_PADDING_BOTTOM | number, default value is 0 | bottom padding amount |
| FOLLOWER_PADDING_LEFT | number, default value is 0 | left padding amount |
| FOLLOWER_PADDING_RIGHT | number, default value is 0 | right padding amount |
| FOLLOWER_PADDING_TOP | number, default value is 0 | top padding amount |
| FOLLOWER_ROW_INDEX | number, default value is 0 | row index for follower |
| FOLLOWER_ROW_SPAN | number, default value is 1 | row span for follower |

GRID

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|----------------------------|--------------|
| GRID_BORDER | number, default value is 1 | margin width |

| | | |
|----------------------|-----------------------------------|---|
| GRID_BORDER_BOTTOM | number, default value is 0 | bottom margin |
| GRID_BORDER_LEFT | number, default value is 0 | left margin |
| GRID_BORDER_RIGHT | number, default value is 0 | right margin |
| GRID_BORDER_TOP | number, default value is 0 | top margin |
| GRID_CELL_DEEP | number, default value is -1 | depth of cell |
| GRID_COLUMN_COUNT | number, default value is 1 | column count |
| GRID_COLUMN_PERCENTS | array.number | the percent of each column, null means average [0.2,0.8] stands for the two columns divided in 20%,80% |
| GRID_DEEP | number, default value is 1 | depth of grid |
| GRID_FILL | boolean, default value is true | filling grid |
| GRID_FILL_ALPHA | number, default value is 1 0-1 | fill transparency |
| GRID_FILL_COLOR | number, default value is gray | fill color |
| GRID_PADDING | number, default value is 1 | padding amount |
| GRID_PADDING_BOTTOM | number, default value is 0 | bottom padding amount |
| GRID_PADDING_LEFT | number, default value is 0 | left padding amount |
| GRID_PADDING_RIGHT | number, default value is 0 | right padding amount |
| GRID_PADDING_TOP | number, default value is 0 | top padding amount |
| GRID_ROW_COUNT | number, default value is 1 | row count |
| GRID_ROW_PERCENTS | array.number | the percent of each row, null means average [0.2,0.8]stands for the two rows divided in 20%,80% |

GROUP

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|-----------------------------------|---|
| GROUP_CAPS_STYLE | string, default value is round | endpoint style of border line, refer to flash.display.Graphics#lineStyle(...) parameters. |
| GROUP_DEEP | number, default value is 0 | depth of group |
| GROUP_FILL | boolean, default value is true | filling group |
| GROUP_FILL_ALPHA | number, default value is 1 0-1 | fill transparency |
| GROUP_FILL_COLOR | number, default value is 0xCCCCFF | fill color |

| | | |
|----------------------|--|---|
| GROUP_GRADIENT | string Const.GRADIENT_* | gradient type |
| GROUP_GRADIENT_ALPHA | number, default value is 1 0-1 | transparency of gradient color |
| GROUP_GRADIENT_COLOR | number, default value is white | gradient color |
| GROUP_JOINT_STYLE | string, default value is round | joint point style of border line refer to flash.display.Graphics# lineStyle(...) parameters |
| GROUP_OUTLINE_ALPHA | number, default value is 1 0-1 | border transparency |
| GROUP_OUTLINE_COLOR | number, default value is 0x5B5B5B | border color |
| GROUP_OUTLINE_WIDTH | number, default value is 1 | border width |
| GROUP_PADDING | number, default value is -3 | padding amount |
| GROUP_PADDING_BOTTOM | number, default value is 0 | bottom padding amount |
| GROUP_PADDING_LEFT | number, default value is 0 | left padding amount |
| GROUP_PADDING_RIGHT | number, default value is 0 | right padding amount |
| GROUP_PADDING_TOP | number, default value is 0 | top padding amount |
| GROUP_PIXEL_HINTING | boolean, default value is true | whether use pixel fine-tuning for border line refer to flash.display.Graphics# lineStyle(...) parameters |
| GROUP_SCALE_MODE | string, default value is normal | zoom style of border line, refer to flash.display.Graphics# lineStyle(...) parameters |
| GROUP_SHAPE | string, default value is rectangle Consts.SHAPE_* | group shape |

ICONS

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|--|----------------------------|
| ICONS_COLORS | array.number | attachment color |
| ICONS_NAMES | array.string | attachment name |
| ICONS_ORIENTATION | string, default value is right Consts.ORIENTATION_* | alignment of attachment |
| ICONS_POSITION | string, default value is topleft Consts.POSITION_* | attachemtn position |
| ICONS_XGAP | number, default value is 1 | gap in x-axis direction |
| ICONS_XOFFSET | number, default value is 0 | offset in X-axis direction |
| ICONS_YGAP | number, default value is 1 | gap in y-axis direction |

| | | |
|---------------|----------------------------|----------------------------|
| ICONS_YOFFSET | number, default value is 0 | offset in y-axis direction |
|---------------|----------------------------|----------------------------|

IMAGE

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|--|---|
| IMAGE_CAPS_STYLE | string, default value is round | endpoint style of border line |
| IMAGE_JOINT_STYLE | string, default value is round | joint point style of border line |
| IMAGE_OUTLINE_ALPHA | number, default value is 1 0-1 | transparency of border |
| IMAGE_OUTLINE_COLOR | number, default value is 0 | border color |
| IMAGE_OUTLINE_WIDTH | number, default value is 0 | border width |
| IMAGE_PADDING | number, default value is 0 | padding amount |
| IMAGE_PADDING_BOTTOM | number, default value is 0 | bottom padding amount |
| IMAGE_PADDING_LEFT | number, default value is 0 | left padding amount |
| IMAGE_PADDING_RIGHT | number, default value is 0 | right padding amount |
| IMAGE_PADDING_TOP | number, default value is 0 | top padding amount |
| IMAGE_PIXEL_HINTING | boolean, default value is true | whether use pixel fine-tuning for border line |
| IMAGE_SCALE_MODE | string, default value is normal | zoom style for border line |
| IMAGE_SHAPE | string, default value is rectangle Consts.SHAPE_* | image shape |
| IMAGE_STRETCH | string, default value is uniform Consts.STRETCH_* | stretch style of image default is no stretch |

INNER

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|--|---|
| INNER_ALPHA | number, default value is 1 0-1 | transparency |
| INNER_BACK | boolean, default value is true | whether show in back of network element |
| INNER_COLOR | number | inner picture color |
| INNER_GRADIENT | string, default value is none Consts.GRADIENT_* | gradient type |
| INNER_GRADIENT_ALPHA | number, default value is 1 0-1 | gradient transparency |
| INNER_GRADIENT_COLOR | number, default value is white | gradient color |

| | | |
|----------------------|--|-----------------------|
| INNER_OUTLINE_ALPHA | number, default value is 1 0-1 | border transparency |
| INNER_OUTLINE_COLOR | number, default value is gray | border color |
| INNER_OUTLINE_WIDTH | number, default value is 1 | border width |
| INNER_PADDING | number, default value is -2 | padding amount |
| INNER_PADDING_BOTTOM | number, default value is 0 | bottom padding amount |
| INNER_PADDING_LEFT | number, default value is 0 | left padding amount |
| INNER_PADDING_RIGHT | number, default value is 0 | right padding amount |
| INNER_PADDING_TOP | number, default value is 0 | top padding amount |
| INNER_SHAPE | string, default value is rectangle Consts.SHAPE_* | inner picture shape |
| INNER_STYLE | string, default value is dye Consts.INNER_STYLE_* | inner picture style |

Inner Shape example:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
    applicationComplete="init()">
<mx:Script>
<![CDATA[
import twaver.*;
import twaver.network.Network;

private function init():void{
    var box:ElementBox = network.elementBox;

    var node:Node = newNode();
    node.setLocation(50,50);
    node.setStyle(Styles.INNER_STYLE, Consts.INNER_STYLE_SHAPE);
    node.setStyle(Styles.INNER_SHAPE, Consts.SHAPE_CIRCLE);
    node.setStyle(Styles.INNER_GRADIENT, Consts.GRADIENT_RADIAL_CENTER);
    node.setStyle(Styles.INNER_GRADIENT_ALPHA, 0.5);
    node.setStyle(Styles.INNER_COLOR, 0xff0000);
    node.setStyle(Styles.INNER_ALPHA, 0.8);
    node.setStyle(Styles.INNER_PADDING, -6);
    node.setStyle(Styles.INNER_BACK, false);
    box.add(node);

    node = newNode();
    node.setLocation(150,50);
    node.setStyle(Styles.INNER_STYLE, Consts.INNER_STYLE_SHAPE);
    node.setStyle(Styles.INNER_SHAPE, Consts.SHAPE_DIAMOND);
    node.setStyle(Styles.INNER_COLOR, 0x00ff00);
    node.setStyle(Styles.INNER_GRADIENT, Consts.GRADIENT_RADIAL_SOUTHWEST);
    node.setStyle(Styles.INNER_PADDING_TOP, 10);
    node.setStyle(Styles.INNER_PADDING_BOTTOM, -5);
    node.setStyle(Styles.INNER_PADDING_LEFT, -20);
    node.setStyle(Styles.INNER_PADDING_RIGHT, -20);
    box.add(node);
}
]]>
```

```

</mx:Script>
<mx:Panel title="Hello TWaver! - Inner Shape"width="100%"height="100%">
  <twaver:Network id="network" width="100%"height="100%"/>
</mx:Panel>
</mx:Application>

```

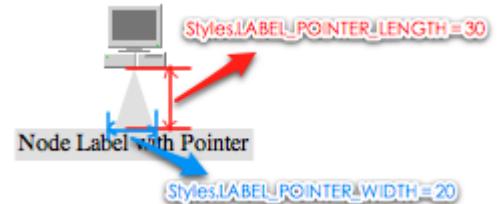
Results:



LABEL

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|---|--|
| LABEL_ALPHA | number, default value is 1 0-1 | tranparency |
| LABEL_BOLD | boolean, default value is false | whether use BOLD font |
| LABEL_COLOR | number | font color |
| LABEL_CONTENT_XSCALE | number, default value is 1 | zoom text in x-axis direction Note: use it when words embed |
| LABEL_CONTENT_YSCALE | number, default value is 1 | zoom text in y-axis direction Note: use it when words embed |
| LABEL_CORNER_RADIUS | number, default value is 0 | radius of corner |
| LABEL_DIRECTION | string, default value is below Consts.ATTACHMENT_DIRECTION_* | label direction for network element |
| LABEL_EMBED | boolean, default value is false | whether embed font |
| LABEL_FILL | boolean, default value is false | whether filling |
| LABEL_FILL_ALPHA | number, default value is 0.5 0-1 | fill color transparency |
| LABEL_FILL_COLOR | number, default value is gray | fill color |
| LABEL_FONT | string | font |
| LABEL_GRADIENT | string Consts.GRADIENT_* | fill gradient type |
| LABEL_GRADIENT_ALPHA | number, default value is 0.5 0-1 | tranparency |
| LABEL_GRADIENT_COLOR | number, default value is white | gradient color |
| LABEL_ITALIC | boolean, default value is false | font italic |
| LABEL_OUTLINE_ALPHA | number, default value is 1 0-1 | border transparency |

| | | |
|----------------------|---|---|
| LABEL_OUTLINE_COLOR | number, default value is 0 | border color |
| LABEL_OUTLINE_WIDTH | number, default value is -1 | border color, blank when value is smaller than 0 |
| LABEL_PADDING | number, default value is 0 | padding amount |
| LABEL_PADDING_BOTTOM | number, default value is 0 | bottom padding amount |
| LABEL_PADDING_LEFT | number, default value is 0 | left padding amount |
| LABEL_PADDING_RIGHT | number, default value is 0 | right padding amount |
| LABEL_PADDING_TOP | number, default value is 0 | top padding amount |
| LABEL_POINTER_LENGTH | number, default value is 0 | length of pointer, default value is 0 Note: use it when fill label var node:Node = new Node(); node.name = "Node Label with Pointer"; node.setStyle(Styles.LABEL_FILL,true); node.setStyle(Styles.LABEL_POINTER_LENGTH,30); node.setStyle(Styles.LABEL_POINTER_WIDTH,20); |
| LABEL_POINTER_WIDTH | number, default value is 8 | pointer width |
| LABEL_POSITION | string, default value is bottom.bottom Consts.POSITION_* | label position for network element |
| LABEL_SIZE | number, default value is 12 | font size |
| LABEL_UNDERLINE | boolean, default value is false | font underline |
| LABEL_XOFFSET | number, default value is 0 | offset in x-axis direction |
| LABEL_YOFFSET | number, default value is 0 | offset in y-axis direction |



LINK

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|-----------------------------------|--|
| LINK_ALPHA | number, default value is 1 0-1 | tranparency |
| LINK_BUNDLE_ENABLE | boolean, default value is true | whether join binding (reference: Links Binding) |
| LINK_BUNDLE_EXPANDED | boolean, default value is true | whether expand |
| LINK_BUNDLE_GAP | number, default value is 12 | boundle gap |

| | | |
|----------------------------|--|---|
| LINK_BUNDLE_ID | int, default value is 0 | the id of boundle group |
| LINK_BUNDLE_INDEPENDENT | boolean, default value is false | whether doundle this group independent |
| LINK_BUNDLE_OFFSET | number, default value is 20 | offset of bundle link |
| LINK_CAPS_STYLE | string, default value is round | endpoint type |
| LINK_COLOR | number, default value is gray | link color |
| LINK_CONTROL_POINT | point | control point of link, only for orthogonal connection |
| LINK_CORNER | string, default value is round Consts.LINK_CORNER_* | corner type |
| LINK_CORNER_XRADIUS | number, default value is 8 | radius of corner x |
| LINK_CORNER_YRADIUS | number, default value is 8 | radius of corner y |
| LINK_EXTEND | number, default value is 20 | offset of link application to links in LINK_TYPE_EXTEND_* and LINK_TYPE_FLEXIONAL* type |
| LINK_FROM_AT_EDGE | boolean, default value is true | whether start point position is at the edge |
| LINK_FROM_POSITION | string, default value is center Consts.POSITION_* | start point position |
| LINK_FROM_XOFFSET | number, default value is 0 | start point offset in x-axis |
| LINK_FROM_YOFFSET | number, default value is 0 | start point offset in y-axis |
| LINK_HANDLER_BOLD | boolean, default value is false | whether set handler text font to be BOLD when link binding |
| LINK_HANDLER_COLOR | number | <p>handler font color</p> <pre> var node:Node=new Node(); node.setLocation(50,50); box.add(node); var node2=new Node(); node2.setLocation(150,100); box.add(node2); var link:Link=new Link(node,node2); link.setStyle(Styles.LINK_HANDLER_COLOR,0xff0000); box.add(link); link=new Link(node,node2); link.setStyle(Styles.LINK_HANDLER_COLOR,0xff0000); box.add(link); </pre>  |
| LINK_HANDLER_CORNER_RADIUS | number, default value is 0 | radius of handler corner |

| | | |
|-----------------------------|--|--|
| LINK_HANDLER_DIRECTION | string, default value is below Consts.ATTACHMENT_DIRECTION_* | handler direction |
| LINK_HANDLER_EMBED | boolean, default value is false | handler embed font |
| LINK_HANDLER_FILL | boolean, default value is false | filling handler |
| LINK_HANDLER_FILL_ALPHA | number, default value is 1 0-1 | fill transparency for handler |
| LINK_HANDLER_FILL_COLOR | number, default value is gray | fill color for handler |
| LINK_HANDLER_FONT | string | handler font |
| LINK_HANDLER_GRADIENT | string Consts.GRADIENT_* | handler gradient type |
| LINK_HANDLER_GRADIENT_ALPHA | number, default value is 1 0-1 | handler gradient transparency |
| LINK_HANDLER_GRADIENT_COLOR | number, default value is white | handler gradient color |
| LINK_HANDLER_ITALIC | boolean, default value is false | handler font italic |
| LINK_HANDLER_POINTER_LENGTH | number, default value is 0 | length of handler pointer |
| LINK_HANDLER_POINTER_WIDTH | number, default value is 8 | width of handler pointer |
| LINK_HANDLER_POSITION | string, default value is topleft.topleft Consts.POSITION_* | handler position |
| LINK_HANDLER_SIZE | number, default value is 12 | handler font size |
| LINK_HANDLER_UNDERLINE | boolean, default value is false | handler text underline |
| LINK_HANDLER_XOFFSET | number, default value is 0 | offset in x-axis direction |
| LINK_HANDLER_YOFFSET | number, default value is 0 | offset in y-axis direction |
| LINK_JOINT_STYLE | string, default value is round | joint style of link Refer to flash.display.Graphics# lineStyle(...) parameters |
| LINK_LOOPED_DIRECTION | string, default value is northwest | self-loop direction of link |
| LINK_LOOPED_GAP | number, default value is 6 | the gap of link self-loop |
| LINK_LOOPED_TYPE | string, default value is arc | link self-loop type |
| LINK_PATTERN | array.number | dashed line parameters[X,Y] of link |
| LINK_PIXEL_HINTING | boolean, default value is true | whether use pixel offset for link Refer to flash.display.Graphics# lineStyle(...) parameters |
| LINK_SCALE_MODE | string, default value is normal | zoom style of link refer to flash.display.Graphics# lineStyle(...) parameters |
| LINK_SPLIT_BY_PERCENT | boolean, default value is false | split link by percent |
| LINK_SPLIT_PERCENT | number, default value is 0.5 | the percent of split link |



| | | |
|------------------|---------------------------------|-------------------------------------|
| LINK_SPLIT_VALUE | number, default value is 20 | split amount of link |
| LINK_TO_AT_EDGE | boolean, default value is true | whether the endpoint is at the edge |
| LINK_TO_POSITION | string, default value is center | position of endpoint |
| LINK_TO_XOFFSET | number, default value is 0 | endpoint offset in x-axis direction |
| LINK_TO_YOFFSET | number, default value is 0 | endpoint offset in y-axis direction |
| LINK_TYPE | string, default value is arc | link type |
| LINK_WIDTH | number, default value is 3 | link width |

NETWORK

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|------------|---|
| NETWORK_LABEL | string | the label of network element in Network, display network element name if it is null |

OUTER

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|-----------------------------------|---|
| OUTER_ALPHA | number, default value is 1 0-1 | transparency of border |
| OUTER_ANGLE | number, default value is 45 | the corner for shadow and light only works when border style is one of these two: Consts.OUTER_STYLE_SHADOW Consts.OUTER_STYLE_GLOW |
| OUTER_BLURX | number, default value is 6 | index of blurry in x-axis direction |
| OUTER_BLURY | number, default value is 6 | index of blurry in y-axis direction |
| OUTER_CAPS_STYLE | string, default value is round | endpoint type of link refer to flash.display.Graphics#lineStyle(...) parameters |
| OUTER_COLOR | number | color |
| OUTER_DISTANCE | number, default value is 4 | distance of shadow and light |
| OUTER_HIDEOBJECT | boolean, default value is false | whether hide the object only works when the type is Consts.OUTER_STYLE_SHADOW refer to flash.filters.DropShadowFilter constructor |
| OUTER_INNER | boolean, default value is false | whether the filter result is inward, default is outward |

| | | |
|----------------------|------------------------------------|---|
| | | <p>Below are inward result and outward result example:</p> <pre>var node:Node=new Node(); node.name="shadow inner"; node.setStyle(Styles.OUTER_COLOR,0xff0000); node.setStyle(Styles.OUTER_STYLE,Consts.OUTER_STYLE_SHADOW); node.setStyle(Styles.OUTER_INNER,true); node.setLocation(50,50); box.add(node); var node2=new Node(); node2.name="shadow outer"; node2.setLocation(150,50); node2.setStyle(Styles.OUTER_COLOR,0xff0000); node2.setStyle(Styles.OUTER_STYLE,Consts.OUTER_STYLE_GLOW); box.add(node2);</pre> <p>Results:</p> <div style="display: flex; justify-content: space-around; align-items: center;">   </div> |
| OUTER_JOINT_STYLE | string, default value is round | joint point style of link refer to flash.display.Graphics# lineStyle(...) parameters |
| OUTER_KNOCKOUT | boolean, default value is false | whether knockout the network element Only works when border is the type in these below two: Consts.OUTER_STYLE_SHADOW Consts.OUTER_STYLE_GLOW Refer to flash.filters.DropShadowFilter/ GlowFilter constructor |
| OUTER_PADDING | number, default value is -1 | padding amount, only works when the type is Consts.OUTER_STYLE_BORDER |
| OUTER_PADDING_BOTTOM | number, default value is 0 | bottom padding amount |
| OUTER_PADDING_LEFT | number, default value is 0 | left padding amount |
| OUTER_PADDING_RIGHT | number, default value is 0 | right padding amount |
| OUTER_PADDING_TOP | number, default value is 0 | top padding amount |
| OUTER_PIXEL_HINTING | boolean, default value is true | whether use pixel offset for line Refer to flash.display.Graphics# lineStyle(...) parameters |
| OUTER_QUALITY | number, default value is 3 0-10 | the quality of filter render |
| OUTER_SCALE_MODE | string, default value is normal | line zoom type Refer to flash.display.Graphics# lineStyle(...) parameters |
| OUTER_SHAPE | string, default value is rectangle | border shape |
| OUTER_STRENGTH | number, default value is 2 0-10 | filter strength refer to flash.display.Graphics# |

| | | |
|-------------|---|---------------------------|
| | | lineStyle(...) parameters |
| OUTER_STYLE | string, default value is border Consts.OUTER_STYLE_* | border style |
| OUTER_WIDTH | number, default value is 2 | border width |

SELECT

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|-------------------------------------|---|
| SELECT_ALPHA | number, default value is 0.7 0-1 | select color transparency |
| SELECT_ANGLE | number, default value is 45 | angle of shadow and light Only works when selection type is in one of these two: Consts.SELECT_STYLE_SHADOW Consts.SELECT_STYLE_GLOW |
| SELECT_BLURX | number, default value is 4 | index of blurry in x-axis direction |
| SELECT_BLURY | number, default value is 4 | index of blurry in y-axis direction |
| SELECT_CAPS_STYLE | string, default value is round | endpoint type of link refer to flash.display.Graphics# lineStyle(...) parameters |
| SELECT_COLOR | number, default value is 0x5B5B5B | selection color |
| SELECT_DISTANCE | number, default value is 4 | distance of shadow or light |
| SELECT_HIDEOBJECT | boolean, default value is false | whether hide the object Only works when the type is Consts.SELECT_STYLE_SHADOW Refer to flash.filters.DropShadowFilter constructor |
| SELECT_INNER | boolean, default value is false | whether filter result is inward, default value is outward |
| SELECT_JOINT_STYLE | string, default value is round | joint style of link refer to flash.display.Graphics# lineStyle(...) parameters |
| SELECT_KNOCKOUT | boolean, default value is false | whether knockout network element Only works when the type is one of them: Consts.SELECT_STYLE_SHADOW Consts.SELECT_STYLE_GLOW Refer to flash.filters.DropShadowFilter/ GlowFilter constructor |
| SELECT_PADDING | number, default value is -1 | padding amount , only works when the type is Consts.SELECT_STYLE_BORDER |
| SELECT_PADDING_BOTTOM | number, default value is 0 | bottom padding amount |

| | | |
|----------------------|--|--|
| SELECT_PADDING_LEFT | number, default value is 0 | left padding amount |
| SELECT_PADDING_RIGHT | number, default value is 0 | right padding amount |
| SELECT_PADDING_TOP | number, default value is 0 | top padding amount |
| SELECT_PIXEL_HINTING | boolean, default value is true | whether use pixel offset for link Refer to flash.display.Graphics#lineStyle(...) parameters |
| SELECT_QUALITY | number, default value is 3 | the quality of filter render |
| SELECT_SCALE_MODE | string, default value is normal | zoom type of link Refer to flash.display.Graphics#lineStyle(...) parameters |
| SELECT_SHAPE | string, default value is rectangle Consts.SHAPE_* | selection shape |
| SELECT_STRENGTH | number, default value is 1 | filter strength Refer to flash.display.Graphics#lineStyle(...) parameters |
| SELECT_STYLE | string, default value is shadow Consts.SELECT_STYLE_* | selection model |
| SELECT_WIDTH | number, default value is 2 | width of selection border |

SHAPELINK (twaver.ShapeLink)

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|--|----------------|
| SHAPELINK_TYPE | string, default value is straight Consts.SHAPELINK_TYPE_* | ShapeLink type |

SHAPENODE

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|---------------------------------|---|
| SHAPENODE_PATTERN | array.number | dashed line types of ShapeNode, stands for solid line when the value is null [X,Y] |
| SHAPENODE_CLOSED | boolean , default value is true | closed or not |

TREE

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|-----------------------------------|---|
| TREE_ALARM_FILL_ALPHA | number, default value is 1 0-1 | the transparency of alarm bubble for tree node |
| TREE_ALARM_FILL_COLOR | number | filling color of alarm bubble |

| | | |
|---------------------------|---|---|
| TREE_ALARM_GRADIENT | string, default value is radial.northeast Consts.GRADIENT_* | transparency type of alarm bubble |
| TREE_ALARM_GRADIENT_ALPHA | number, default value is 1 0-1 | gradient transparency |
| TREE_ALARM_GRADIENT_COLOR | number, default value is white | gradient color |
| TREE_ALARM_HEIGHT | number, default value is 8 | the height of alarm bubble |
| TREE_ALARM_OUTLINE_ALPHA | number, default value is 0.7 0-1 | the transparency of alarm bubble border |
| TREE_ALARM_OUTLINE_COLOR | number, default value is white | the border color of alarm bubble |
| TREE_ALARM_OUTLINE_WIDTH | number, default value is 0 | border width |
| TREE_ALARM_POSITION | string, default value is bottomleft.topright Consts.POSITION_* | alarm bubble position |
| TREE_ALARM_SHAPE | string, default value is circle Consts.SHAPE_* | alarm bubble shape |
| TREE_ALARM_WIDTH | number, default value is 8 | width of alarm bubble |
| TREE_ALARM_XOFFSET | number, default value is 0 | alarm bubble offset in x-axis direction |
| TREE_ALARM_YOFFSET | number, default value is 0 | alarm bubble offset in y-axis direction |
| TREE_ICONS_COLORS | array.number | icon |
| TREE_ICONS_GAP | number, default value is 1 | icon gap |
| TREE_ICONS_NAMES | array.string | icon name |
| TREE_ICON_HEIGHT | number, default value is 18 | icon height |
| TREE_ICON_PADDING | number, default value is 1 | padding amount of icon |
| TREE_ICON_WIDTH | number, default value is 18 | icon width |
| TREE_LABEL | string | label |
| TREE_LABEL_BOLD | boolean, default value is false | label bold |
| TREE_LABEL_BORDER | boolean, default value is false | label border |
| TREE_LABEL_COLOR | number | label color |
| TREE_LABEL_EMBED | boolean, default value is false | label embed text |
| TREE_LABEL_FONT | string | label font |
| TREE_LABEL_ITALIC | boolean, default value is false | label font italic |
| TREE_LABEL_SIZE | number | label font size |
| TREE_LABEL_UNDERLINE | boolean, default value is false | label font underline |

| | | |
|-------------------------------|--|---|
| TREE_LAYOUT | string, default value is label.icons.message Consts.TREE_LAYOUT_* | alignment type there are three part of tree node: label, icons and message . Set different layout style to put these three components |
| TREE_LAYOUT_GAP | number, default value is 2 | layout gap |
| TREE_MESSAGE | string | message |
| TREE_MESSAGE_BACKGROUND_ALPHA | number, default value is 1 0-1 | the transparency of message background |
| TREE_MESSAGE_BACKGROUND_COLOR | number | the background color of message |
| TREE_MESSAGE_BOLD | boolean, default value is false | message text bold |
| TREE_MESSAGE_CAPS_STYLE | string, default value is round | endpoint type of message Refer to flash.display.Graphics# lineStyle(...) parameters |
| TREE_MESSAGE_COLOR | number | message text color |
| TREE_MESSAGE_DEEP | number, default value is 0 | the depth of message view |
| TREE_MESSAGE_EMBED | boolean, default value is false | the font of message embed text |
| TREE_MESSAGE_FILL_ALPHA | number, default value is 1 0-1 | the transparency of message filling |
| TREE_MESSAGE_FILL_COLOR | number, default value is 16776960 | the filling color of message |
| TREE_MESSAGE_FONT | string | message text font |
| TREE_MESSAGE_GRADIENT | string, default value is spread.north | message gradient |
| TREE_MESSAGE_GRADIENT_ALPHA | number, default value is 1 0-1 | message gradient transparency |
| TREE_MESSAGE_GRADIENT_COLOR | number, default value is white | fill gradient color for message |
| TREE_MESSAGE_ITALIC | boolean, default value is false | fill message text font italic |
| TREE_MESSAGE_JOINT_STYLE | string, default value is round | endpoint style of message link Refer to flash.display.Graphics# lineStyle(...) parameters |
| TREE_MESSAGE_OUTLINE_ALPHA | number, default value is 1 0-1 | the transparency of message border |
| TREE_MESSAGE_OUTLINE_COLOR | number, default value is 0 | border color of message |
| TREE_MESSAGE_OUTLINE_WIDTH | number, default value is 0 | border width of message |
| TREE_MESSAGE_PERCENT | number, default value is 1 | percent of message filling |
| TREE_MESSAGE_PIXEL_HINTING | boolean, default value is true | whether use pixel fine-tuning for line Refer to flash.display.Graphics# lineStyle(...) parameters |
| TREE_MESSAGE_SCALE_MODE | string, default value is normal | zoom type of line |

| | | |
|------------------------|--|---|
| | | Refer to flash.display.Graphics#lineStyle(...) parameters |
| TREE_MESSAGE_SHAPE | string, default value is roundrect Consts.SHAPE_* | message shape |
| TREE_MESSAGE_SIZE | number | the font size of message text |
| TREE_MESSAGE_UNDERLINE | boolean, default value is false | message text underline |
| TREE_MESSAGE_WIDTH | number | messgae width |
| TREE_MESSAGE_XPADDING | number, default value is 2 | padding amount of message in x-axis direction |
| TREE_MESSAGE_YPADDING | number, default value is 0 | padding amount of message in y-axis direction |
| TREE_OUTER_STYLE | string, default value is border Consts.OUTER_* | border style |
| TREE_OUTER_WIDTH | number, default value is 2 | border width |

VECTOR

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|--|--|
| VECTOR_CAPS_STYLE | string, default value is round | endpoint type of line refer to flash.display.Graphics#lineStyle(...) parameters |
| VECTOR_DEEP | number, default value is 0 | the depth of picture |
| VECTOR_FILL | boolean, default value is true | fill picture |
| VECTOR_FILL_ALPHA | number, default value is 1 0-1 | fill transparency |
| VECTOR_FILL_COLOR | number, default value is 0xCCCCFF | fill color |
| VECTOR_GRADIENT | string, default value is none Consts.GRADIENT_* | gradient |
| VECTOR_GRADIENT_ALPHA | number, default value is 1 0-1 | gradient color transparency |
| VECTOR_GRADIENT_COLOR | number, default value is white | gradient color |
| VECTOR_GRADIENT_RECT | rectangle | gradient rectangle |
| VECTOR_JOINT_STYLE | string, default value is round | endpoint type of link refer to flash.display.Graphics#lineStyle(...) parameters |
| VECTOR_OUTLINE_ALPHA | number, default value is 1 0-1 | border transparency |
| VECTOR_OUTLINE_COLOR | number, default value is 0x5B5B5B | border color |
| VECTOR_OUTLINE_WIDTH | number, default value is 0 | border width |

| | | |
|-----------------------|--|--|
| VECTOR_PADDING | number, default value is 0 | padding amount |
| VECTOR_PADDING_BOTTOM | number, default value is 0 | bottom padding amount |
| VECTOR_PADDING_LEFT | number, default value is 0 | left padding amount |
| VECTOR_PADDING_RIGHT | number, default value is 0 | right padding amount |
| VECTOR_PADDING_TOP | number, default value is 0 | top padding amount |
| VECTOR_PIXEL_HINTING | boolean, default value is true | whether us pixel fine-tuning for line refer to flash.display.Graphics# lineStyle(...) parameters |
| VECTOR_SCALE_MODE | string, default value is normal | zoom type Refer to flash.display.Graphics# lineStyle(...) parameters |
| VECTOR_SHAPE | string, default value is rectangle Consts.SHAPE_* | shape |

WHOLE

| Style (twaver.Styles#) | Evaluation | Note |
|--------------------------|-----------------------------------|--|
| WHOLE_ALPHA | number, default value is 1 0-1 | the whole transparency of network element |