



TWaver[®] .NET

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

- TWaver .Net Home 
 - TWaver .NET Introduction
 - At a Glance
 - Getting Started
 - Convention
 - Setup Environment
 - Basic Knowledge
 - The Design Model and Frame of TWaver .NET
 - The Data Model of TWaver .NET
 - The Data Element of TWaver .NET
 - The Data Container of TWaver .NET
 - The View Components of TWaver .NET
 - Network 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
 - Basic Data Container
 - Alarm Model
 - QuickFinder
 - Network Component
 - Network Hierarchy
 - Network Backgroud
 - Network Filter
 - Network Function for Style Rule
 - Network GUI Interaction
 - FAQ
 - Using Element ID as value of Parent and Host When Serializing
 - Using TWaver WPF In WinForm
 - Integrate TWaver Silverlight with Microsoft Bing Maps Silverlight Control

TWaver .Net Home

This documentation introduces you how to use TWaver .NET library to create .NET application.

- [TWaver .NET Introduction](#)
- [At a Glance](#)
- [Getting Started](#)
- [Basic Knowledge](#)
- [Data Element](#)
- [Data Container](#)
- [Network Component](#)
- [FAQ](#)

TWaver .NET Introduction

TWaver .NET is a branch of TWaver based on Microsoft .NET technology. TWaver .NET provides the same functions as TWaver Java or TWaver Flex but with .NET 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 .NET technologies.

TWaver .NET 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 .NET 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 .NET predefines 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 .NET Object: TWaver .NET offers the graphical component network, other components like tree, table, map and etc. are coming soon.
4. GUI Special Effects: TWaver .NET takes full advantage of .NET technology to display abundance effects such as video, animation, skin, gradients and etc.
5. Auto-layout: TWaver .NET 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 server technologies.
7. API and detailed documents.

TWaver .NET Runtime Environment

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

TWaver .NET Development

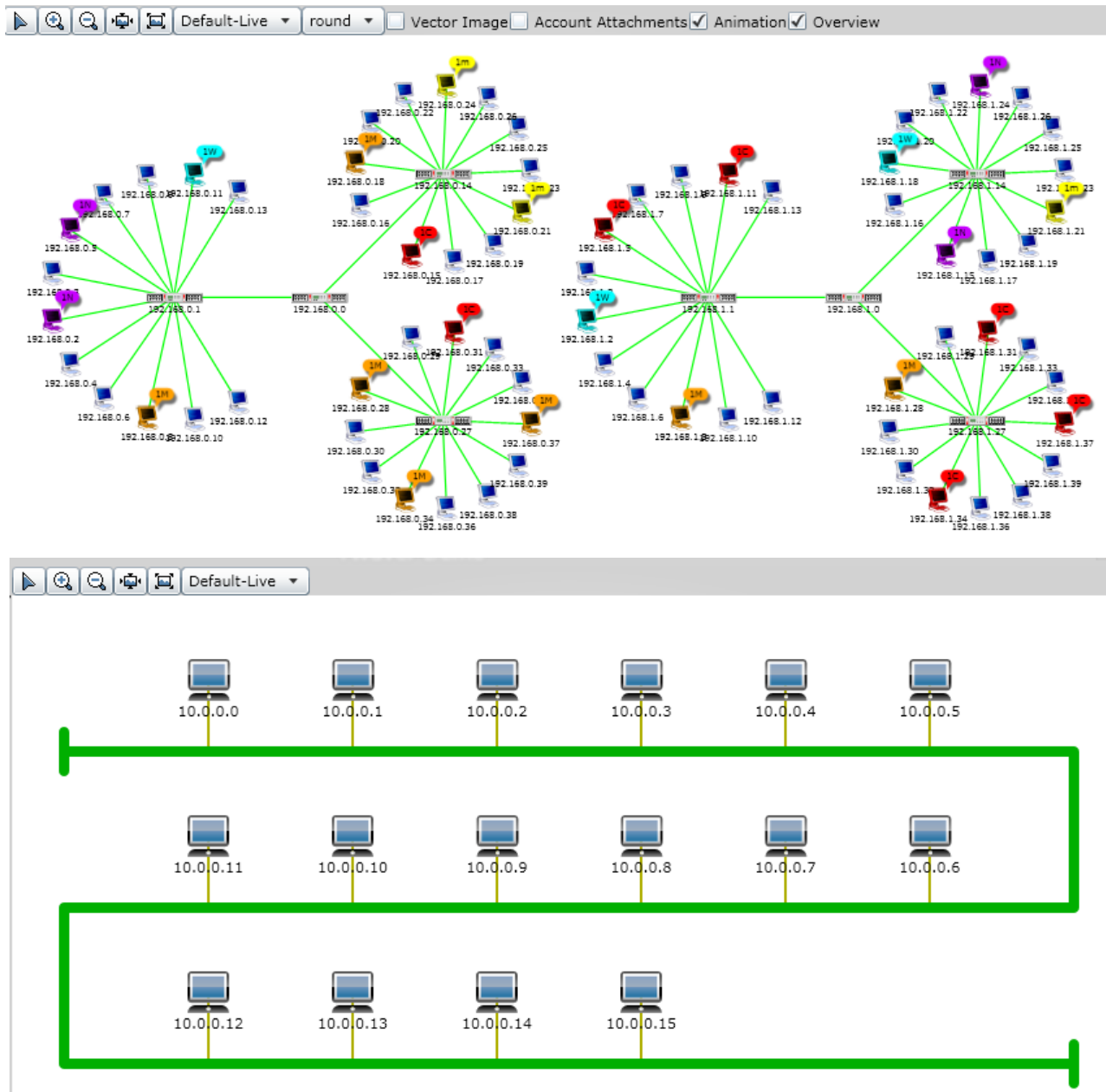
TWaver .NET supports Microsoft Silverlight 3.0 and Microsoft .NET Framework 3.5 or later.

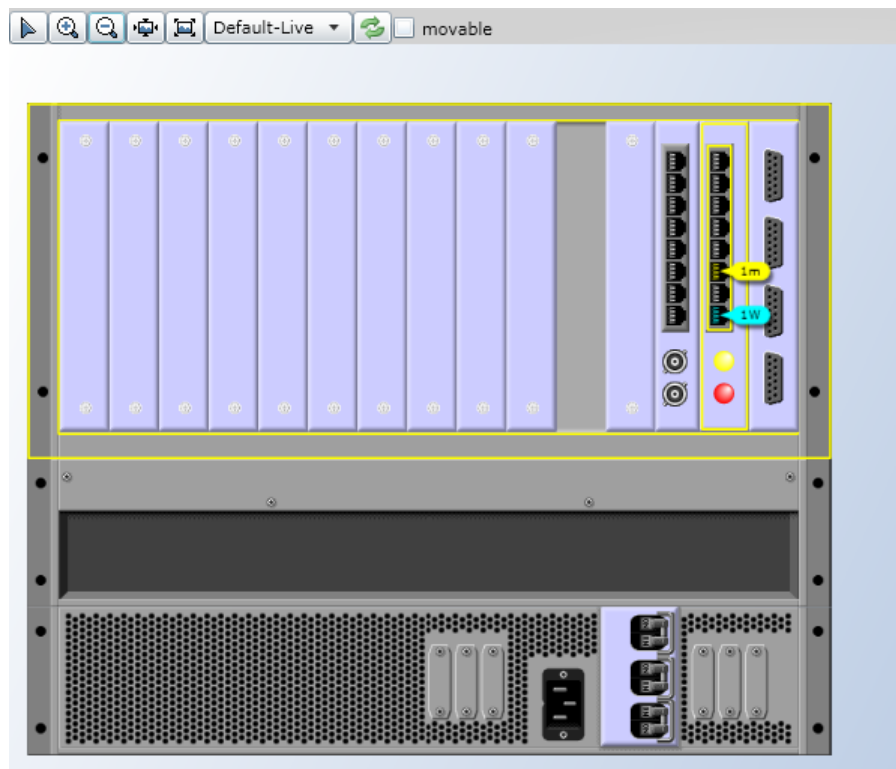
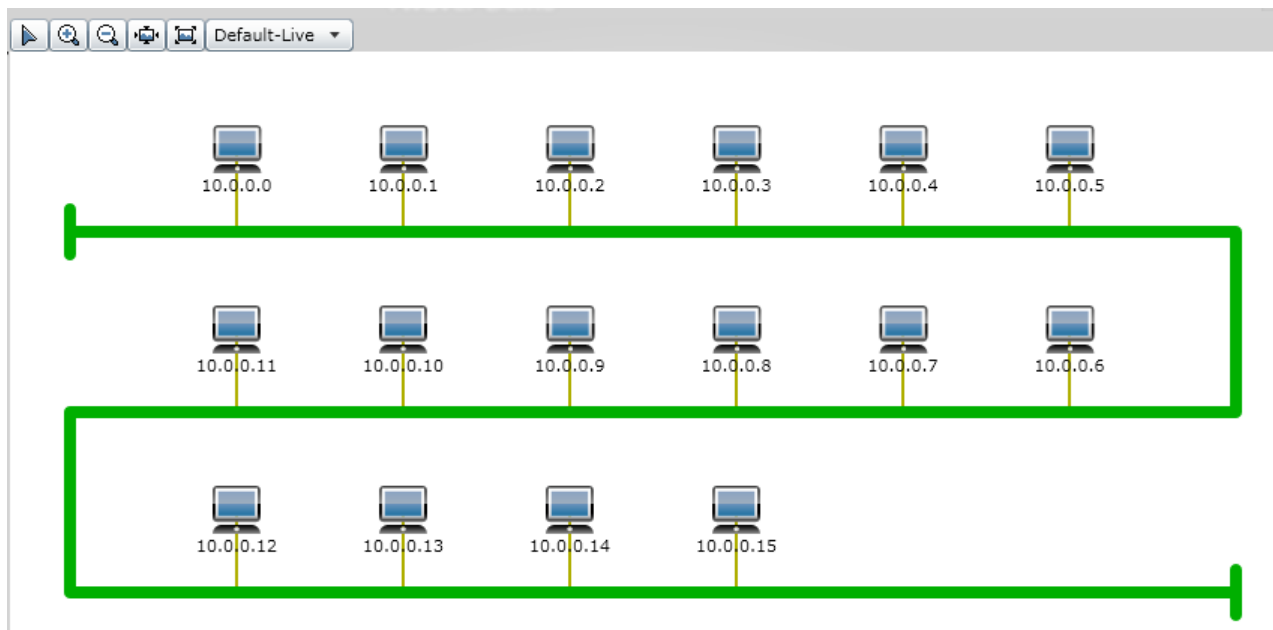
At a Glance

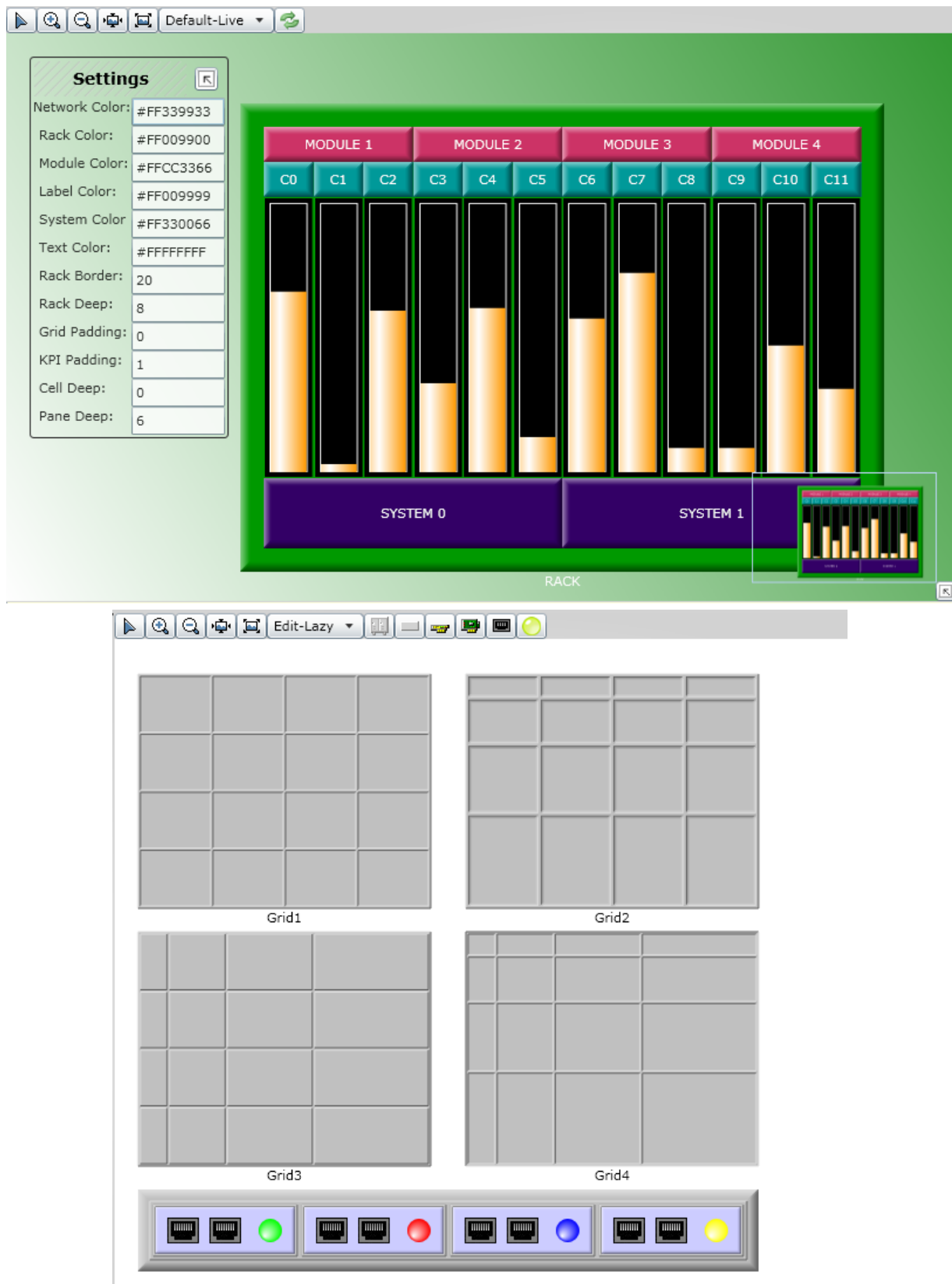
TWaver .NET consists of a series of visual graphic components. It offers a complete set of easy-to-use API that developers use to redevelop and customize. Data of TWaver .NET is maintained by DataBox designed to drive all visual components.

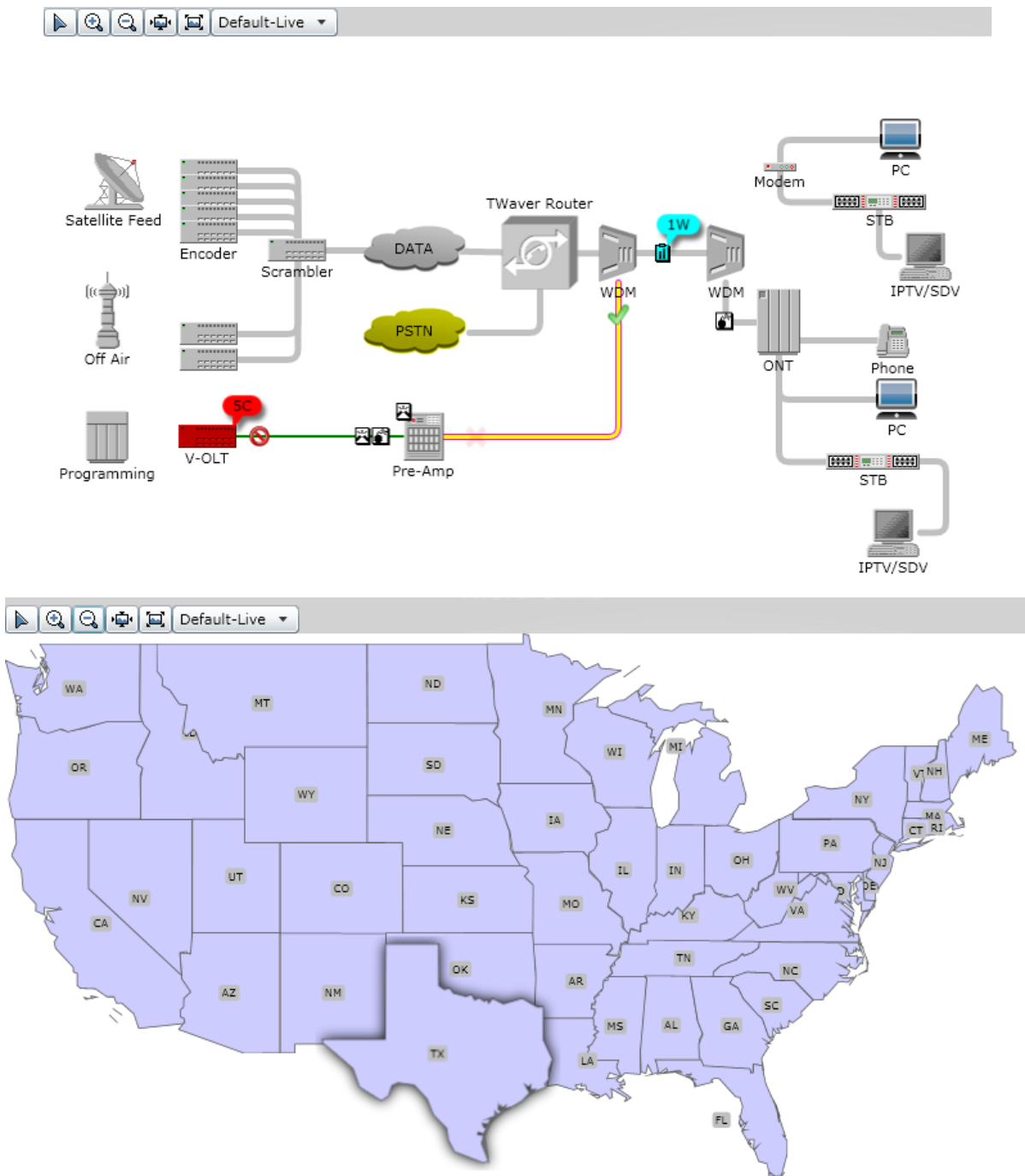
Network Component

Network component is the core component of TWaver .NET. It displays all kinds of topology, maps, equipment and etc.



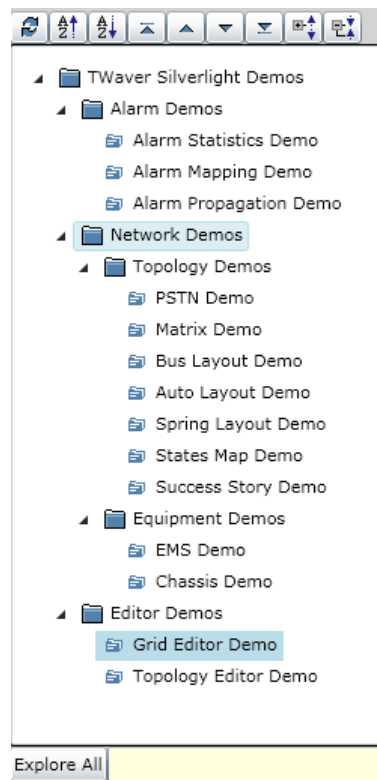






Tree Component

Tree Component is a customized tree which can display the hierarchy relationships of data and interact with Network component.



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 .NET to create a new project, and how to initially utilize TWaver .NET components.

Chapter Content:

- [Convention](#)
- [Setup Environment](#)

Convention

There are name conventions for TWaver .NET and phrase comparison with TWaver Java and TWaver Flex. 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 .NET

TWaver .NET	TWaver Flex	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
NA	Table	TElementTable	Table Component

The element property comparison between TWaver Java, TWaver Flex and TWaver .NET

TWaver .NET	TWaver Flex	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 .NET

TWaver .NET	TWaver Flex	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

Setup Environment

To use TWaver .NET, you need TWaver.Silverlight.dll/TWaver.Wpf.dll library, this can be downloaded from Serva Software website www.servasoftware.com. Also you need Microsoft Visual Studio 2008 or Microsoft Visual Studio 2010. If you are using Microsoft Visual Studio 2008, you should update to Service Pack 1 and install Microsoft Silverlight 3 Tools for Visual Studio 2008 SP1. Here are related download links from Microsoft website:

[Microsoft .NET Framework 3.5 Service Pack 1](#)

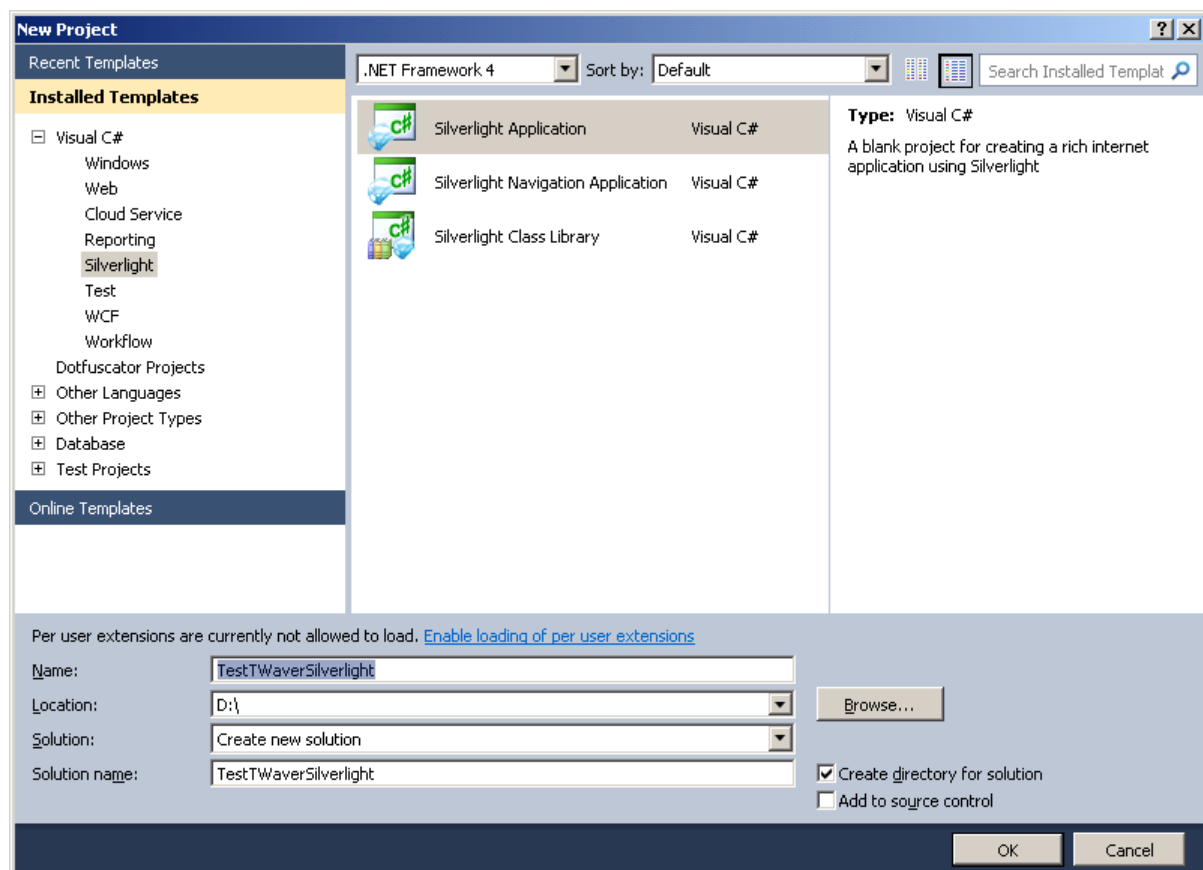
[Microsoft Visual Studio 2008 Service Pack 1](#)

[Microsoft Silverlight 3 Tools for Visual Studio 2008 SP1](#)

Create TWaver Silverlight Application

Create a Silverlight Project

1. Start Visual Studio 2008
2. On File menu, click New and then Project, The New Project dialog box appears.
3. In the Installed Templates pane, expand the Visual C# node and select Silverlight.
4. In the list of templates, select Silverlight Application.
5. Specify a name(TestTWaverSilverlight) and a location for the application and then click OK. The New Silverlight Application dialog box appears.
6. Uncheck the Host the Silverlight application in a new Web site check box, click OK button.



Add TWaver Silverlight library to project

1. Select the Silverlight project you just created, then select Project, then Add Reference from the Visual Studio main menu.
2. In the Add Reference dialog box, click the Browse tab.

3. Open the TWaver Silverlight library folder, select TWaver.Silverlight.dll file and click OK.

Create first application

1. Add TWaver Silverlight assembly and TWaver Network to MainPage.xaml

```
<UserControl x:Class="TestTWaverSilverlight.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:tn="clr-namespace:TWaver.Network;assembly=TWaver.Silverlight">

    <Grid x:Name="LayoutRoot" Background="White">
        <tn:Network x:Name="network"/>
    </Grid>
</UserControl>
```

2. Add Node to Databox

```
using System.Windows.Controls;
using TWaver;

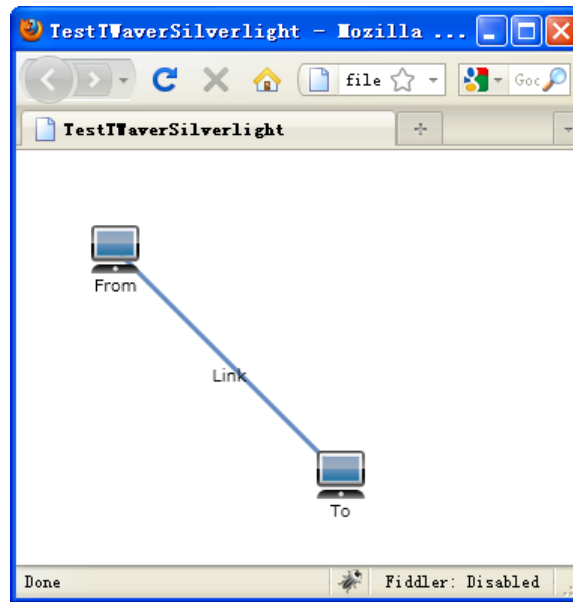
namespace TestTWaverSilverlight
{
    public partial class MainPage : UserControl
    {
        public MainPage()
        {
            InitializeComponent();
            InitDataBox();
        }

        private void InitDataBox()
        {
            Node node1 = new Node();
            node1.Location = new System.Windows.Point(50, 50);
            node1.Name = "Node1";
            network.ElementBox.Add(node1);

            Node node2 = new Node();
            node2.Location = new System.Windows.Point(200, 200);
            node2.Name = "Node2";
            network.ElementBox.Add(node2);

            Link link = new Link(node1, node2);
            link.Name = "Link";
            network.ElementBox.Add(link);
        }
    }
}
```

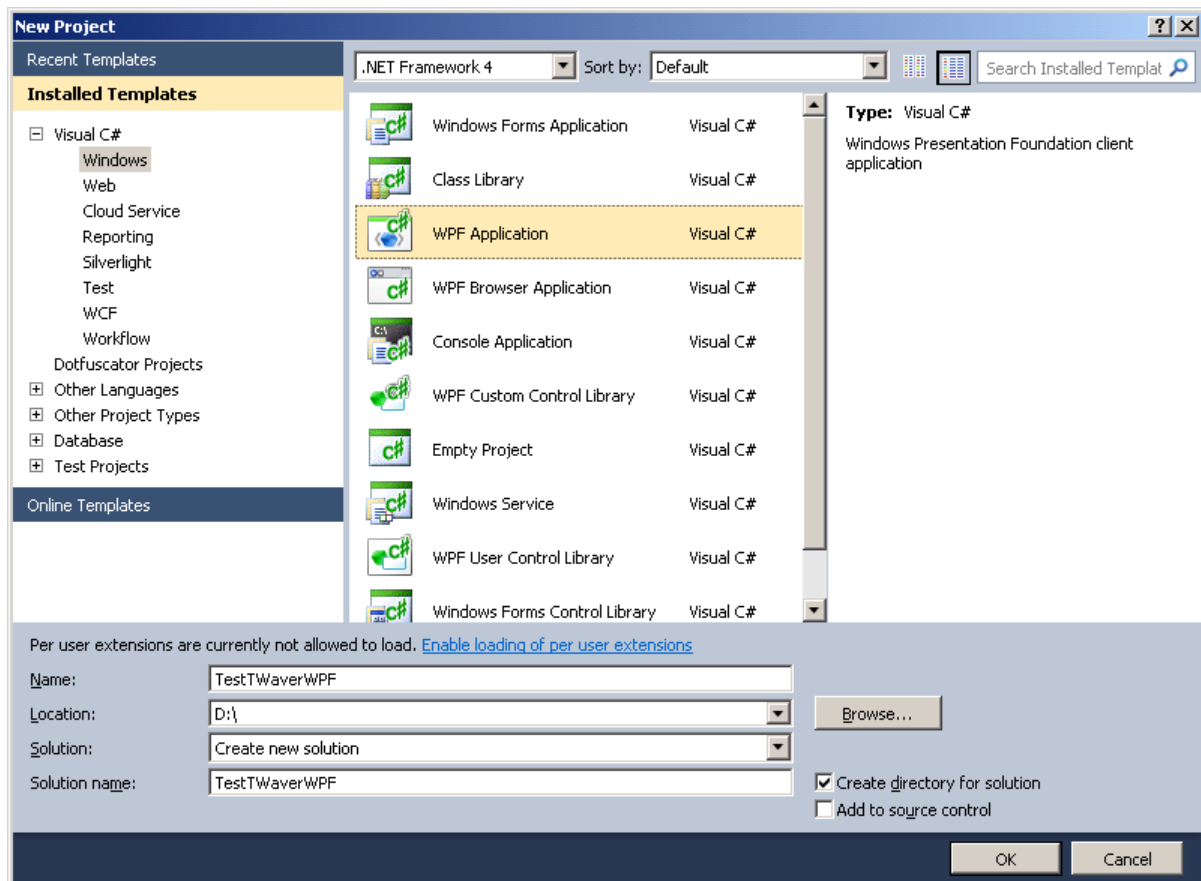
3. Run application



Create TWaver WPF Application

Create a WPF Project

1. Start Visual Studio 2008
2. On File menu, click New and then Project, The New Project dialog box appears.
3. In the Installed Templates pane, expand the Visual C# node and select Windows.
4. In the list of templates, select WPF Application.
5. Specify a name(TestTWaverWPF) and a location for the application and then click OK.



Add TWaver WPF library to project

1. Select the WPF project you just created, then select Project, then Add Reference from the Visual Studio main menu.
2. In the Add Reference dialog box, click the Browse tab.
3. Open the TWaver WPF library folder, select TWaver.WPF.dll file and click OK.

Create first application

1. Add TWaver WPF assembly and TWaver Network to MainWindow.xaml

```
<Window x:Class="TestTWaverWPF.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:tn="clr-namespace:TWaver.Network;assembly=TWaver.WPF"
    Title="TWaver WPF Demo" Height="300" Width="350">
    <Grid>
        <tn:Network x:Name="network"/>
    </Grid>
</Window>
```

2. Add Node to Databox

```
using System.Windows;
using TWaver;

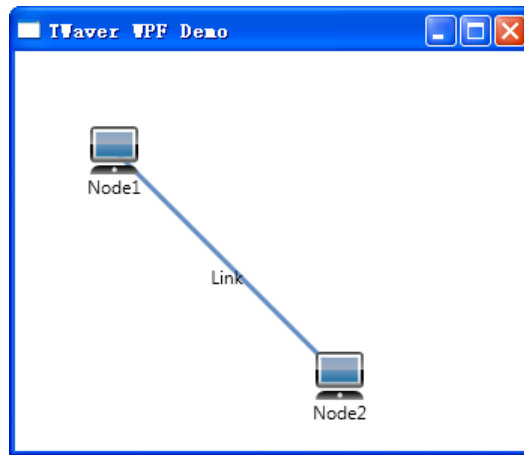
namespace TestTWaverWPF
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            InitDataBox();
        }

        private void InitDataBox()
        {
            Node node1 = new Node();
            node1.Location = new System.Windows.Point(50, 50);
            node1.Name = "Node1";
            network.ElementBox.Add(node1);

            Node node2 = new Node();
            node2.Location = new System.Windows.Point(200, 200);
            node2.Name = "Node2";
            network.ElementBox.Add(node2);

            Link link = new Link(node1, node2);
            link.Name = "Link";
            network.ElementBox.Add(link);
        }
    }
}
```

3. Run application



Basic Knowledge

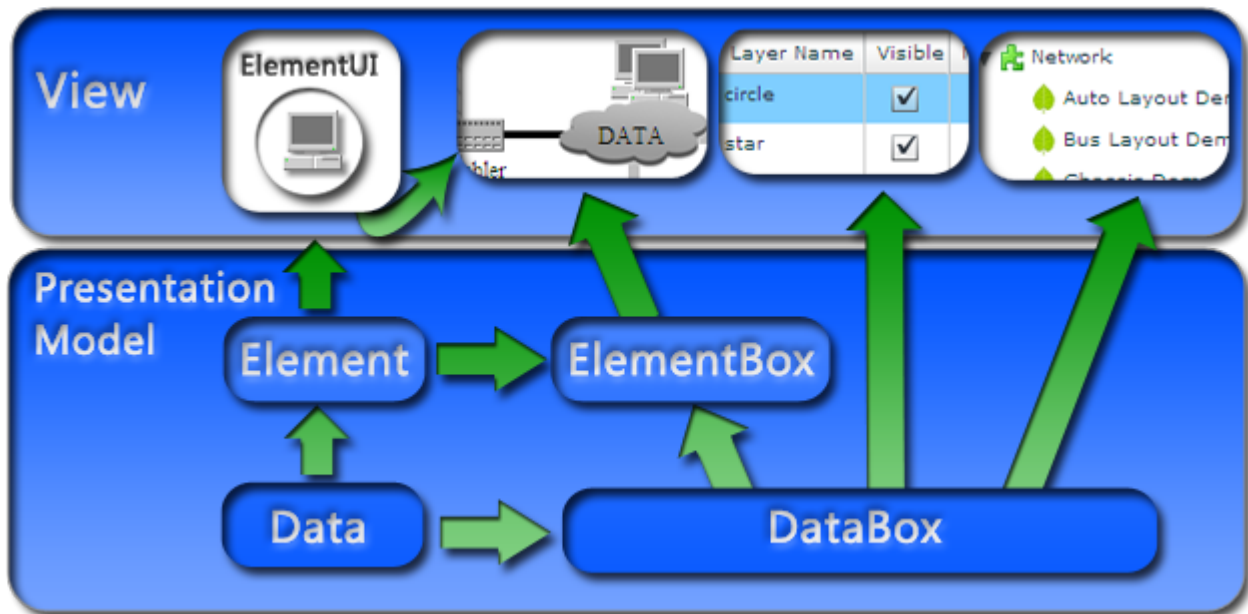
This chapter introduces the UI components and design model of TWaver .NET. Also included is a brief overview of the data model and the types of visual components. After reading this chapter you will know the principles of TWaver .NET.

- [The Design Model and Frame of TWaver .NET](#)
- [The Data Model of TWaver .NET](#)
- [The View Components of TWaver .NET](#)
- [Data Serialization](#)

The Design Model and Frame of TWaver .NET

TWaver .NET 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.Network will serve as the data container components. This is the frame of TWaver .NET.

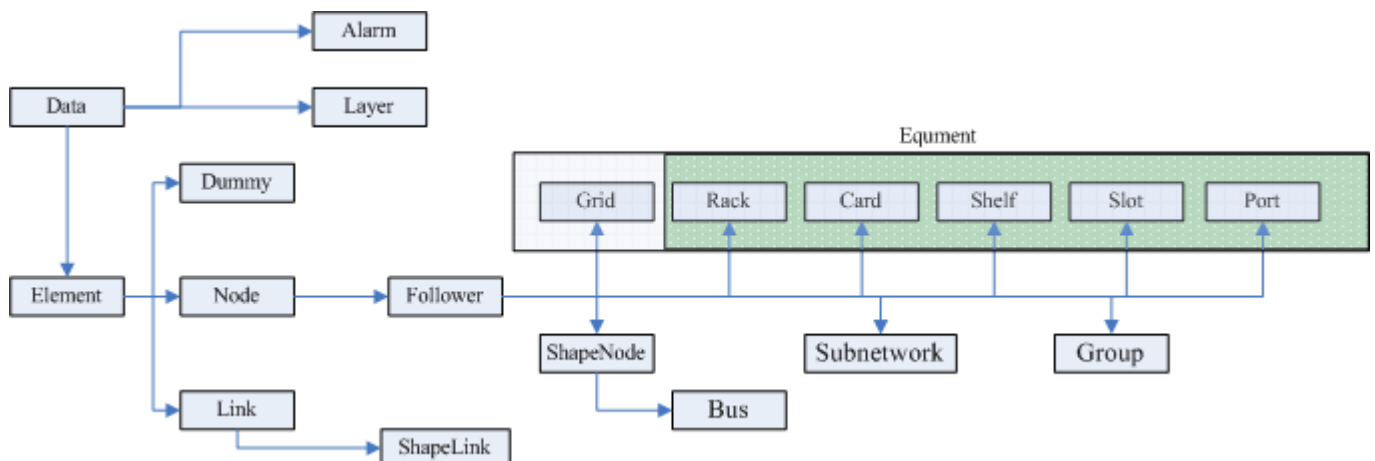
TWaver Design Model



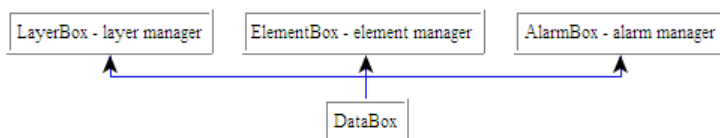
The Data Model of TWaver .NET

Base on two basic elements, which TWaver.IData and TWaver.DataBox correspond to be the basic data element and data container, TWaver .NET 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 .NET](#)
- [The Data Container of TWaver .NET](#)

The Data Element of TWaver .NET

TWaver .NET takes TWaver.IData as the basic data element, extends series of data element with GUI and business logic including IAlarm, ILayer, IElement...

- **TWaver.IData**

IData is the basic interface of TWaver .NET 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.

TWaver.Data serve TWaver.IData as its interface, and contains delegate TWaver.EventListener, realizes the dispatching and listening of property change.

```
namespace TWaver
{
    public delegate void EventListener<TEvent>(TEvent evt);
}
```

```
public event EventListener<PropertyChangeEvent<IData>> PropertyChange;
public virtual bool DispatchPropertyChangeEvent(string propertyName,
    object oldValue, object newValue)
{
    if (oldValue == newValue)
    {
        return false;
    }
    if (PropertyChange != null)
    {
        PropertyChange(new PropertyChangeEvent<IData>(this, propertyName,
            oldValue, newValue));
    }
    this.OnPropertyChanged(propertyName, oldValue, newValue);
    return true;
}

protected virtual void OnPropertyChanged(string propertyName, object oldValue,
    object newValue)
{
}
```

More, Data owns other functions:

```
public int ChildrenCount
public bool HasChildren
public IList<IData> Children
public IList<IData> ToChildren()
public IList<TData> ToChildren<TData>() where TData : class, IData
public IList<IData> ToChildren(Predicate<IData> match)
public IList<TData> ToChildren<TData>(Predicate<TData> match) where TData : class, IData
```

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 .NET 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.

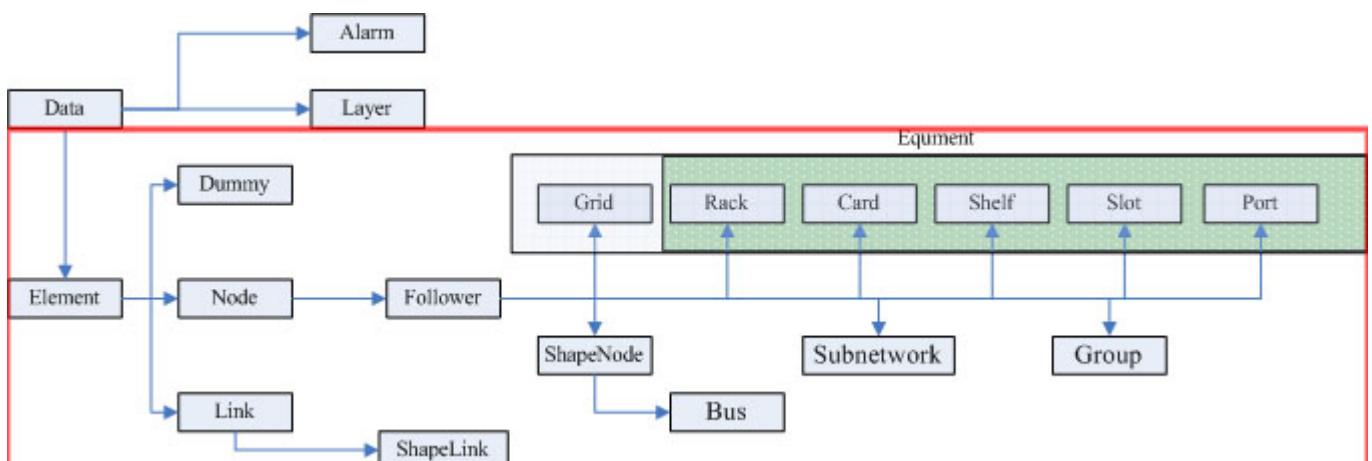
• TWaver.IElement

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 other views like TWaver.Network.Network.



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 .NET

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

- **DataBox**

DataBox contains EventListener of DataBoxChange, DataPropertyChange, PropertyChange and HierarchyChange. User can master all Element transformation by listening DataBox, and can handle the event by rewriting on***Changed function:

```
public event EventListener<DataBoxChangeEvent<TData>> DataBoxChange;
public event EventListener<PropertyChangeEvent<TData>> DataPropertyChange;
public event EventListener<PropertyChangeEvent<DataBox<TData>>> PropertyChange;
public event EventListener<HierarchyChangeEvent<TData>> HierarchyChange;
protected virtual void OnPropertyChanged(string propertyName,
    object oldValue, object newValue)
protected virtual void OnDataPropertyChange(PropertyChangeEvent<TData> e)
protected virtual void OnClientChanged(string clientProp,
    object oldValue, object newValue)
```

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

```
public class DataBox<TData> : IClient where TData: class, IData
public virtual void Add(TData data)
public virtual void Add(TData data, int index)
public virtual void Remove(TData data)
public virtual void RemoveByID(object id)
public virtual void Clear()
public TData GetDataByID(object id)
public bool Contains(TData data)
public bool ContainsByID(object id)
```

There are iterate methods in DataBox as below:

```
public void ForEach(Action<TData> callbackFunction)
public void ForEachPredicate(Predicate<TData> callbackFunction)
public void ForEachByBreadthFirst(Action<TData> callbackFunction)
public void ForEachByBreadthFirst(Action<TData> callbackFunction, TData data)
public void ForEachPredicateByDepthFirst(Predicate<TData> callbackFunction)
public void ForEachPredicateByDepthFirst(Predicate<TData> callbackFunction, TData data)
public void ForEachByDepthFirst(Action<TData> callbackFunction)
public void ForEachByDepthFirst(Action<TData> callbackFunction, TData data)
public void ForEachPredicateByBreadthFirst(Predicate<TData> callbackFunction)
public void ForEachPredicateByBreadthFirst(Predicate<TData> callbackFunction, TData data)
```

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

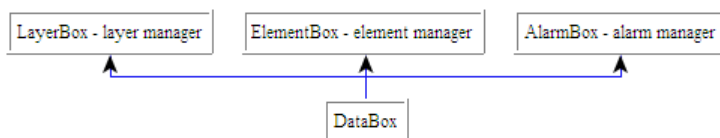
```
public void MoveDown(TData data)
```

```
public void MoveTo(TData data, int newIndex)
public void MoveToBottom(TData data)
public void MoveToTop(TData data)
public void MoveUp(TData data)
```

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

```
public SelectionModel<TData> SelectionModel
public void MoveSelectionDown()
public void MoveSelectionDown(SelectionModel<TData> sm)
public void MoveSelectionToBottom()
public void MoveSelectionToBottom(SelectionModel<TData> sm)
public void MoveSelectionToTop()
public void MoveSelectionToTop(SelectionModel<TData> sm)
public void MoveSelectionUp()
public void MoveSelectionUp(SelectionModel<TData> sm)
```

TWaver .NET 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:

```
public ElementBox ElementBox
public ILayer DefaultLayer()
public ILayer GetLayerByElement(IElement element)
```

• AlarmBox

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

```
public ElementBox ElementBox
public IList<IArm> GetCorrespondingAlarms(IElement element)
public IList<IElement> GetCorrespondingElements(IArm alarm)
```

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 IAlarmElementMapping AlarmElementMapping
```

More, there are other options:

```
//When network is removed from ElementBox, whether the related alarm need to be deleted
public bool IsRemoveAlarmWhenElementIsRemoved
//Whether need to remove Alarm when the alarm level is Cleared
public bool IsRemoveAlarmWhenAlarmIsCleared
```

- **ElementBox**

ElementBox includes AlarmBox, LayerBox, is the network management container. TWaver .NET defined proper view components for ElementBox to show the topology relationships. ElementBox also contains AlarmStatePropagator, it is used to handle alarm propagation.

```
public LayerBox LayerBox
public AlarmBox AlarmBox
public AlarmStatePropagator AlarmStatePropagator
```

Network:



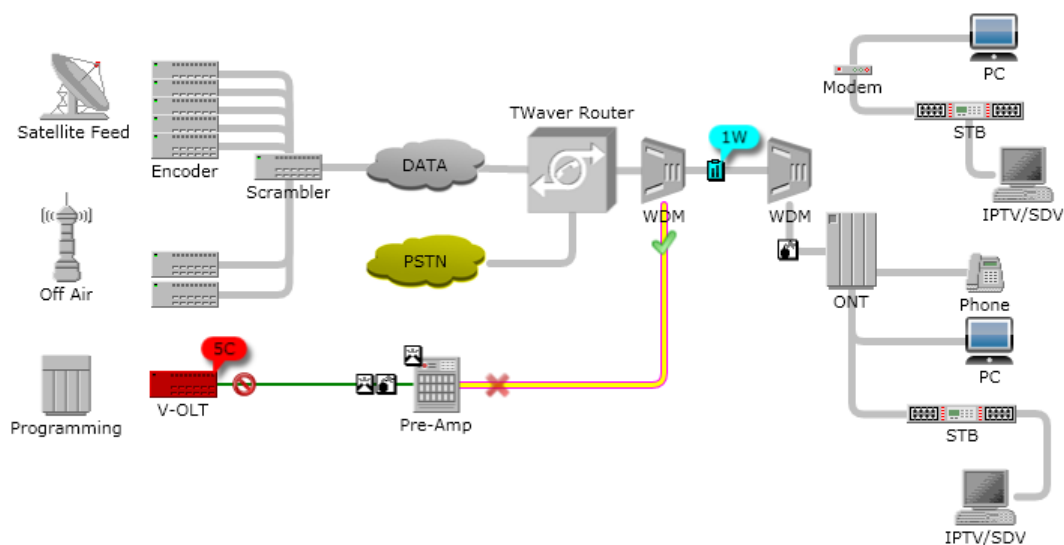
Network Introduction

• 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 Silverlight/WPF, includes many containers to hold different components.

Network hierarchy:

```
-> ScrollViewer
  -> NetworkCanvas
    -> RootCanvas
      -> TopCanvas
        -> AttachmentCanvas
          -> LayerCanvas
            -> Layer n
            -> Layer ...
            -> Default Layer
          -> BottomCanvas
            -> BackgroundCanvas
```

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 ElementUI GetElementUI(IElement element)
```

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.

```
//Attachment constructor
public Attachment(ElementUI elementUI, bool showInAttachmentCanvas)
```

```
//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 GraphCanvas AttachmentCanvas
```

```
//ElementUI can get all attachments of network element
public IList<Attachment> Attachments
```

• 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 void SetDefaultInteractionHandlers(bool lazyMode);
//switch to Edit Interaction Model
public void SetEditInteractionHandlers(bool lazyMode);
//switch to Create Element Interaction Model
public void SetCreateElementInteractionHandlers(Type type);
//switch to Create Link Interaction Model
public void SetCreateLinkInteractionHandlers(Type type);
//switch to Create ShapeLink Interaction Model
public void SetCreateShapeLinkInteractionHandlers(Type type);
//switch to Create ShapeNode Interaction Model
public void SetCreateShapeNodeInteractionHandlers(Type type);
```

Function

```
public Predicate<IElement> VisibleFunction
public Predicate<IElement> MovableFunction
public Predicate<IElement> EditableFunction
```

Some Properties Listed

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

```
//set current SubNetwork
public ISubNetwork CurrentSubNetwork
//up into higher level SubNetwork
public void UpSubNetwork(bool animate, Action finishFunction)
//Interaction event listener
public event EventHandler<InteractionEvent> Interaction;
//refresh view of network element
public void InvalidateElementUI(IElement element, bool checkAttachments)
//get view of network element
public ElementUI GetElementUI(IElement element)

//zoom operation
public double Zoom
public void ZoomIn(bool animate)
public void ZoomOut(bool animate)
public void ZoomReset(bool animate)
public void ZoomOverview(bool animate)

//get container of components in topology
public GraphCanvas RootCanvas
public GraphCanvas TopCanvas
public GraphCanvas AttachmentCanvas
public GraphCanvas LayerCanvas
public GraphCanvas BottomCanvas
public GraphCanvas BackgroundCanvas
```


Tree Introduction

TWaver.Controls.Tree implements TreeView component of Silverlight/WPF, it bounds with DataBox containers such as LayerBox, AlarmBox, ElementBox and etc. The hierarchy of Tree view is decided by the father-child relationships in DataBox container. TWaver Tree is very easy to use, interact with network by sharing DataBox. And there are bunch of features to customize Tree.

1. ShareSelectionModel, Tree shares selection model with DataBox by default, this means the Node on tree is selected too if user select the Node from Network, you can set it to false to create its own selection model.
2. VisibleFunction, it's used to filter if a node is visible on the tree. The default value is null and the the define is shown as below:

```
public Predicate<TData> VisibleFunction { get; set; }
```

3. CompareFunction, it's used to sort the tree. The default value is null and the the define is shown as below:

```
public Comparison<TData> CompareFunction { get; set; }
```

4. LabelFunction, it's used to generate label for the tree node. The default value is Defaults.TREE_LABEL_FUNCTION and the the define is shown as below:

```
public Func<IData, string> LabelFunction { get; set; }
```

```
public static Func<IElement, string> LABEL_FUNCTION = (element) =>
{
    string label = element.GetStyle<string>(Styles.NETWORK_LABEL);
    if (label != null){
        return label;
    }
    return element.Name;
};
```

5. IconFunction, it's used to generate icon for the tree node. The default value is Defaults.ICON_FUNCTION and the the define is shown as below:

```
public Func<IData, string> IconFunction { get; set; }
```

```
public static Func<IData, string> ICON_FUNCTION = (data) =>
{
    return data.Icon;
};
```

6. ToolTipFunction, it's used to generate tooltip for the tree node. The default value is Defaults.TOOLTIP_FUNCTION and the the define is shown as below:

```
public Func<IData, string> ToolTipFunction { get; set; }
```

```
public static Func<IData, string> TOOLTIP_FUNCTION = (data) => data.ToolTip;
```

7. InnerColorFunction, it's used to generate inner border for the tree node. The default value is Defaults.INNER_COLOR_FUNCTION and the the define is shown as below:

```
public Func<IData, Color?> InnerColorFunction { get; set; }
```

```
public static Func<IData, Color?> INNER_COLOR_FUNCTION = (data) =>
{
    IElement element = data as IElement;
    if (element != null)
    {
        AlarmSeverity severity = element.AlarmState.HighestNativeAlarmSeverity;
        if (severity != null)
        {
            return severity.Color;
        }
        return element.GetStyle<Color?>(Styles.INNER_COLOR);
    }
    return null;
};
```

8. OuterColorFunction, it's used to generate outer border for the tree node. The default value is Defaults.OUTER_COLOR_FUNCTION and the the define is shown as below:

```
public Func<IData, Color?> OuterColorFunction { get; set; }
```

```
public static Func<IData, Color?> OUTER_COLOR_FUNCTION = (data) =>
{
    IElement element = data as IElement;
    if (element != null)
    {
        AlarmSeverity severity = element.AlarmState.PropagateSeverity;
        if (severity != null)
        {
            return severity.Color;
        }
        return element.GetStyle<Color?>(Styles.OUTER_COLOR);
    }
    return null;
};
```

9. MakeVisibleOnSelected, it's used to determine whether the tree node is visible on the tree when the node is selected. The default value is Defaults.TREE_MAKE_VISIBLE_ON_SELECTED.

10. Other useful methods:

```
public void CollapseAll();  
public void CollapseAll(TData data);  
public void Expand(TData data);  
public void ExpandAll();  
public void ExpandAll(TData data);  
public void ExpandDescendant(TData data);  
public TreeItem<TData> ExpandTo(TData data);  
public bool IsSelected(TData data);  
public void MakeVisible(TData data);
```

Data Serialization

This topic describes how to serialize DataBox to XML String or deserialize XML String to DataBox.

1. Construct an instance of XMLSerializer

```
XMLSerializer<IElement> xmlSerializer = new XMLSerializer<IElement>(this.network.ElementBox);
```

2. Configure serializing option by using property SerializationSettings of XMLSerialize. For example, add the following code if you don't want to output element ID.

```
xmlSerializer.SerializationSettings.RegisterProperty("ID", null, false);
```

3. Serialize or deserialize.

```
this.txtResult.Text = xmlSerializer.Serialize();
```

```
xmlSerializer.Deserialize(this.txtResult.Text);
```

Here is the whole source code:

```
<UserControl x:Class="TestTWaverSilverlight.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    >

    <Grid x:Name="LayoutRoot">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="500" />
        </Grid.ColumnDefinitions>

        <Grid x:Name="networkPane" Grid.Column="0" Grid.Row="0"/>
        <Grid Grid.Column="1" Grid.Row="0">
            <Grid.RowDefinitions>
                <RowDefinition Height="*" />
                <RowDefinition Height="Auto" />
            </Grid.RowDefinitions>

            <TextBox x:Name="txtResult" AcceptsReturn="True" Grid.Column="0" Grid.Row="0"/>
            <StackPanel Grid.Column="0" Grid.Row="1" Orientation="Horizontal" HorizontalAlignment="Right">
                <Button x:Name="btnSerialize" Content="Serialize" Click="btnSerialize_Click" />
                <Button x:Name="btnDeserialize" Content="Deserialize" Click="btnDeserialize_Click" />
            </StackPanel>
        </Grid>
    </Grid>
</UserControl>
```

```
using System.Windows;
using System.Windows.Controls;
using TWaver;
using TWaver.Network;

namespace TestTWaverSilverlight
{
    public partial class MainPage : UserControl
    {
        private Network network = new Network();

        public MainPage()
        {
            InitializeComponent();
            this.networkPane.Children.Add(network);
            InitDataBox();
        }

        private void InitDataBox()
        {
            Node from = new Node();
            from.Location = new Point(50, 50);
            from.Name = "From";
            network.ElementBox.Add(from);

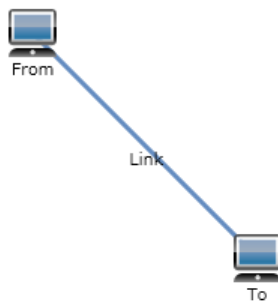
            Node to = new Node();
            to.Location = new Point(200, 200);
            to.Name = "To";
            network.ElementBox.Add(to);

            Link link = new Link(from, to);
            link.Name = "Link";
            network.ElementBox.Add(link);
        }

        private void btnSerialize_Click(object sender, System.Windows.RoutedEventArgs e)
        {
            XMLSerializer<IElement> xmlSerializer = new XMLSerializer<IElement>(this.network.ElementBox);
            this.txtResult.Text = xmlSerializer.Serialize();
        }

        private void btnDeserialize_Click(object sender, System.Windows.RoutedEventArgs e)
        {
            XMLSerializer<IElement> xmlSerializer = new XMLSerializer<IElement>(this.network.ElementBox);
            this.network.ElementBox.Clear();
            xmlSerializer.Deserialize(this.txtResult.Text);
        }
    }
}
```

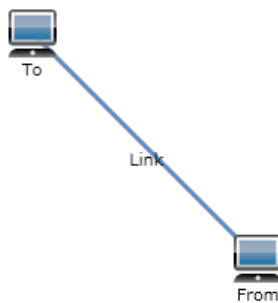
Click button Serialize:



```
<?xml version="1.0" encoding="utf-16"?>
<TWaver V="1.0" P="SILVERLIGHT">
  <DataBox Type="TWaver.ElementBox">
    <LayerBox>
      <Layer Name="default" IsVisible="True" IsEditable="True" IsMovable="True" />
    </LayerBox>
  </DataBox>
  <Data Type="TWaver.Node" Ref="0" ID="567a011e-a2ab-47c0-a48a-b57ffebde34c">
    <P N="Name"><![CDATA[From]]></P>
    <P N="Location" X="50" Y="50" />
    <P N="Width">32</P>
    <P N="Height">32</P>
  </Data>
  <Data Type="TWaver.Node" Ref="1" ID="632b219f-2329-4129-921c-d8f2845c2238">
    <P N="Name"><![CDATA[To]]></P>
    <P N="Location" X="200" Y="200" />
    <P N="Width">32</P>
    <P N="Height">32</P>
  </Data>
  <Data Type="TWaver.Link" Ref="2" ID="ffb62b15-2524-48a1-9e4a-6922b13f5004">
    <S N="select.shadow.depth">2</S>
    <S N="outer.radius">8</S>
    <S N="outer.width">1</S>
    <P N="Name"><![CDATA[Link]]></P>
    <P N="FromNode" Ref="0" />
    <P N="ToNode" Ref="1" />
  </Data>
</TWaver>
```

Serialize Deserialize

Change "From" to "To", "To" to "From", then click button Deserialize:



```
<?xml version="1.0" encoding="utf-16"?>
<TWaver V="1.0" P="SILVERLIGHT">
  <DataBox Type="TWaver.ElementBox">
    <LayerBox>
      <Layer Name="default" IsVisible="True" IsEditable="True" IsMovable="True" />
    </LayerBox>
  </DataBox>
  <Data Type="TWaver.Node" Ref="0" ID="567a011e-a2ab-47c0-a48a-b57ffebde34c">
    <P N="Name"><![CDATA[To]]></P>
    <P N="Location" X="50" Y="50" />
    <P N="Width">32</P>
    <P N="Height">32</P>
  </Data>
  <Data Type="TWaver.Node" Ref="1" ID="632b219f-2329-4129-921c-d8f2845c2238">
    <P N="Name"><![CDATA[From]]></P>
    <P N="Location" X="200" Y="200" />
    <P N="Width">32</P>
    <P N="Height">32</P>
  </Data>
  <Data Type="TWaver.Link" Ref="2" ID="ffb62b15-2524-48a1-9e4a-6922b13f5004">
    <S N="select.shadow.depth">2</S>
    <S N="outer.radius">8</S>
    <S N="outer.width">1</S>
    <P N="Name"><![CDATA[Link]]></P>
    <P N="FromNode" Ref="0" />
    <P N="ToNode" Ref="1" />
  </Data>
</TWaver>
```

Serialize Deserialize

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 .NET implement interface `TWaver.IData`. TWaver .NET 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 .NET 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 contain event EventListener<PropertyChangeEvent<IData>> PropertyChange which gives its the event dispatching and listening function. It realizes event dispatching and listening by following methods:

```
public event EventListener<PropertyChangeEvent<IData>> PropertyChange;
```

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

```
bool DispatchPropertyChangeEvent(string propertyName, object oldValue, object newValue);
```

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

```
public IList<IData> Children { get; }
public int ChildrenCount { get; }
public bool HasChildren { get; }
public virtual bool IsDescendantOf(IData data);
public virtual bool IsParentOf(IData data);
public virtual bool IsRelatedTo(IData data);
```

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 .NET 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 Data();
public Data(object id);
public virtual string Icon { get; set; }
public object ID { get; }
public virtual IData this[int index] { get; }
public virtual string Name { get; set; }
public virtual IData Parent { get; set; }
public virtual string ToolTip { get; set; }
```

User can put up other properties for elment by implementing setPropertyValue(...) method which is similar to Directory in .Net.

```
public void SetPropertyValue(String property, Object value);
public Object GetPropertyValue(String property);
```

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


```
public void SerializeXML(XMLSerializer serializer, IData newInstance);  
public void DeserializeXML(XMLSerializer serializer, XML xml);
```

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 SetClient method.

The properties of Alarm:

```
public AlarmSeverity AlarmSeverity { get; set; }
public object ElementID { get; }
public bool IsAcked { get; set; }
public bool IsCleared { get; set; }
```

Alarm Levels

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

```
public static AlarmSeverity CRITICAL = Register(500, "Critical", "C", Color.FromArgb(255, 255, 0, 0), null);
public static AlarmSeverity MAJOR = Register(400, "Major", "M", Color.FromArgb(255, 255, 160, 0), null);
public static AlarmSeverity MINOR = Register(300, "Minor", "m", Color.FromArgb(255, 255, 255, 0), null);
public static AlarmSeverity WARNING = Register(200, "Warning", "W", Color.FromArgb(255, 0, 255, 255), null);
public static AlarmSeverity INDETERMINATE = Register(100, "Indeterminate", "N", Color.FromArgb(255, 200, 0, 255), null);
public static AlarmSeverity CLEARED = Register(0, "Cleared", "R", Color.FromArgb(255, 0, 255, 0), null);
```

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 .NET offer method to clear all alarm levels(Pay attention to use clearing all levels method, it will effect whole program).

```
public static AlarmSeverity Register(int value, string name, string nickName, Color color, string displayName);
```

```
public static AlarmSeverity Unregister(string name);
public static void Clear();
```

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

```
public static event EventListener<AlarmSeverityChangeEvent> AlarmSeverityChange;
```

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 .NET 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 AlarmSeverity HighestAcknowledgedAlarmSeverity { get; }
public AlarmSeverity HighestNativeAlarmSeverity { get; }
public AlarmSeverity HighestNewAlarmSeverity { get; }
public AlarmSeverity HighestOverallAlarmSeverity { get; }
public AlarmSeverity PropagateSeverity { get; set; }

public uint GetAcknowledgedAlarmCount();
public uint GetAcknowledgedAlarmCount(AlarmSeverity severity);
public uint GetAlarmCount();
public uint GetAlarmCount(AlarmSeverity severity);
public uint GetNewAlarmCount();
public uint GetNewAlarmCount(AlarmSeverity severity);
```

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

```
public void DecreaseAcknowledgedAlarm(AlarmSeverity severity);
public void DecreaseAcknowledgedAlarm(AlarmSeverity severity, uint decrement);
public void DecreaseNewAlarm(AlarmSeverity severity);
public void DecreaseNewAlarm(AlarmSeverity severity, uint decrement);

public void IncreaseAcknowledgedAlarm(AlarmSeverity severity);
public void IncreaseAcknowledgedAlarm(AlarmSeverity severity, uint increment);
public void IncreaseNewAlarm(AlarmSeverity severity);
public void IncreaseNewAlarm(AlarmSeverity severity, uint increment);

public void AcknowledgeAlarm(AlarmSeverity severity);
public void AcknowledgeAllAlarms();
public void AcknowledgeAllAlarms(AlarmSeverity severity);

public void RemoveAllAcknowledgedAlarms();
public void RemoveAllAcknowledgedAlarms(AlarmSeverity severity);
public void RemoveAllNewAlarms();
public void RemoveAllNewAlarms(AlarmSeverity severity);
```

```
public void SetAcknowledgedAlarmCount(AlarmSeverity severity, uint count);
public void SetNewAlarmCount(AlarmSeverity severity, uint count);

public void Clear();
```

Other functions:

```
public bool IsEmpty();
public bool IsEnablePropagation { get; set; }
```

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:

```
using TWaver;
using TWaver.Network;

namespace TestTWaverSilverlight
{
    public class TestAlarm
    {
        private Network network = new Network();

        public TestAlarm(MainPage mainPage)
        {
            mainPage.LayoutRoot.Children.Add(this.network);
            InitDataBox();
        }

        private void InitDataBox()
        {
            Node node = new Node();
            node.SetLocation(50, 50);
            network.ElementBox.Add(node);

            addAlarm("alarm 1", node.ID, AlarmSeverity.CRITICAL, network.ElementBox.AlarmBox);
        }

        //Generally we add alarm to AlarmBox which is same as adding Element to ElementBox
        private void addAlarm(object alarmID, object elementID,
            AlarmSeverity alarmSeverity, AlarmBox alarmBox)
        {
            Alarm alarm = new Alarm(alarmID, elementID, alarmSeverity);
            alarmBox.Add(alarm);
        }
    }
}
```



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.

```
namespace TWaver
{
    public interface ILayer : IData
    {
        bool IsVisible { get; set; }
        bool IsMovable { get; set; }
        bool IsEditable { get; set; }
    }
}
```

The hierarchy of TWaver .NET is maintained by LayerBox, the default order is decided by parent-child relations and the sequence when adding. Each element belongs to certain layer by setting property LayerID of Element, thus, all network elements in topology will be shown in hierarchy.

Here is an example:

```
<UserControl x:Class="TestTWaverSilverlight.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

    <Grid x:Name="LayoutRoot">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="250"/>
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>

        <Grid x:Name="treePane" Grid.Column="0" Grid.Row="0"/>
        <Grid x:Name="networkPane" Grid.Column="1" Grid.Row="0"/>
    </Grid>
</UserControl>
```

```
using System.Windows.Controls;
using System.Windows.Media;
using TWaver;
using TWaver.Controls;
using TWaver.Network;

namespace TestTWaverSilverlight
{
    public partial class MainPage : UserControl
    {
        private Network network = new Network();
        private Tree<ILayer> tree = new Tree<ILayer>();

        public MainPage()
        {
            InitializeComponent();
            this.networkPane.Children.Add(this.network);
            this.treePane.Children.Add(this.tree);
            InitDataBox();
        }
    }
}
```

```

        this.tree.ExpandAll();
    }

    public void InitDataBox()
    {
        tree.DataBox = this.network.ElementBox.LayerBox;
        network.SetEditInteractionHandlers(false);

        Layer layer1 = new Layer("unmovable", "unmovable layer");
        layer1.IsMovable = false;
        Layer layer2 = new Layer("uneditable", "uneditable layer");
        layer2.IsEditable = false;
        Layer layer3 = new Layer("invisible", "invisible layer");
        layer3.IsVisible = false;





        network.ElementBox.LayerBox.Add(layer1);
        network.ElementBox.LayerBox.Add(layer2);
        network.ElementBox.LayerBox.Add(layer3, 0);

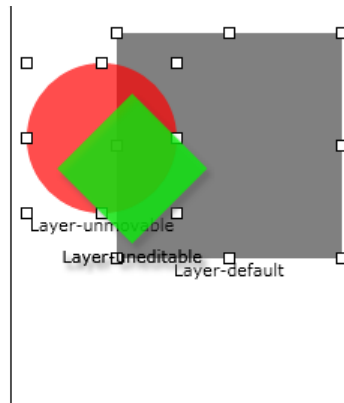
        CreateNode(layer1, Consts.SHAPE_CIRCLE, 10, 40, 100, 100, 0xB2FF0000);
        CreateNode(layer2, Consts.SHAPE_DIAMOND, 30, 60, 100, 100, 0xB200FF00);
        CreateNode(layer3, Consts.SHAPE_RECTANGLE, 50, 80, 100, 100, 0xB20000FF);
        CreateNode(network.ElementBox.LayerBox.DefaultLayer, Consts.SHAPE_RECTANGLE, 70, 20, 150, 150,
0xFF808080);
    }

    private Node CreateNode(ILayer layer, string shape, int x, int y, int width, int height, uint fillColor)
    {
        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_SHAPE, shape);
        node.SetSize(width, height);
        node.SetLocation(x, y);
        node.SetStyle(Styles.VECTOR_FILL_COLOR, ToColor(fillColor));
        network.ElementBox.Add(node);
        return node;
    }

    private static Color ToColor(uint argb)
    {
        byte a = (byte)((argb & -16777216) >> 0x18);
        byte r = (byte)((argb & 0xff0000) >> 0x10);
        byte g = (byte)((argb & 0xff00) >> 8);
        byte b = (byte)(argb & 0xff);
        return Color.FromArgb(a, r, g, b);
    }
}

```

-  invisible layer
-  default
-  unmovable layer
-  uneditable layer



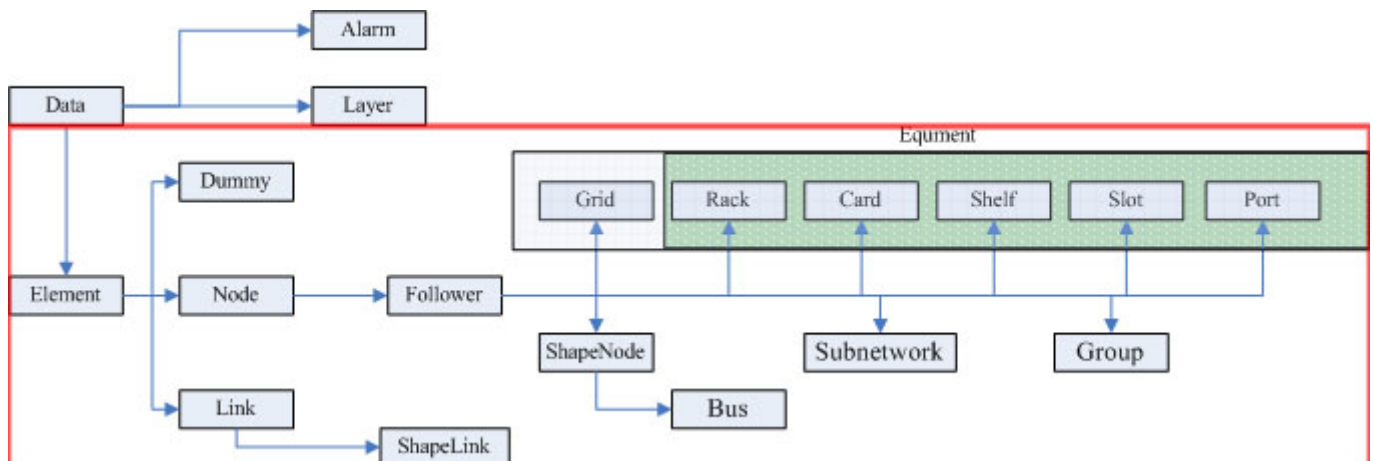
Topology Element

Interface `IElement` is to define the network element in topology, It is the most important data element in TWaver .NET. 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.

```
public interface IElement : IData, IStyle
{
    AlarmState AlarmState { get; }
    object LayerID { get; set; }
    bool IsAdjustedToBottom();
    Type ElementUIClass { get; }
}
```

Thereinto `elementUIClass` is the `ElementUI` class or class 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` extends `Panel`. It is .NET 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.

```
public interface IStyle
{
    IStyle SetStyle(string styleProp, object newValue);
    object GetStyle(string styleProp);
    object GetStyle(string styleProp, bool returnDefaultIfNull);
    TValue GetStyle<TValue>(string styleProp);
}
```



```
TValue GetStyle<TValue>(string styleProp, bool returnDefaultIfNull);
}
```

Set label color to be red for network element:

```
Node node = new Node();
node.SetStyle(Styles.LABEL_COLOR, Colors.Red);
node.Name = "Node A";
```

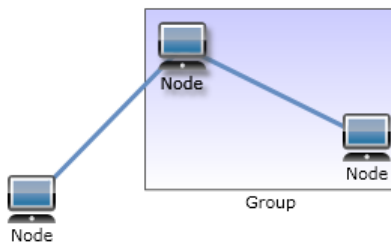
Here is the result:



Otherwise, Element defined bool `IsAdjustedToBottom()` method to represent whether view in bottom, the default value is false.

```
public virtual bool IsAdjustedToBottom()
```

`TWaver.Group` overwrite this method, and the return value is true which means 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.

```
public static void RegisterImage(string name, BitmapImage bitmapImage,
    double width, double height)
{
}
```

TWaver.Node set picture:

```
public string Image { get; set; }
```

For example:

```
using System;
using System.Windows.Controls;
using System.Windows.Media.Imaging;
using TWaver;

namespace TestTWaverSilverlight
{
    public partial class MainPage : UserControl
    {
        public MainPage()
        {
            InitializeComponent();
            InitDataBox();
        }

        private void InitDataBox()
        {
            RegisterImage("router", 73, 13);
            RegisterImage("pc", 28, 32);

            Node route = new Node();
            route.Location = new System.Windows.Point(50, 50);
            route.Image = "router";
            route.Name = "Route";
            network.ElementBox.Add(route);

            Node pc = new Node();
            pc.Location = new System.Windows.Point(200, 200);
            pc.Image = "pc";
            pc.Name = "PC";
            network.ElementBox.Add(pc);

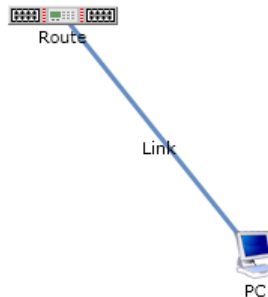
            Link link = new Link(route, pc);
            link.Name = "Link";
            network.ElementBox.Add(link);
        }

        private void RegisterImage(string name, double width, double height)
        {
            string uri = "Images/" + name + ".png";
```

```

        BitmapImage bitmapImage = new BitmapImage(new Uri(uri,
            UriKind.RelativeOrAbsolute));
        Utils.RegisterImage(name, bitmapImage, width, height);
    }
}

```



Related properties and methods for the location and size of node

```

public Point CenterLocation { get; set; }
public virtual double Height { get; set; }
public IList<Link> Links { get; }
public virtual Point Location { get; set; }
public Rect Rect { get; }
public Size Size { get; set; }
public virtual double Width { get; set; }
public double X { get; }
public double Y { get; }
public void SetCenterLocation(double x, double y);
public void SetLocation(double x, double y);
public void SetSize(double width, double height);
public void Translate(double dx, double dy);

```

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

```

public IList<Link> ToAgentLinks { get; }
public IList<Link> ToLinks { get; }
public IList<Link> LoopedLinks { get; }
public IList<Follower> Followers { get; }
public IList<Link> FromAgentLinks { get; }
public IList<Link> FromLinks { get; }
public IList<Link> AgentLinks { get; }

```

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

```

public IList<Follower> Followers { get; }

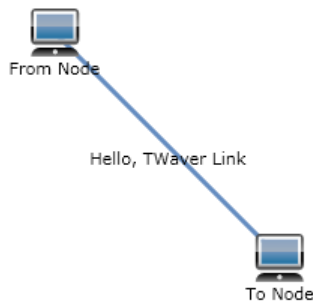
```

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 Node FromNode { get; set; }
public Node ToNode { get; set; }
```



Start Node Agent and End Node Agent

The node connecting with link directly named fromNode, toNode in TWaver .NET. 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 Node FromAgent { get; }
public Node ToAgent { get; }
```

For example:

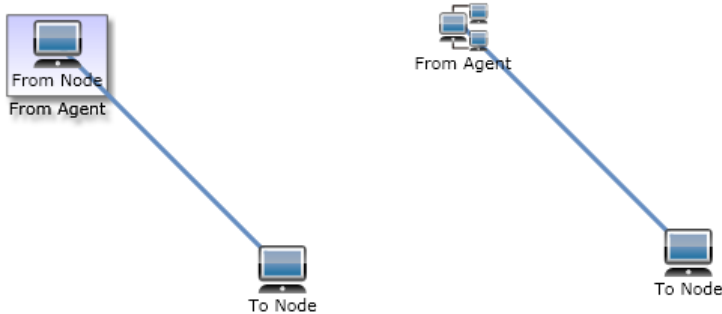
```
Node from = new Node();
from.Location = new System.Windows.Point(50, 50);
from.Name = "From Node";
network.ElementBox.Add(from);

Group fromGroup = new Group();
fromGroup.Location = new System.Windows.Point(50, 50);
fromGroup.Name = "From Agent";
fromGroup.AddChild(from);
network.ElementBox.Add(fromGroup);

Node to = new Node();
to.Location = new System.Windows.Point(200, 200);
to.Name = "To Node";
network.ElementBox.Add(to);

Link link = new Link(from, to);
```

```
network.ElementBox.Add(link);
```



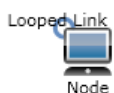
Self-Loop, The Start and End Point of This Link Share the Same Node

```
public bool IsLooped();
```

For example:

```
Node node = new Node();
node.Location = new System.Windows.Point(50, 50);
node.Name = "Node";
network.ElementBox.Add(node);

Link link = new Link(node, node);
link.Name = "Looped Link";
link.SetStyle(Styles.LABEL_POSITION, Consts.POSITION_TOP);
network.ElementBox.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.

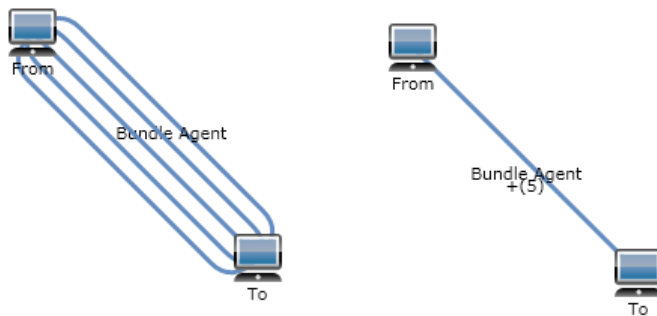
```
public int BundleCount { get; }
public int BundleIndex { get; }
public BundleLinks BundleLinks { get; internal set; }
public bool IsBundleAgent();
public bool ReverseBundleExpanded();
```

For example:

```
Node from = new Node();
from.Location = new System.Windows.Point(50, 50);
from.Name = "From";
network.ElementBox.Add(from);

Node to = new Node();
to.Location = new System.Windows.Point(200, 200);
to.Name = "To";
network.ElementBox.Add(to);

Link link = new Link(from, to);
link.Name = "Bundle Agent";
network.ElementBox.Add(link);
network.ElementBox.Add(new Link(from, to));
network.ElementBox.Add(new Link(from, to));
network.ElementBox.Add(new Link(from, to));
network.ElementBox.Add(new Link(from, to));
```



- [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, binding 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 binding, 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 bundle links with ID "1" and "2"

```

using System.Windows.Controls;
using System.Windows.Media;
using TWaver;

namespace TestTWaverSilverlight
{
    public partial class MainPage : UserControl
    {
        public MainPage()
        {
            InitializeComponent();
            InitDataBox();
        }

        private void InitDataBox()
        {
            Node from = new Node();
            from.Location = new System.Windows.Point(50, 50);
            from.Name = "From";
            network.ElementBox.Add(from);

            Node to = new Node();
            to.Location = new System.Windows.Point(200, 200);
            to.Name = "To";
            network.ElementBox.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, Colors.Red);
            CreateLink(from, to, "g2_2", 2, Colors.Red);
            CreateLink(from, to, "g2_3", 2, Colors.Red);
        }

        private Link CreateLink(Node from, Node to, string name, int groupID)
        {
            return CreateLink(from, to, name, groupID, null);
        }
    }
}
    
```

```

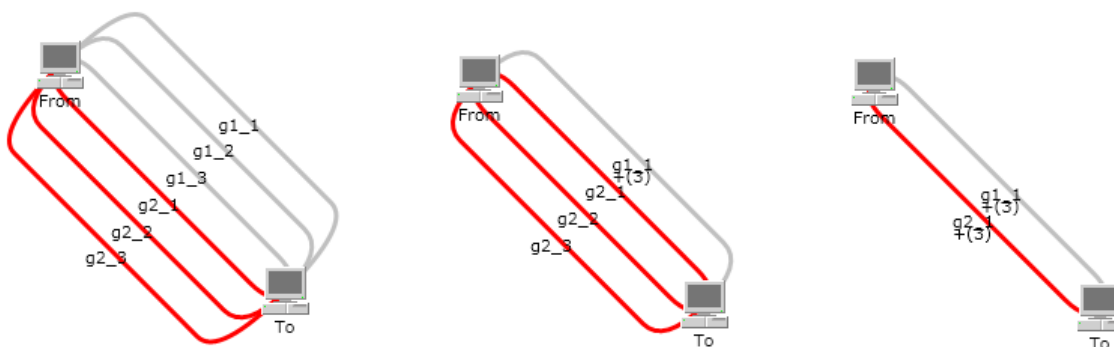
private Link CreateLink(Node from, Node to, string name, int groupID, Color? color)
{
    return CreateLink(from, to, name, groupID, color, null, false, 25, -1, true);
}

private Link CreateLink(Node from, Node to, string name, int groupID, Color? color,
    string type, bool groupIndependent, int gap, int offset, bool bundleEnable)
{
    Link link = new Link(from, to);
    link.Name = name;
    if (type != null)
    {
        link.SetStyle(Styles.LINK_TYPE, type);
    }
    if (color != null)
    {
        link.SetStyle(Styles.LINK_COLOR, color.Value);
    }
    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);
    network.ElementBox.Add(link);

    return link;
}
}

```

Run application as follows:



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

```

public void InitDataBox()
{
    Node from = new Node();
    from.Location = new System.Windows.Point(50, 50);
    from.Name = "From";
    network.ElementBox.Add(from);
}

```



```

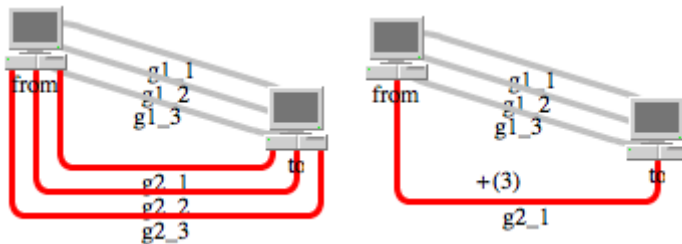
Node to = new Node();
to.Location = new System.Windows.Point(200, 200);
to.Name = "To";
network.ElementBox.Add(to);

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

CreateLink(from, to, "g2_1", 2, Colors.Red, Consts.LINK_TYPE_EXTEND_BOTTOM, true);
CreateLink(from, to, "g2_2", 2, Colors.Red, Consts.LINK_TYPE_EXTEND_BOTTOM, true);
CreateLink(from, to, "g2_3", 2, Colors.Red, 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.

```

public void InitDataBox()
{
    Node from = new Node();
    from.Location = new System.Windows.Point(50, 50);
    from.Name = "From";
    network.ElementBox.Add(from);

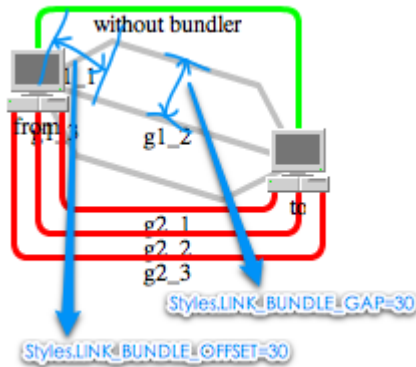
    Node to = new Node();
    to.Location = new System.Windows.Point(200, 200);
    to.Name = "To";
    network.ElementBox.Add(to);

    int gap = 30;
    int offset = 30;
    CreateLink(from, to, "g1_1", 1, null, Consts.LINK_TYPE_TRIANGLE, false, gap, offset);
    CreateLink(from, to, "g1_2", 1, null, Consts.LINK_TYPE_TRIANGLE, false, gap, offset);
    CreateLink(from, to, "g1_3", 1, null, Consts.LINK_TYPE_TRIANGLE, false, gap, offset);

    CreateLink(from, to, "g2_1", 2, Colors.Red, Consts.LINK_TYPE_EXTEND_BOTTOM, true);
    CreateLink(from, to, "g2_2", 2, Colors.Red, Consts.LINK_TYPE_EXTEND_BOTTOM, true);
    CreateLink(from, to, "g2_3", 2, Colors.Red, Consts.LINK_TYPE_EXTEND_BOTTOM, true);
}

```

Run as follows:



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

```
public void InitDataBox()
{
    Node from = new Node();
    from.Location = new System.Windows.Point(50, 50);
    from.Name = "From";
    network.ElementBox.Add(from);

    Node to = new Node();
    to.Location = new System.Windows.Point(200, 200);
    to.Name = "To";
    network.ElementBox.Add(to);

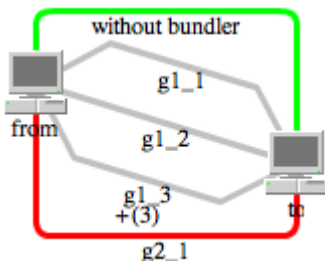
    int gap = 30;
    int offset = 30;
    CreateLink(from, to, "g1_1", 1, null, Consts.LINK_TYPE_TRIANGLE, false, gap, offset);
    CreateLink(from, to, "g1_2", 1, null, Consts.LINK_TYPE_TRIANGLE, false, gap, offset);
    CreateLink(from, to, "g1_3", 1, null, Consts.LINK_TYPE_TRIANGLE, false, gap, offset);

    Link link1 = CreateLink(from, to, "g2_1", 2, Colors.Red, Consts.LINK_TYPE_EXTEND_BOTTOM, true);
    Link link2 = CreateLink(from, to, "g2_2", 2, Colors.Red, Consts.LINK_TYPE_EXTEND_BOTTOM, true);
    Link link3 = CreateLink(from, to, "g2_3", 2, Colors.Red, Consts.LINK_TYPE_EXTEND_BOTTOM, true);

    CreateLink(from, to, "without bundler", -1, Colors.Green, Consts.LINK_TYPE_EXTEND_TOP, false, 0, 0, false);

    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 .NET 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

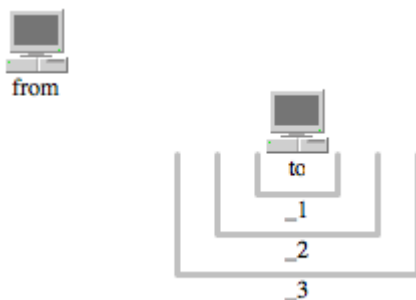
```

public void InitDataBox()
{
    Node from = new Node();
    from.Location = new System.Windows.Point(50, 50);
    from.Name = "From";
    network.ElementBox.Add(from);

    Node to = new Node();
    to.Location = new System.Windows.Point(200, 200);
    to.Name = "To";
    network.ElementBox.Add(to);

    int i = 3;
    while (i > 0)
    {
        Link link = CreateLink(to, to, "_" + i, -1);
        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 .NET has two ways:

One of is TWaver.Link which decided the links direction by setting 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 const string LINK_TYPE_ARC = "arc";
public const string LINK_TYPE_TRIANGLE = "triangle";
public const string LINK_TYPE_PARALLEL = "parallel";

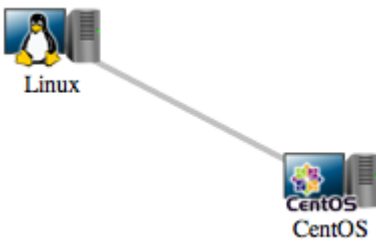
public const string LINK_TYPE_FLEXIONAL = "flexional";
public const string LINK_TYPE_FLEXIONAL_HORIZONTAL = "flexional.horizontal";
public const string LINK_TYPE_FLEXIONAL_VERTICAL = "flexional.vertical";

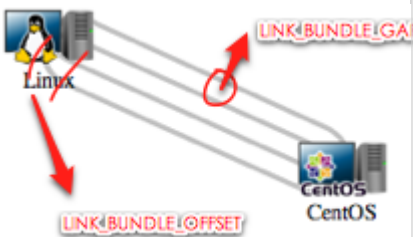
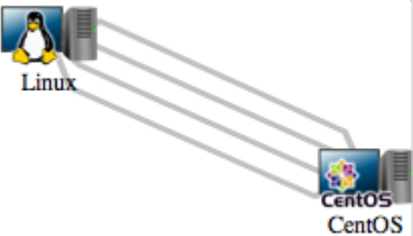
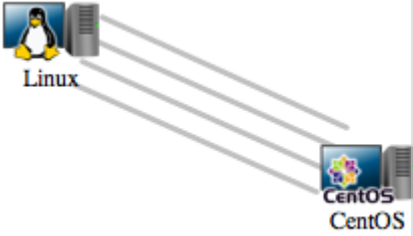

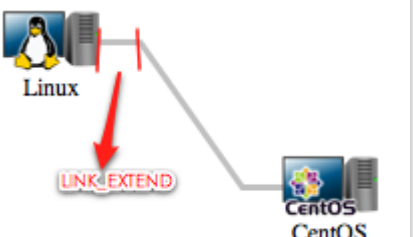

public const string LINK_TYPE_ORTHOGONAL = "orthogonal";
public const string LINK_TYPE_ORTHOGONAL_HORIZONTAL = "orthogonal.horizontal";
public const string LINK_TYPE_ORTHOGONAL_VERTICAL = "orthogonal.vertical";
public const string LINK_TYPE_HORIZONTAL_VERTICAL = "orthogonal.H.V";
public const string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

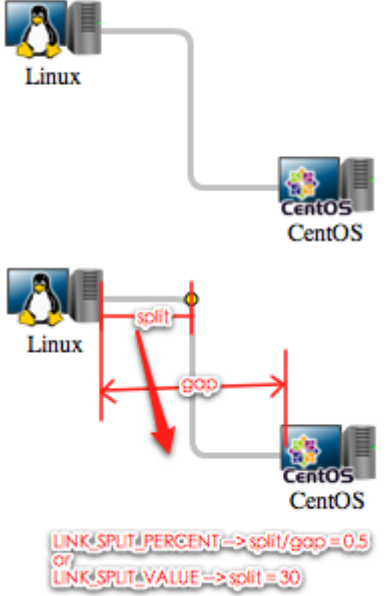
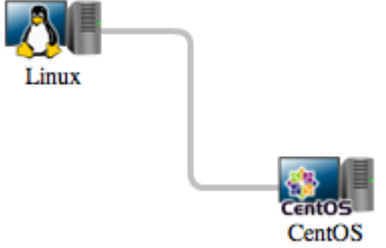


Set link types

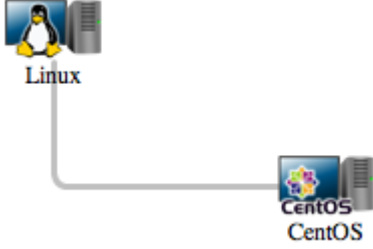
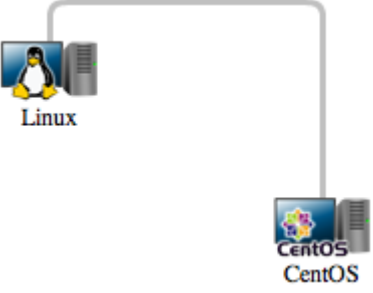
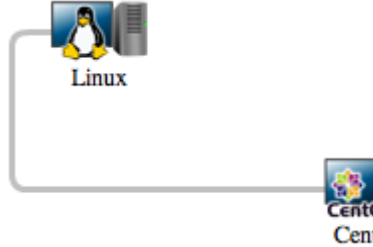
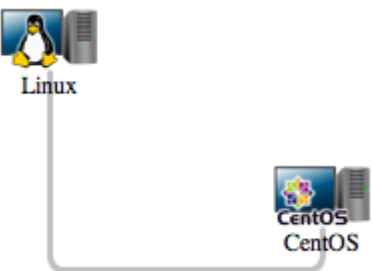

```
link.setStyle(Styles.LINK_TYPE, Consts.LINK_TYPE_ORTHOGONAL);
```

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

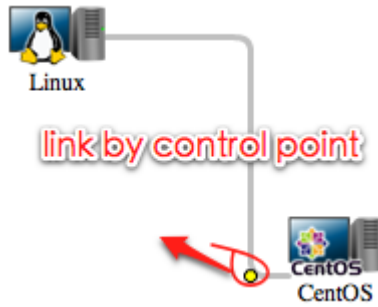
Link Type	View	Control Parameters
<p>Basic Type is to connect start and end node directly.</p> <p>When there are more than one links between two nodes, TWaver classify three types by</p> <p>links expanding effects</p> <p>LINK_TYPE_ARC</p> <p>LINK_TYPE_TRIANGLE</p> <p>LINK_TYPE_PARALLEL</p> <p>These types 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>
<p>LINK_TYPE_ORTHOGONAL_HORIZONTAL</p>		<p>The same as above</p>
<p>LINK_TYPE_ORTHOGONAL_VERTICAL</p>		<p>The same as above</p>
<p>LINK_TYPE_HORIZONTAL_VERTICAL From start point, align horizontal direction at first then 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</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. The follower will move together with host when host is moving.

Set Host Node

One node can be attached more than one followers

```
//get host node
public Node Host { get; set; }
//Whether attach itself to certain node
public bool IsHostOn(Node node);
```

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 bool IsLoopedHostOn(Follower follower);
```

For example:

```
<UserControl x:Class="TestTWaverSilverlight.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  >

  <Grid x:Name="LayoutRoot" Background="White">
    </Grid>
  </UserControl>
```

```
using System.Windows.Controls;
using TWaver.Network;

namespace TestTWaverSilverlight
{
    public partial class MainPage : UserControl
    {
        private Network network = new Network();

        public MainPage()
        {
            InitializeComponent();
            LayoutRoot.Children.Add(network);
            new TestFollower(network).InitDataBox();
        }
    }
}
```

}

```
using System.Windows.Media;
using TWaver;
using TWaver.Network;

namespace TestTWaverSilverlight
{
    public class TestFollower
    {
        private Network network = null;

        public TestFollower(Network network)
        {
            this.network = network;
        }

        public void InitDataBox()
        {
            Node host = new Node();
            host.Location = new System.Windows.Point(50, 50);
            host.Name = "Host";
            network.ElementBox.Add(host);

            Follower follower = new Follower();
            follower.Location = new System.Windows.Point(200, 200);
            follower.Name = "Follower";
            follower.Host = host;
            network.ElementBox.Add(follower);
        }
    }
}
```



TWaver.ISubNetwork

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

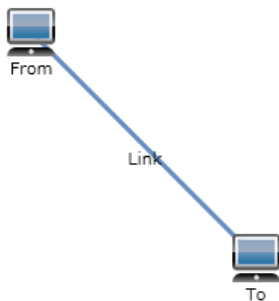
TWaver.SubNetwork is the implement class of twaver.ISubNetwork. In topology double click can enter into SubNetwork, double click backgroud can go back upper SubNetwork. The Null value of current SubNetwork stands for it is the highest SubNetwork.

Switch SubNetwork in Network:

```
//Get/Set current SubNetwork
public ISubNetwork CurrentSubNetwork { get; set; }
public void SetCurrentSubNetwork(ISubNetwork currentSubNetwork, bool animate, Action finishFunction);
//Go back to upper SubNetwork
public void UpSubNetwork(bool animate, Action finishFunction);
```



Double click to go into SubNetwork:



For example:

```
using System.Windows.Media;
using TWaver;
using TWaver.Network;

namespace TestTWaverSilverlight
{
    public class TestSubnetwork
    {
        private Network network = null;

        public TestSubnetwork(Network network)
        {
            this.network = network;
        }
    }
}
```

```
public void InitDataBox()
{
    SubNetwork subNetwork = new SubNetwork();
    subNetwork.Location = new System.Windows.Point(100, 100);
    subNetwork.Name = "SubNetwork";
    network.ElementBox.Add(subNetwork);

    Node from = new Node();
    from.Location = new System.Windows.Point(50, 50);
    from.Name = "From";
    from.Parent = subNetwork;
    network.ElementBox.Add(from);

    Node to = new Node();
    to.Location = new System.Windows.Point(200, 200);
    to.Name = "To";
    to.Parent = subNetwork;
    network.ElementBox.Add(to);

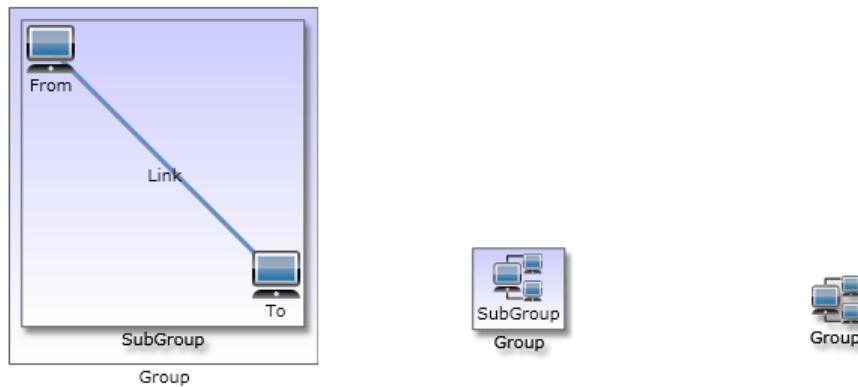
    Link link = new Link(from, to);
    link.Name = "Link";
    link.Parent = subNetwork;
    network.ElementBox.Add(link);
}
}
```

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:

```
using System.Windows.Media;
using TWaver;
using TWaver.Network;

namespace TestTWaverSilverlight
{
    public class TestGroup
    {
        private Network network = null;

        public TestGroup(Network network)
        {
            this.network = network;
        }

        public void InitDataBox()
        {
            Group group = new Group();
            group.Location = new System.Windows.Point(100, 100);
            group.Name = "Group";
            network.ElementBox.Add(group);

            Group subGroup = new Group();
            subGroup.Location = new System.Windows.Point(100, 100);
            subGroup.Name = "SubGroup";
            subGroup.Parent = group;
            network.ElementBox.Add(subGroup);

            Node from = new Node();
            from.Location = new System.Windows.Point(50, 50);
            from.Name = "From";
            from.Parent = subGroup;
            network.ElementBox.Add(from);
        }
    }
}
```

```
Node to = new Node();
to.Location = new System.Windows.Point(200, 200);
to.Name = "To";
to.Parent = subGroup;
network.ElementBox.Add(to);

Link link = new Link(from, to);
link.Name = "Link";
link.Parent = subGroup;
network.ElementBox.Add(link);
    }
}
}
```

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 IList<Point> Points { get; set; }
public void AddPoint(Point point);
public void AddPointAt(Point point, int index);
public void RemovePoint(Point point);
public void RemovePointAt(int index);
public void SetPointAt(Point point, int index);
```

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

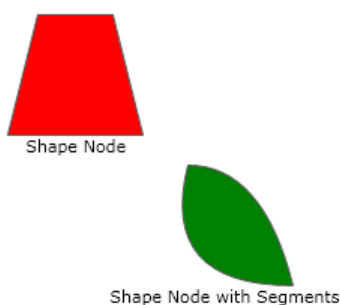
TWaver has three drawing methods: moveTo, lineTo, QuadTo


```
public const string SEGMENT_CUBICTO = "cubicto";
public const string SEGMENT_LINETO = "lineTo";
public const string SEGMENT_MOVETO = "moveTo";
public const string SEGMENT_QUADTO = "quadTo";
```

Set segments

```
public IList<string> Segments { get; set; }
```

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.



 Note: One curve segment needs two control points

The whole program of this application:

```
using System.Windows.Media;
using TWaver;
using TWaver.Network;
using System.Windows;
```

```

using System.Collections.Generic;

namespace TestTWaverSilverlight
{
    public class TestShapNode
    {
        private Network network = null;

        public TestShapNode(Network network)
        {
            this.network = network;
        }

        public void InitDataBox()
        {
            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, Colors.Red);
            shapeNode.Name = "Shape Node";
            shapeNode.SetLocation(100, 100);
            network.ElementBox.Add(shapeNode);

            ShapeNode shapeNodeWithSegment = new ShapeNode();
            shapeNodeWithSegment.AddPoint(new Point(30, 10));
            shapeNodeWithSegment.AddPoint(new Point(80, 10));
            shapeNodeWithSegment.AddPoint(new Point(100, 90));
            shapeNodeWithSegment.AddPoint(new Point(10, 90));
            shapeNodeWithSegment.AddPoint(new Point(30, 10));
            shapeNodeWithSegment.SetStyle(Styles.VECTOR_FILL_COLOR, Colors.Green);
            shapeNodeWithSegment.SetLocation(200, 200);

            IList<string> segments = new List<string>();
            segments.Add(Consts.SEGMENT_MOVETO);
            segments.Add(Consts.SEGMENT_QUADTO);
            segments.Add(Consts.SEGMENT_QUADTO);
            shapeNodeWithSegment.Segments = segments;
            shapeNodeWithSegment.Name = "Shape Node with Segments";
            network.ElementBox.Add(shapeNodeWithSegment);
        }
    }
}

```


TWaver.Grid

TWaver.Grid network element present as grid in topology, which is same as Grid control of .NET. 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.

!grid.png!For example:

```
using TWaver.Network;
using TWaver;
using System.Windows.Media;
namespace TestTWaverSilverlight
{
    public class TestGrid
    {
        private Network network = null;

        public TestGrid(Network network)
        {
            this.network = network;
        }

        public void InitDataBox()
        {
            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, new double[] { 0.2, 0.6, 0.2 });
            grid.SetStyle(Styles.GRID_DEEP, 8);
            network.ElementBox.Add(grid);

            Grid cell = 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, Colors.Red);
            cell.SetStyle(Styles.FOLLOWER_PADDING, 10);
            cell.SetStyle(Styles.GRID_DEEP, 5);
            cell.Host = grid;
            grid.AddChild(cell);
            network.ElementBox.Add(cell);
        }
    }
}
```

Data Container

- [Basic Data Container](#)
- [Alarm Model](#)
- [QuickFinder](#)

Basic Data Container

TWaver.DataBox is the base class of other data container AlarmBox, LayerBox and ElementBox. It's the model layer of MVP model. It maintains datas, dispatch event when data is changed and takes charge of selection by using SelectionModel.

Properties

```
//The icon showed for the root node on the tree, the default value is Defaults.ICON_DATABOX
public virtual string Icon { get; set; }
//The max count of items of DataBox, the default value is -1 which means no limit
public int Limit { get; set; }
//The name showed for the root node on the tree.
public virtual string Name { get; set; }
//The tooltip showed for the root node on the tree.
public virtual string ToolTip { get; set; }
```

Maintains Datas

1. Methods starting with "Add" are used to add data to DataBox, and event DataBoxChange with type DataBoxChange.ADD is dispatched when they are invoked.
2. Methods starting with "Remove" are used to remove data from DataBox, and event DataBoxChange with type DataBoxChange.REMOVE is dispatched when they are invoked.
3. Method "Clear" is used to clear DataBox, and event DataBoxChange with type DataBoxChange.CLEAR is dispatched when it's invoked.


```
public virtual void Add(TData data);
public virtual void Add(TData data, int index);
public virtual void Clear();
public virtual void Remove(TData data);
public virtual void RemoveByID(object id);
public void RemoveFirst();
public void RemoveFirst(int count);
public void RemoveSelection();
```

SelectionModel

```
public SelectionModel<TData> SelectionModel { get; }
```

SelectionModel is used to maintain which datas are selected in the DataBox.

1. SelectionMode: there are three type of SelectionMode, the default is MULTIPLE_SELECTION
SelectionModel.NONE_SELECTION : no data can be selected.
SelectionModel.SINGLE_SELECTION : only one data can be selected.
SelectionModel.MULTIPLE_SELECTION : more than one data can be selected.

 Note: when the SelectionModel is changed, TWaver will invoke clear method too.

2. Predicate<TData> filterFunction, it's used to filter which data can be selected.
3. EventListener<SelectionChangeEvent<TData>> SelectionChange : there are five type of Event SelectionChangeEvent, and property Datas of SelectionChangeEvent represent the data of selection changed.
SelectionChangeEvent.ALL : method SelectAll dispatchs this event.
SelectionChangeEvent.APPEND : method AppendSelection dispatchs this event.

SelectionChangeEvent.CLEAR : method ClearSelection dispatchs this event.
 SelectionChangeEvent.REMOVE : method RemoveSelection dispatchs this event.
 SelectionChangeEvent.SET : method SetSelection dispatchs this event.

Events

```
public event EventListener<DataBoxChangeEvent<TData>> DataBoxChange;
public event EventListener<PropertyChangeEvent<TData>> DataPropertyChange;
public event EventListener<HierarchyChangeEvent<TData>> HierarchyChange;
public event EventListener<PropertyChangeEvent<DataBox<TData>>> PropertyChange;
```

Event DataBoxChange

1. Event DataBoxChange.ADD is dispatched when data is added to DataBox, property Datas of DataBoxChange is null, property Data is the data is added.
2. Event DataBoxChange.REMOVE is dispatched when data is removed from DataBox, property Datas of DataBoxChange is null, property Data is the data is removed.
3. Event DataBoxChange.CLEAR is dispatched when DataBox is cleared, property Datas of DataBoxChange is the datas is removed, property Data is null.

Event DataPropertyChange

DataBox add listener to event PropertyChange of data when data is added to DataBox, you can use DataPropertyChange to listen change of property of any data in the DataBox.

Event HierarchyChange

Event HierarchyChangeEvent is dispatch the method start with "Move" is invoked.

Event PropertyChange

Event PropertyChangeEvent is dispatched when property Icon, Limit, Name, ToolTip or propertis of DataBox is changed.

Example:

```
public static void Main(string[] args)
{
    DataBox<IData> box = new DataBox<IData>("TestDataBox");

    box.DataBoxChange += (DataBoxChangeEvent<IData> evt) =>
    {
        System.Diagnostics.Debug.WriteLine("DataBoxChange:\t" + "Type:"
            + evt.Kind + "\tData:" + evt.Data + "\tDatas:" + ToString(evt.Datas));
    };

    box.DataPropertyChange += (PropertyChangeEvent<IData> evt) =>
    {
        System.Diagnostics.Debug.WriteLine("DataPropertyChange:\t"
            + "PropertyName:" + evt.PropertyName + "\tSource:" + evt.Source
            + "\tOldValue:" + evt.OldValue + "\tNewValue:" + evt.NewValue);
    };

    box.HierarchyChange += (HierarchyChangeEvent<IData> evt) =>
    {
        System.Diagnostics.Debug.WriteLine("HierarchyChange:\t" + "Data:" + evt.Data
            + "\tOldIndex:" + evt.OldIndex + "\tNewIndex:" + evt.NewIndex);
    };
}
```

```

box.PropertyChange += (PropertyChangeEvent<DataBox<IData>> evt) =>
{
    System.Diagnostics.Debug.WriteLine("PropertyChange:\t"
        + "PropertyName:" + evt.PropertyName + "\tSource:" + evt.Source
        + "\tOldValue:" + evt.OldValue + "\tNewValue:" + evt.NewValue);
};

box.SelectionModel.SelectionChange += (SelectionChangeEvent<IData> evt) =>
{
    System.Diagnostics.Debug.WriteLine("SelectionChange:\t"
        + "Kind:" + evt.Kind + "\tDatas:" + ToString(evt.Datas));
};

for (int i = 0; i < 5; i++)
{
    Data data = new Data("Data" + i);
    data.Name = data.ID.ToString();
    box.Add(data);
}

box.RemoveByID("Data1");

IData data2 = box.GetDataByID("Data2");
data2.SetClient("Client1", "Value1");

box.MoveDown(data2);

box.Name = "New Name";

box.SelectionModel.SetSelection(data2);

box.SelectionModel.AppendSelection(box.GetDataByID("Data3"));

box.SelectionModel.SelectAll();

box.SelectionModel.RemoveSelection(data2);

box.SelectionModel.ClearSelection();

box.SelectionModel.SelectionMode = SelectionModel<IData>.NONE_SELECTION;

box.SelectionModel.AppendSelection(data2);

System.Diagnostics.Debug.WriteLine(box.SelectionModel.Count);

box.Clear();
}

private static string ToString(ICollection<IData> datas)
{
    StringBuilder builder = new StringBuilder();
    if (datas != null)
    {
        foreach (IData data in datas)
        {
            builder.Append(" " + data);
        }
    }
    return builder.ToString();
}

```

Output:

```

DataBoxChange: Type:add      Data:Data0      Datas:
DataBoxChange: Type:add      Data:Data1      Datas:
DataBoxChange: Type:add      Data:Data2      Datas:
DataBoxChange: Type:add      Data:Data3      Datas:
DataBoxChange: Type:add      Data:Data4      Datas:
DataBoxChange: Type:remove   Data:Data1      Datas:
DataPropertyChange: PropertyName:C:Client1 Source:Data2 OldValue:      NewValue:Value1
HierarchyChange:      Data:Data2      OldIndex:1      NewIndex:2
PropertyChange: PropertyName:Name      Source:New Name OldValue:TestDataBox      NewValue:New Name
SelectionChange:      Kind:set      Datas: Data2
SelectionChange:      Kind:append   Datas: Data3
SelectionChange:      Kind:all      Datas: Data0 Data4
SelectionChange:      Kind:remove   Datas: Data2
SelectionChange:      Kind:clear    Datas: Data0 Data3 Data4
0
DataBoxChange: Type:clear    Data:      Datas: Data0 Data2 Data3 Data4

```

Alarm Model

AlarmBox is a property of ElementBox, and is used to maintain Alarm. An Alarm can link to more than one Element, and a Element can contain more than one Alarm. The relationship between Alarm and Element is maintained by Property AlarmElementMapping of AlarmBox.

AlarmElementMapping

```
public IAlarmElementMapping AlarmElementMapping { get; set; }
```

The followin code is the define of Interface IAlarmElementMapping:

```
public interface IAlarmElementMapping
{
    IList<IAlarm> GetCorrespondingAlarms(IElement element);
    IList<IElement> GetCorrespondingElements(IAlarm alarm);
}
```

The default implements of interface IAlarmElementMapping is AlarmElementMapping, it use property ElementID of Alarm to map Alarm and Element.

```
public IList<IAlarm> GetCorrespondingAlarms(IElement element)
{
    return this.alarmsFinder.Find(element.ID);
}

public IList<IElement> GetCorrespondingElements(IAlarm alarm)
{
    IElement element = this.elementBox.GetDataByID(alarm.ElementID);
    return element == null ? EMPTY_LIST : new IElement[] { element };
}
```

Example

```
public class TestAlarmElementMapping
{
    private Network network = new Network();

    public TestAlarmElementMapping(MainPage mainPage)
    {
        mainPage.LayoutRoot.Children.Add(this.network);
        InitDataBox();
    }

    private void InitDataBox()
    {
        network.ElementBox.AlarmBox.AlarmElementMapping = new
        CustomAlarmElementMapping(network.ElementBox);
        this.CreateData("Geography Group", 250, 60);
        this.CreateData("Business Group", 60, 160);
        this.CreateData("Manufacturer Group", 430, 160);
    }
}
```

```

        for (int i = 1; i <= 5; i++)
        {
            Alarm alarm = new Alarm();
            alarm.AlarmSeverity = AlarmSeverity.Severities[i];
            alarm.SetClient(CustomAlarmElementMapping.MAPPINGID, i);
            network.ElementBox.AlarmBox.Add(alarm);
        }
        for (int i = 1; i <= 5; i++)
        {
            Alarm alarm = new Alarm();
            alarm.AlarmSeverity = AlarmSeverity.Severities[i];
            alarm.IsAcked = true;
            alarm.SetClient(CustomAlarmElementMapping.MAPPINGID, 5 - i + 1);
            network.ElementBox.AlarmBox.Add(alarm);
        }
    }

    private void CreateData(string name, int x, int y)
    {
        IDummy dummy = new Dummy(name);
        dummy.Name = name;
        network.ElementBox.Add(dummy);

        for (int i = 1; i <= 5; i++)
        {
            Node node = new Node();
            if (name == "Geography Group")
            {
                node.SetLocation(i * 100, 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_COLOR, Utils.CreateColor(0x3FFFFFFF));
                node.SetStyle(Styles.INNER_PADDING, -6);
                node.SetStyle(Styles.INNER_BACK, false);
            }
            else if (name == "Business Group")
            {
                node.SetLocation(i * 100, 150);
                node.SetStyle(Styles.INNER_STYLE, Consts.INNER_STYLE_SHAPE);
                node.SetStyle(Styles.INNER_SHAPE, Consts.SHAPE_DIAMOND);
                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);
            }
            else if (name == "Manufacturer Group")
            {
                node.SetLocation(i * 100, 250);
            }
            node.Name = "BSC_" + i;
            node.Parent = dummy;
            node.SetClient(CustomAlarmElementMapping.MAPPINGID, i);
            network.ElementBox.Add(node);
        }
    }

    public class CustomAlarmElementMapping : IAlarmElementMapping
    {
        public const string MAPPINGID = "MAPPINGID";

        private QuickFinder<IElement> elementFinder = null;
        private QuickFinder<IAlarm> alarmFinder = null;

        public CustomAlarmElementMapping(ElementBox box)
    }

```



```

{
    this.elementFinder = new QuickFinder<IElement>(box, MAPPINGID, Consts.PROPERTY_TYPE_CLIENT);
    this.alarmFinder = new QuickFinder<IAlarm>(box.AlarmBox, MAPPINGID, Consts.PROPERTY_TYPE_CLIENT);
}

public IList<IAlarm> GetCorrespondingAlarms(IElement element)
{
    return this.alarmFinder.Find(element.GetClient(MAPPINGID));
}

public IList<IElement> GetCorrespondingElements(IAlarm alarm)
{
    return this.elementFinder.Find(alarm.GetClient(MAPPINGID));
}

public void Dispose()
{
    this.alarmFinder.Dispose();
    this.elementFinder.Dispose();
    this.alarmFinder = null;
    this.elementFinder = null;
}
}

```

Result:



QuickFinder

QuickFinder is used to find data from DataBox by any feature of data. You can find data by three way.

1. By property: set PropertyType to Consts.PROPERTY_TYPE_ACCESSOR, and this is the default, the PropertyName can be "Name", "Icon", "ToolTip", etc.
2. By client: set PropertyType to Consts.PROPERTY_TYPE_CLIENT, the PropertyName is the parameter clientProp when invoking method SetClient of data
3. By style: set PropertyType to Consts.PROPERTY_TYPE_STYLE, the PropertyName is the parameter styleProp when invoking method SetStyle of data

Constructor

```
public QuickFinder(DataBox<TData> dataBox, string propertyName);
public QuickFinder(DataBox<TData> dataBox, string propertyName, string propertyType);
public QuickFinder(DataBox<TData> dataBox, string propertyName, string propertyType,
    Func<TData, object> valueFunction, Predicate<TData> filterFunction);
```

dataBox : the target data container to look up

propertyName : its meaning is different according to parameter propertyType

propertyType : there are three valid value: Consts.PROPERTY_TYPE_ACCESSOR, Consts.PROPERTY_TYPE_CLIENT, Consts.PROPERTY_TYPE_STYLE

valueFunction : if it's null, TWaver will generate a default one, it's used to find data

filterFunction : it's used to determine which data will be ignore when finding

Finde data

```
//Find all datas matched
public IList<TData> Find(object value);
//Find first one matched
public TData FindFirst(object value);
```

The parameter value is the search condition, and is the value related to PropertyName.

Sample

```
public static void Main(string[] args)
{
    DataBox<Node> box = new DataBox<Node>();

    for (int i = 0; i < 5; i++)
    {
        Node node = new Node("Node" + i);
        node.Name = "Name" + i;
        box.Add(node);
    }

    box.GetDataByID("Node2").SetClient("Client1", "Value1");
    box.GetDataByID("Node3").SetClient("Client1", "Value1");
    box.GetDataByID("Node0").SetStyle(Styles.LABEL_SIZE, 13.0);
    box.GetDataByID("Node4").SetStyle(Styles.LABEL_SIZE, 13.0);

    QuickFinder<Node> finderByProperty = new QuickFinder<Node>(box, "Name");
    QuickFinder<Node> finderByClient = new QuickFinder<Node>(box, "Client1",
        Consts.PROPERTY_TYPE_CLIENT);
```

```

QuickFinder<Node> finderByStyle = new QuickFinder<Node>(box, Styles.LABEL_SIZE,
    Consts.PROPERTY_TYPE_STYLE, null, (node)=>{return node.Name == "Name0";});

System.Diagnostics.Debug.WriteLine("ByProperty:\t" + ToString(finderByProperty.Find("Name1")));
System.Diagnostics.Debug.WriteLine("ByClient:\t" + ToString(finderByClient.Find("Value1")));
System.Diagnostics.Debug.WriteLine("ByStyle:\t" + ToString(finderByStyle.Find(13.0)));
}

private static string ToString(ICollection<Node> datas)
{
    StringBuilder builder = new StringBuilder();
    if (datas != null)
    {
        foreach (IData data in datas)
        {
            builder.Append(" " + data.Name);
        }
    }
    return builder.ToString();
}

```

Result:

```

ByProperty:    Name1
ByClient:      Name2 Name3
ByStyle:       Name0

```

Network Component

In 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 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:

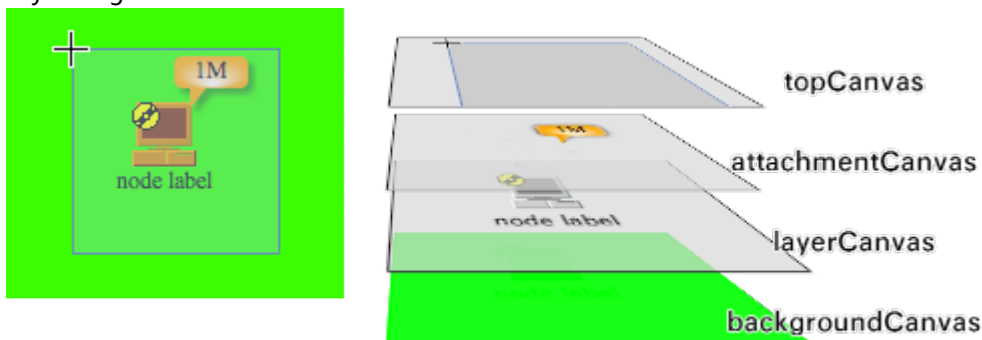
Network hierarchy:

- > ScrollViewer
- > NetworkCanvas
 - > 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 showInAttachmentCanvas is true, Attachment will be placed in attachmentCanvas layer

Layer diagram:



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

```
public NetworkCanvas NetworkCanvas { get; }
public GraphCanvas RootCanvas { get; }
public GraphCanvas TopCanvas { get; }
public GraphCanvas AttachmentCanvas { get; }
public GraphCanvas LayerCanvas { get; }
public GraphCanvas BottomCanvas { get; }
public GraphCanvas BackgroundCanvas { get; }
```

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

```
using System;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using TWaver;
using TWaver.Network;
namespace TestTWaverSilverlight
{
```

```

public class TestNetworkHierarchy
{
    private Network network = null;

    public TestNetworkHierarchy(Network network)
    {
        this.network = network;
    }

    public void InitDataBox()
    {
        Utils.RegisterImage("linkStatus2", new BitmapImage(new Uri("linkStatus2.png", UriKind.Relative)), 10, 10);

        Node node = new Node();
        node.SetLocation(80, 30);
        node.Name = "node label";
        node.AlarmState.IncreaseNewAlarm(AlarmSeverity.MAJOR);
        node.SetStyle(Styles.ICON_NAMES, "linkStatus2");
        network.ElementBox.Add(node);

        Button buttonInRootCanvas = new Button();
        buttonInRootCanvas.Name = "button in rootCanvas";
        buttonInRootCanvas.Margin = new System.Windows.Thickness(100, 40, 0, 0);
        buttonInRootCanvas.Height = 100;
        network.RootCanvas.Children.Add(buttonInRootCanvas);

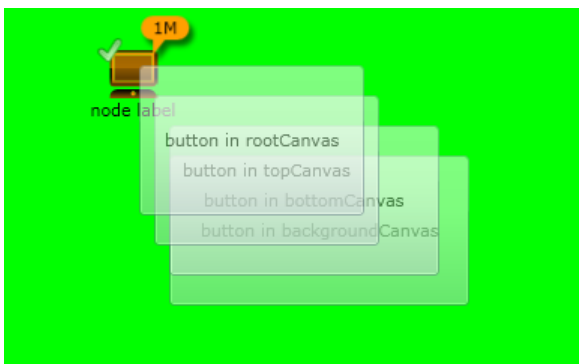
        Button buttonInTopCanvas = new Button();
        buttonInTopCanvas.Name = "button in topCanvas";
        buttonInTopCanvas.Margin = new System.Windows.Thickness(110, 60, 0, 0);
        buttonInTopCanvas.Height = 100;
        network.TopCanvas.Children.Add(buttonInTopCanvas);

        Button buttonInBottomCanvas = new Button();
        buttonInBottomCanvas.Name = "button in bottomCanvas";
        buttonInBottomCanvas.Margin = new System.Windows.Thickness(120, 80, 0, 0);
        buttonInBottomCanvas.Height = 100;
        network.BottomCanvas.Children.Add(buttonInBottomCanvas);

        Button buttonInBackgroundCanvas = new Button();
        buttonInBackgroundCanvas.Name = "button in backgroundCanvas";
        buttonInBackgroundCanvas.Margin = new System.Windows.Thickness(120, 100, 0, 0);
        buttonInBackgroundCanvas.Height = 100;
        network.BackgroundCanvas.Children.Add(buttonInBackgroundCanvas);

        network.ElementBox.SetStyle(Styles.BACKGROUND_TYPE, Consts.BACKGROUND_TYPE_VECTOR);
        network.ElementBox.SetStyle(Styles.BACKGROUND_VECTOR_FILL_COLOR, Colors.Green);
    }
}

```



Network Background

TWaver .NET supports multi style background including raster formatted picture, color, color gradient, combination of picture and color gradient and etc.

The background property of TWaver .NET 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 convinient to serialize. TWaver take background as parts of data, it shouldn't be lost when serializing.

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

Background Style

```
public const string BACKGROUND_IMAGE = "background.image";
public const string BACKGROUND_COLOR = "background.color";
public const string BACKGROUND_GRADIENT = "background.gradient";
public const string BACKGROUND_GRADIENT_COLOR = "background.gradient.color";
```

For example:

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

```
using System;
using System.Windows;
using System.Windows.Media;
using TWaver;
using TWaver.Network;

namespace TestTWaverSilverlight
{
    public class TestNetworkBackground
    {
        private Network network = new Network();

        public TestNetworkBackground(MainPage mainPage)
        {
            mainPage.LayoutRoot.Children.Add(this.network);
            this.InitDataBox();
        }

        public void InitDataBox()
        {
            //Register picture, All picture in TWaver need to be registered before use
            Utils.RegisterPNGImage("background", new Uri("/TestTWaverSilverlight;component/Images/europe.png",
            UriKind.Relative));

            //Set background picture
            network.ElementBox.SetStyle(Styles.BACKGROUND_IMAGE, "background");

            SubNetwork subnetworkA = AddSubnetwork("subnetwork A");
            subnetworkA.Location = new Point(50, 50);
            //Set filling color and gradient color
            subnetworkA.SetStyle(Styles.BACKGROUND_COLOR, ToColor(0xFF9CB5D8));
```

```

subnetworkA.SetStyle(Styles.BACKGROUND_GRADIENT_COLOR, ToColor(0xFFFFFFFF));
//Set gradient style to be vertical spread
subnetworkA.SetStyle(Styles.BACKGROUND_GRADIENT, Consts.GRADIENT_SPREAD_VERTICAL);

SubNetwork subnetworkB = AddSubnetwork("subnetwork B");
subnetworkB.Location = new Point(250, 50);
//Set background color
subnetworkB.SetStyle(Styles.BACKGROUND_COLOR, ToColor(0xFF0000FF));

SubNetwork subnetworkC = AddSubnetwork("subnetwork C");
subnetworkC.Location = new Point(150, 250);
//Set background picture
subnetworkC.SetStyle(Styles.BACKGROUND_IMAGE, "background");
//Set filling color and gradient color
subnetworkC.SetStyle(Styles.BACKGROUND_COLOR, ToColor(0xFF9CB5D8));
subnetworkC.SetStyle(Styles.BACKGROUND_GRADIENT_COLOR, ToColor(0xFFFFFFFF));
//Set gradient style to be vertical spread
subnetworkC.SetStyle(Styles.BACKGROUND_GRADIENT, Consts.GRADIENT_SPREAD_VERTICAL);

network.ElementBox.Add(new Link(subnetworkA, subnetworkB));
network.ElementBox.Add(new Link(subnetworkB, subnetworkC));
network.ElementBox.Add(new Link(subnetworkC, subnetworkA));
}

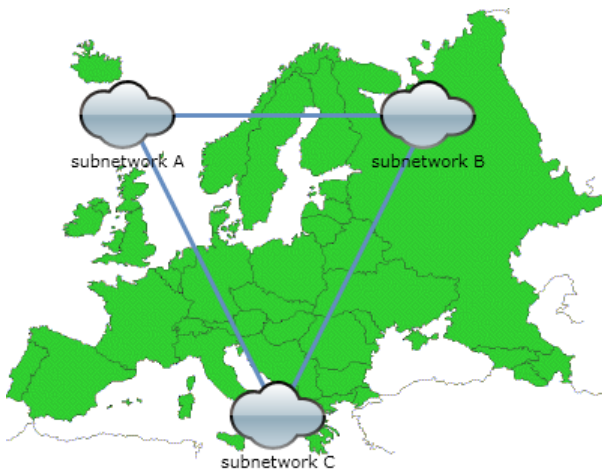
private SubNetwork AddSubnetwork(String name)
{
    SubNetwork subnetwork = new SubNetwork();
    subnetwork.Name = name;
    network.ElementBox.Add(subnetwork);
    Node from = new Node();
    from.Parent = subnetwork;
    from.Name = "From";
    from.Location = new Point(20, 20);
    network.ElementBox.Add(from);
    Node to = new Node();
    to.Parent = subnetwork;
    to.Name = "To";
    to.Location = new Point(150, 60);
    network.ElementBox.Add(to);
    Link link = new Link(from, to);
    link.Parent = subnetwork;
    link.Name = "Hello TWaver!";
    network.ElementBox.Add(link);
    return subnetwork;
}

private static Color ToColor(uint argb)
{
    byte a = (byte)((argb & -16777216) >> 0x18);
    byte r = (byte)((argb & 0xff0000) >> 0x10);
    byte g = (byte)((argb & 0xff00) >> 8);
    byte b = (byte)(argb & 0xff);
    return Color.FromArgb(a, r, g, b);
}
}

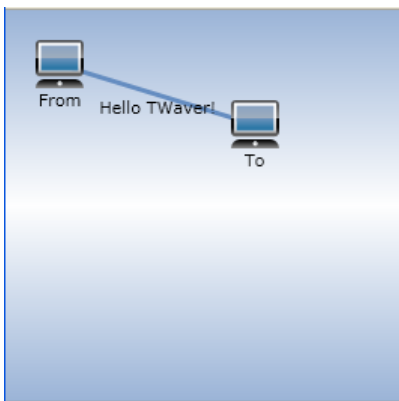
```

Results:

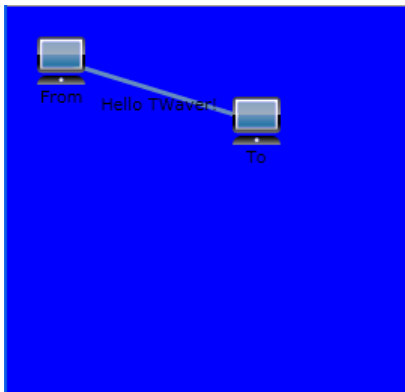
Picture background:



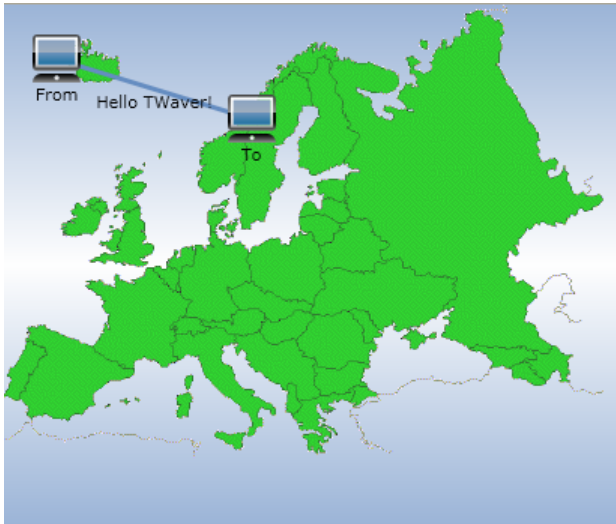
Gradient color background:



Pure color background:



Picture and Gradient color background:



Network Filter

TWaver .NET offers a series of filter including visible filter, movable filter, editable filter and etc. TWaver .NET 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 Predicate<IElement> VisibleFunction { get; set; }
//movable filter
public Predicate<IElement> MovableFunction { get; set; }
//editable filter
public Predicate<IElement> EditableFunction { get; set; }
```

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 with children to be visible:

```
using TWaver;
using TWaver.Network;

namespace TestTWaverSilverlight
{
    public class TestNetworkFilter
    {
        private Network network = null;

        public TestNetworkFilter(Network network)
        {
            this.network = network;
        }

        public void InitDataBox()
        {
            network.VisibleFunction = (e) => { return e.ChildrenCount > 0; };

            SubNetwork subNetwork = new SubNetwork();
            network.SubNetworkAnimate = false;
            subNetwork.Location = new System.Windows.Point(100, 100);
            subNetwork.Name = "SubNetwork";
            network.ElementBox.Add(subNetwork);

            Node from = new Node();
            from.Location = new System.Windows.Point(50, 50);
            from.Name = "From";
            from.Parent = subNetwork;
            network.ElementBox.Add(from);

            Node to = new Node();
            to.Location = new System.Windows.Point(200, 200);
            to.Name = "To";
            to.Parent = subNetwork;
            network.ElementBox.Add(to);

            Link link = new Link(from, to);
            link.Name = "Link";
            link.Parent = subNetwork;
            network.ElementBox.Add(link);

            Node notVisibleNode = new Node();
```

```
notVisibleNode.Location = new System.Windows.Point(50, 100);  
notVisibleNode.Name = "Not Visible Node";  
network.ElementBox.Add(notVisibleNode);  
}  
}  
}
```

Network Function for Style Rule

TWaver .NET 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:

Func<TWaver.Network.Network, IElement, ElementUI> 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. Default is Defaults.ELEMENTUI_FUNCTION.

Func<IElement, string> LabelFunction : Label function for network element text, is to set label for network element. Default is Defaults.LABEL_FUNCTION.

Func<IData, string> ToolTipFunction : The hint of toolbar, default is Defaults.TOOLTIP_FUNCTION.

Func<IData, Color?> InnerColorFunction : Rendering color function inner network element, default is Defaults.INNER_COLOR_FUNCTION.

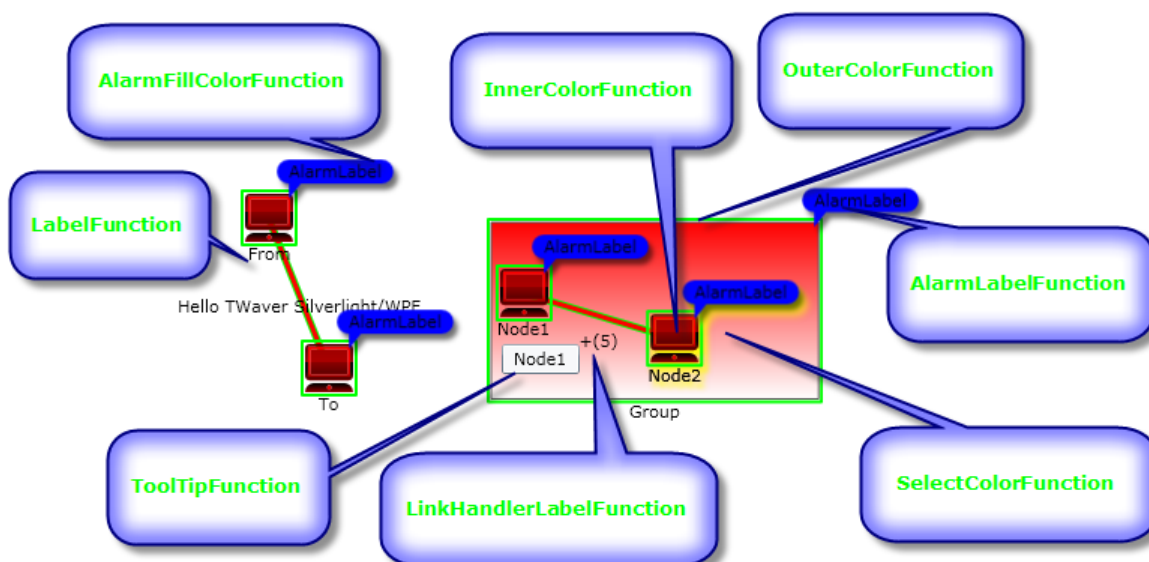
Func<IData, Color?> OuterColorFunction : Border color of network element, default is Defaults. OUTER_COLOR_FUNCTION.

Func<IData, Color> SelectColorFunction : Selected color of network element, default is Defaults.SELECT_COLOR_FUNCTION.

Func<IData, Color?> AlarmFillColorFunction : Alarm bubble color, default is Defaults.ALARM_FILL_COLOR_FUNCTION.

Func<IElement, string> AlarmLabelFunction : Alarm bubble text, default is Defaults.ALARM_LABEL_FUNCTION.

Func<Link, string> LinkHandlerLabelFunction : Bundle link text, default is Defaults.LINK_HANDLER_LABEL_FUNCTION.



Default

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

```
//elementUIFunction
public static Func<TWaver.Network.Network, IElement, ElementUI> ELEMENTUI_FUNCTION = (network, element)
=>
{
    Type type = element.ElementUIClass;
    if (type != null)
    {
        return (ElementUI)Activator.CreateInstance(type, network, element);
    }
    return null;
};

//labelFunction
public static Func<IElement, string> LABEL_FUNCTION = (element) =>
{
    string label = element.GetStyle<string>(Styles.NETWORK_LABEL);
    if (label != null)
    {
        return label;
    }
    return element.Name;
};

//toolTipFunction
public static Func<IData, string> TOOLTIP_FUNCTION = (data) => data.ToolTip;

//innerColorFunction
public static Func<IData, Color?> INNER_COLOR_FUNCTION = (data) =>
{
    IElement element = data as IElement;
    if (element != null){
        AlarmSeverity severity = element.AlarmState.HighestNativeAlarmSeverity;
        if (severity != null){
            return severity.Color;
        }
        return element.GetStyle<Color?>(Styles.INNER_COLOR);
    }
    return null;
};

//outerColorFunction
public static Func<IData, Color?> OUTER_COLOR_FUNCTION = (data) =>
{
    IElement element = data as IElement;
    if (element != null){
        AlarmSeverity severity = element.AlarmState.PropagateSeverity;
        if (severity != null){
            return severity.Color;
        }
        return element.GetStyle<Color?>(Styles.OUTER_COLOR);
    }
    return null;
};

//selectColorFunction : selected color of network element
public static Func<IData, Color> SELECT_COLOR_FUNCTION = (data) =>
{
    if (data is IElement)
    {
        return ((IElement)data).GetStyle<Color>(Styles.SELECT_COLOR);
    }
}
```

```

    }
    return Consts.COLOR_DARK;
};

//alarmFillColorFunction : alarm bubble color
public static Func<IData, Color?> ALARM_FILL_COLOR_FUNCTION = (data) =>
{
    if (data is IElement)
    {
        AlarmSeverity severity = ((IElement)data).AlarmState.HighestNewAlarmSeverity;
        if(severity != null){
            return severity.Color;
        }
    }
    return null;
};

//alarmLabelFunction
public static Func<IElement, string> ALARM_LABEL_FUNCTION = (element) =>
{
    AlarmSeverity severity = element.AlarmState.HighestNewAlarmSeverity;
    if(severity != null){
        string label = element.AlarmState.GetNewAlarmCount(severity) + severity.NickName;
        if(element.AlarmState.HasLessSevereNewAlarms){
            label += "+";
        }
        return label;
    }
    return null;
};

//linkHandlerLabelFunction
public static Func<Link, string> LINK_HANDLER_LABEL_FUNCTION = (link) =>
{
    if (link.IsBundleAgent())
    {
        return "(" + link.BundleCount + ")";
    }
    return null;
};

```

Rule Making

```

using System.Windows.Media;
using TWaver;
using TWaver.Network;

namespace TestTWaverSilverlight
{
    public class TestNetworkFunction
    {
        private Network network = new Network();

        public TestNetworkFunction(MainPage mainPage)
        {
            mainPage.LayoutRoot.Children.Add(this.network);
            this.InitDataBox();
        }

        public void InitDataBox()
        {
            Group group = new Group();

```

```

group.Name = "Group";
network.ElementBox.Add(group);
Node node1 = CreateTWaverNode("Node1", 220, 100);
group.AddChild(node1);
Node node2 = CreateTWaverNode("Node2", 320, 130);
group.AddChild(node2);
group.IsExpanded = true;
Link bundleLink1 = new Link(node1, node2);
network.ElementBox.Add(bundleLink1);
network.ElementBox.Add(new Link(node1, node2));
network.ElementBox.Add(new Link(node1, node2));
network.ElementBox.Add(new Link(node1, node2));
network.ElementBox.Add(new Link(node1, node2));
bundleLink1.ReverseBundleExpanded();

Node from = CreateTWaverNode("From", 50, 50);
Node to = CreateTWaverNode("To", 90, 150);
Link link = new Link(from, to);
link.Name = "Hello TWaver .NET";
network.ElementBox.Add(link);

//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 = data => ToColor(0xFFFF0000);

//border color of node
network.OuterColorFunction = data => ToColor(0xFF00FF00);

//alarm bubble color, blank if alarmLabel return null or color is null
network.AlarmFillColorFunction = data =>
{
    if (data is Element && !(data as Element).AlarmState.IsEmpty())
    {
        return ToColor(0xFF0000FF);
    }
    return null;
};

network.AlarmLabelFunction = element =>
{
    if (!element.AlarmState.IsEmpty())
    {
        return "AlarmLabel";
    }
    return null;
};

network.SelectColorFunction = data => ToColor(0xFFFFFFFF00);
}

private Node CreateTWaverNode(string name, int x, int y)
{
    Node node = new Node();
    node.Name = name;
    node.ToolTip = name;
    node.AlarmState.IncreaseNewAlarm(AlarmSeverity.MAJOR);
    node.SetLocation(x, y);
    network.ElementBox.Add(node);
    return node;
}

private static Color ToColor(uint argb)

```



```
{  
    byte a = (byte)((argb & -16777216) >> 0x18);  
    byte r = (byte)((argb & 0xff0000) >> 0x10);  
    byte g = (byte)((argb & 0xff00) >> 8);  
    byte b = (byte)(argb & 0xff);  
    return Color.FromArgb(a, r, g, b);  
}  
}
```

Network GUI Interaction

InteractionHandler

Mouse and keyboard interaction on topology is composed by a set of InteractionHandler. And each InteractionHandler contains installing and uninstalling event listener operation. InteractionHandler defined as below:

```
namespace TWaver.Network.Interaction
{
    public interface InteractionHandler
    {
        void InstallListeners();
        void UninstallListeners();
    }
}
```

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 IList<InteractionHandler> InteractionHandlers { get; set; }
```

Set default interaction:

```
public void SetDefaultInteractionHandlers(bool lazyMode)
{
    IList<InteractionHandler> list = new List<InteractionHandler>();
    list.Add(new SelectInteractionHandler(this));
    list.Add(new MoveInteractionHandler(this, lazyMode));
    list.Add(new DefaultInteractionHandler(this));
    this.InteractionHandlers = list;
}
```

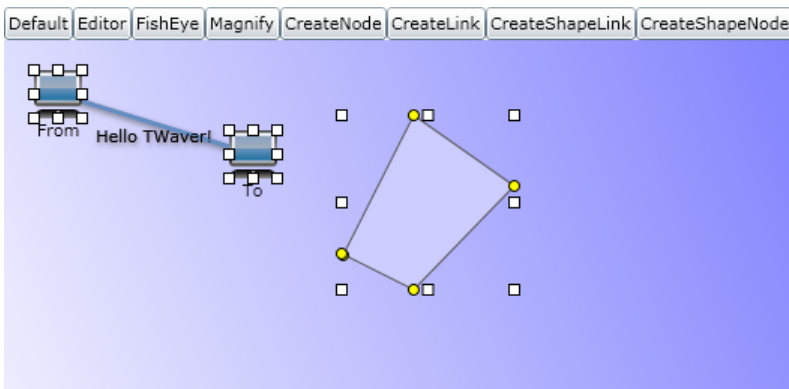
Commonly Used Interaction Model

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

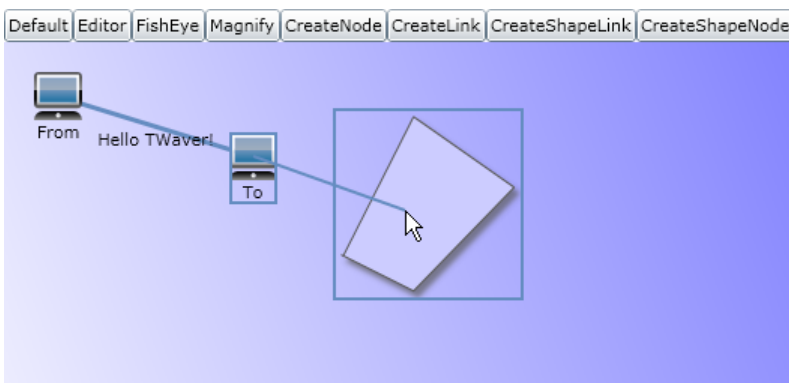
```
public void SetDefaultInteractionHandlers(bool lazyMode);
public void SetEditInteractionHandlers(bool lazyMode);
public void SetCreateElementInteractionHandlers(Type type);
public void SetCreateLinkInteractionHandlers(Type type);
public void SetCreateShapeLinkInteractionHandlers(Type type);
```

```
public void SetCreateShapeNodeInteractionHandlers(Type type);
```

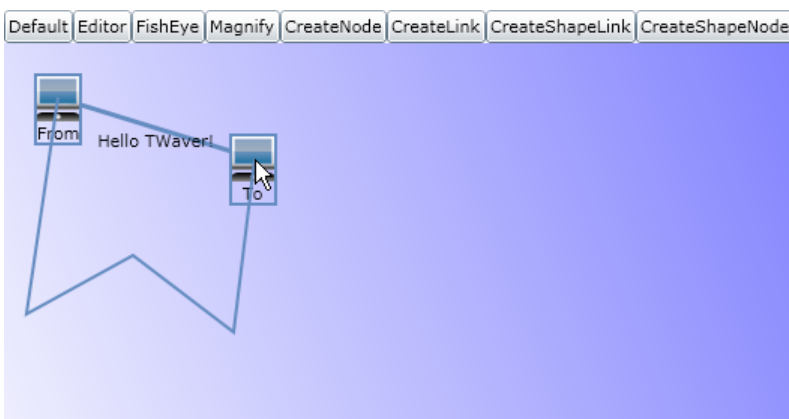
Editable Model



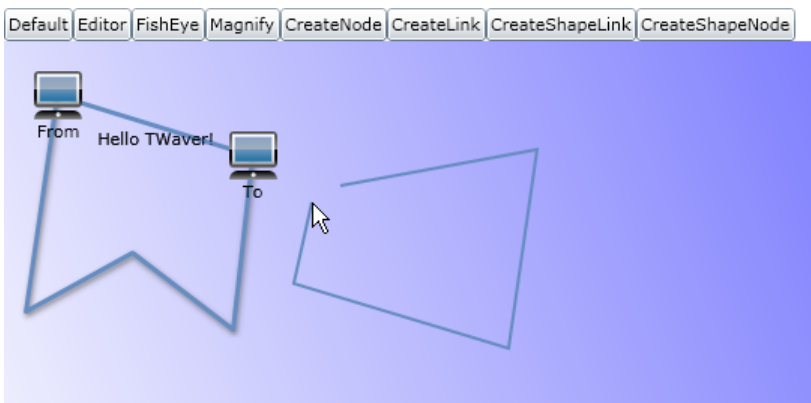
Creating Link Interaction Model



Creating ShapeLink Interaction Model



Creating ShapeNode Interaction Model

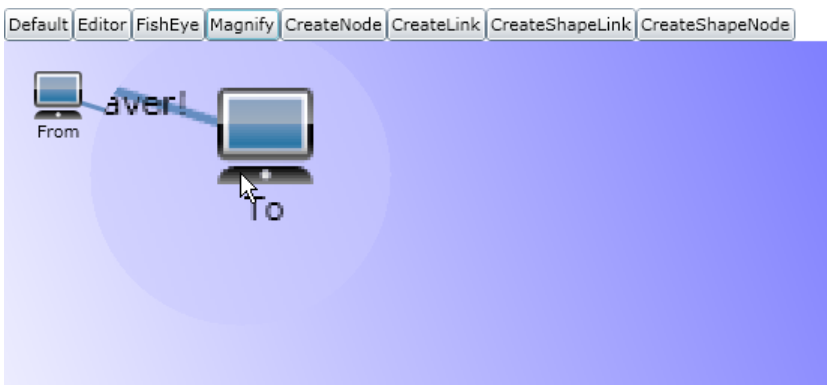


Magnifier Interaction:

```

IList<InteractionHandler> list = new List<InteractionHandler>();
list.Add(new SelectInteractionHandler(network));
list.Add(new EditInteractionHandler(network));
list.Add(new MoveInteractionHandler(network, false));
list.Add(new DefaultInteractionHandler(network));
list.Add(new MapFilterInteractionHandler(network, Consts.MAP_FILTER_MAGNIFY));
network.InteractionHandlers = list;

```

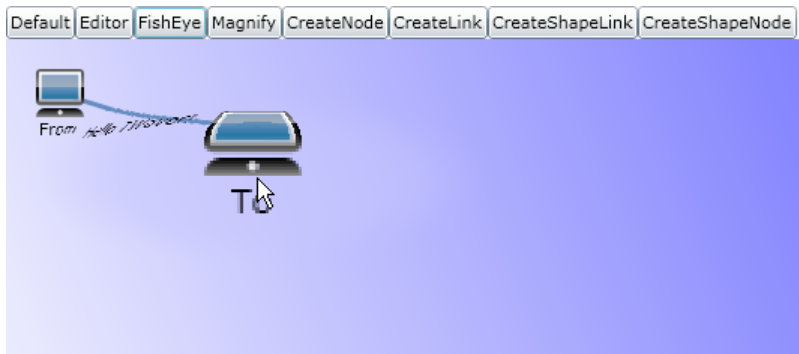


Fish-eye Interaction:

```

IList<InteractionHandler> list = new List<InteractionHandler>();
list.Add(new SelectInteractionHandler(network));
list.Add(new EditInteractionHandler(network));
list.Add(new MoveInteractionHandler(network, false));
list.Add(new DefaultInteractionHandler(network));
list.Add(new MapFilterInteractionHandler(network));
network.InteractionHandlers = list;

```



Here is the example:

```
<UserControl x:Class="TestTWaverSilverlight.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

  <Grid x:Name="LayoutRoot">
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="*/>
    </Grid.RowDefinitions>

    <StackPanel x:Name="buttonPane" Orientation="Horizontal" Grid.Column="0" Grid.Row="0"/>
    <Grid x:Name="networkPane" Grid.Column="0" Grid.Row="1"/>
  </Grid>
</UserControl>
```

```
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using TWaver;
using TWaver.Network;
using TWaver.Network.Interaction;

namespace TestTWaverSilverlight
{
  public partial class MainPage : UserControl
  {
    private Network network = new Network();

    public MainPage()
    {
      InitializeComponent();
      InitButtons();
      this.networkPane.Children.Add(this.network);
      InitDataBox();
    }

    public void InitButtons()
    {
      this.AddButton("Default", (sender, e) => { network.SetDefaultInteractionHandlers(false); });
      this.AddButton("Editor", (sender, e) => { network.SetEditInteractionHandlers(false); });
      this.AddButton("FishEye", (sender, e) =>
      {
        IList<InteractionHandler> list = new List<InteractionHandler>();
        list.Add(new SelectInteractionHandler(network));
        list.Add(new EditInteractionHandler(network));
      });
    }
  }
}
```

```

        list.Add(new MoveInteractionHandler(network, false));
        list.Add(new DefaultInteractionHandler(network));
        list.Add(new FishEyeInteractionHandler(network));
        network.InteractionHandlers = list;
    });
    this.AddButton("Magnify", (sender, e) =>
    {
        IList<InteractionHandler> list = new List<InteractionHandler>();
        list.Add(new SelectInteractionHandler(network));
        list.Add(new EditInteractionHandler(network));
        list.Add(new MoveInteractionHandler(network, false));
        list.Add(new DefaultInteractionHandler(network));
        list.Add(new MagnifyInteractionHandler(network));
        network.InteractionHandlers = list;
    });
    this.AddButton("CreateNode", (sender, e) => { network.SetCreateElementInteractionHandlers(typeof(Node)); });
    this.AddButton("CreateLink", (sender, e) => { network.SetCreateLinkInteractionHandlers(typeof(Link)); });
    this.AddButton("CreateShapeLink", (sender, e) =>
    { network.SetCreateShapeLinkInteractionHandlers(typeof(ShapeLink)); });
    this.AddButton("CreateShapeNode", (sender, e) =>
    { network.SetCreateShapeNodeInteractionHandlers(typeof(ShapeNode)); });
    }

    private void AddButton(string name, RoutedEventHandler handler)
    {
        Button button = new Button();
        button.Content = name;
        button.Click += handler;
        this.buttonPane.Children.Add(button);
    }

    private void InitDataBox()
    {
        network.ElementBox.SetStyle(Styles.BACKGROUND_COLOR, Colors.Blue);
        network.ElementBox.SetStyle(Styles.BACKGROUND_GRADIENT_COLOR, Colors.White);
        network.ElementBox.SetStyle(Styles.BACKGROUND_GRADIENT, Consts.GRADIENT_LINEAR_SOUTHWEST);

        Node from = new Node();
        from.Name = "From";
        from.Location = new Point(20, 20);
        network.ElementBox.Add(from);
        Node to = new Node();
        to.Name = "To";
        to.Location = new Point(150, 60);
        network.ElementBox.Add(to);
        Link link = new Link(from, to);
        link.Name = "Hello TWaver!";
        network.ElementBox.Add(link);
        network.SetEditInteractionHandlers(true);
    }
}

```

Default Editor FishEye Magnify CreateNode CreateLink CreateShapeLink CreateShapeNode



FAQ

- [Using Element ID as value of Parent and Host When Serializing](#)
- [Using TWaver WPF In WinForm](#)
- [Integrate TWaver Silverlight with Microsoft Bing Maps Silverlight Control](#)

Using Element ID as value of Parent and Host When Serializing

By default, TWaver uses Ref ID which is an auto-increased number from 0 as value of Parent and Host when serializing. This topic describes how to replace Ref ID with Element ID as value of parent and host when serializing.

1. Create a new class which extends from TWaver.XMLSerializer.
2. Override method SerializeValue.
3. Handle property Parent and Host specially.

```
if ("Parent" == property || "Host" == property)
{
    writer.WriteStartElement(c);
    writer.WriteAttributeString("N", property);

    IElement element = (value as IElement);
    if (element != null)
    {
        object id = element.ID;
        writer.WriteAttributeString("Ref", id.ToString());
    }

    writer.WriteEndElement();
}
else
{
    base.SerializeValue(writer, c, property, value, newInstanceValue, type, isCDATA);
}
```

4. Nothing to do with deserializing, TWaver will lookup element by element ID when Ref ID can not be found.

```
using System.IO;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Xml;
using Microsoft.Win32;
using TWaver;
using TWaver.Network;

namespace Demo.Databox
{
    public partial class CustomSerializerDemo : UserControl, IDemo
    {
        private ElementBox box;
        private Network network;

        public CustomSerializerDemo()
        {
            InitializeComponent();

            this.network = new Network();
            this.box = this.network.ElementBox;
            this.LayoutRoot.Children.Add(this.network);
            DemoUtils.InitToolBar(this.network, this.Toolbar);
            Button btnSerialize = DemoUtils.AddToolBarButton(this.Toolbar, "", "Serialize",
                (sender, e) => { Serialize(); });
            InitBox();
        }
    }
}
```



```

private void InitBox()
{
    SubNetwork home = new SubNetwork("MyHome");
    home.Name = "MyHome";
    home.Location = new Point(150, 150);
    this.box.Add(home);

    Node father = new Node("Father");
    father.Name = "Father";
    father.Location = new Point(100, 100);
    father.Parent = home;
    this.box.Add(father);

    Follower me = new Follower("Myself");
    me.Name = "Myself";
    me.Location = new Point(200, 200);
    me.Parent = home;
    me.Host = father;
    this.box.Add(me);
}

private void Serialize()
{
    CustomXMLSerializer serializer = new CustomXMLSerializer(this.box);
    serializer.SerializationSettings.IsAssemblyQualifiedName = false;
    SaveFileDialog dialog = new SaveFileDialog();
    dialog.Filter = "XML Files(*.xml) | *.xml";
    dialog.DefaultExt = "xml";
    bool? result = dialog.ShowDialog();
    if (result == true)
    {
        Stream stream = dialog.OpenFile();
        byte[] content = Encoding.UTF8.GetBytes(serializer.Serialize());
        stream.Write(content, 0, content.Length);
        stream.Close();
    }
}

public Network Network { get { return this.network; } }

public void OnShown()
{
}

public void OnHidden()
{
}

class CustomXMLSerializer : XMLSerializer<IElement>
{
    public CustomXMLSerializer(DataBox<IElement> box)
        : base(box)
    {
    }

    public override void SerializeValue(XmlWriter writer, string c, string property, object value,
        object newInstanceValue, string type, bool isCDATA)
    {
        if ("Parent" == property || "Host" == property)
        {
            writer.WriteStartElement(c);
            writer.WriteAttributeString("N", property);

            IElement element = (value as IElement);
            if (element != null)
            {

```

```
        object id = element.ID;
        writer.WriteAttributeString("Ref", id.ToString());
    }

    writer.WriteEndElement();
}
else
{
    base.SerializeValue(writer, c, property, value, newInstanceValue, type, isCDATA);
}
}
}
}
```

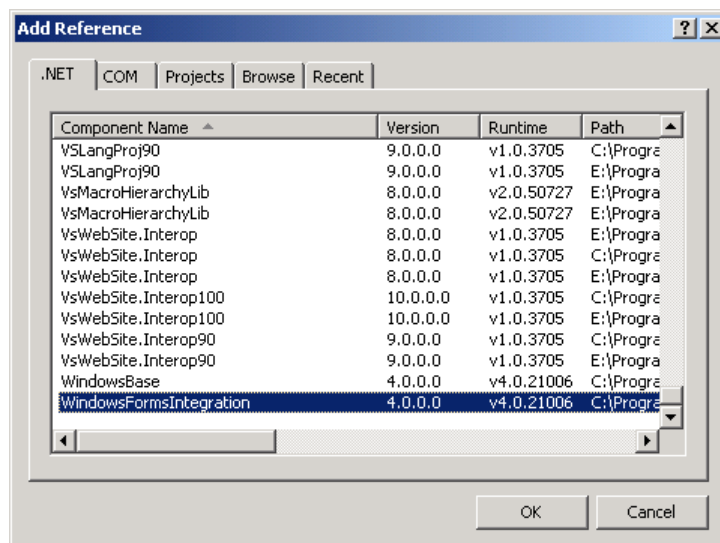
Using TWaver WPF In WinForm

This topic describes how to embed TWaver WPF network inside of a WinForm application.

Add reference

Add WindowsFormsIntegration component reference

1. Select the WinForm project, then select Project, then Add Reference from the Visual Studio main menu.
2. In the Add Reference dialog box, click the .NET tab.
3. Scroll to the bottom on the .NET tab and choose the WindowsFormsIntegration component, and click OK.

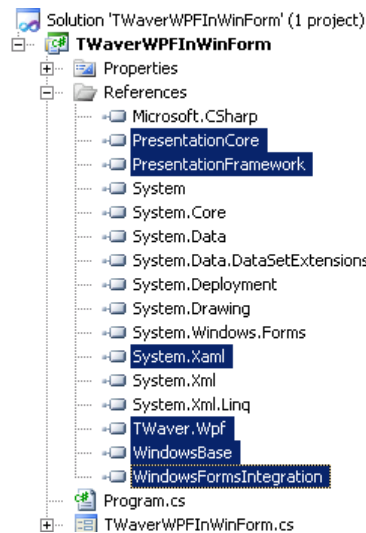


Add TWaver WPF library

1. Select the WinForm project, then select Project, then Add Reference from the Visual Studio main menu.
2. In the Add Reference dialog box, click the Browse tab.
3. Open the TWaver WPF library folder, select TWaver.Wpf.dll file and click OK.

Add other .Net component references

1. Select the WinForm project, then select Project, then Add Reference from the Visual Studio main menu.
2. In the Add Reference dialog box, click the .NET tab.
3. Select PresentationCore, PresentationFramework, System.Xaml and WindowsBase components, and click OK.



Add ElementHost component and set TWaver network as child of ElementHost

```
using System;
using System.Windows.Forms;
using System.Windows.Forms.Integration;
using System.Windows.Media.Imaging;
using TWaver;
using TWaver.Network;

namespace TWaverWPFIInWinForm
{
    public partial class TWaverWPFIInWinForm : Form
    {
        private ElementBox box = new ElementBox();

        public TWaverWPFIInWinForm()
        {
            RegisterImage("router", 73, 13);
            InitializeComponent();
            InitDataBox();
            InitWPFHostComponent();
        }

        private void InitWPFHostComponent()
        {
            ElementHost elementHost = new ElementHost();
            elementHost.Dock = System.Windows.Forms.DockStyle.Fill;
            Network network = new Network(box);
            elementHost.Child = network;
            this.Controls.Add(elementHost);
        }

        private void InitDataBox()
        {
            Node node1 = new Node();
            node1.Location = new System.Windows.Point(50, 50);
            node1.Image = "router";
            node1.Name = "Node1";
            box.Add(node1);

            Node node2 = new Node();
            node2.Location = new System.Windows.Point(200, 200);
            node2.Image = "router";
            node2.Name = "Node2";
        }
    }
}
```

```

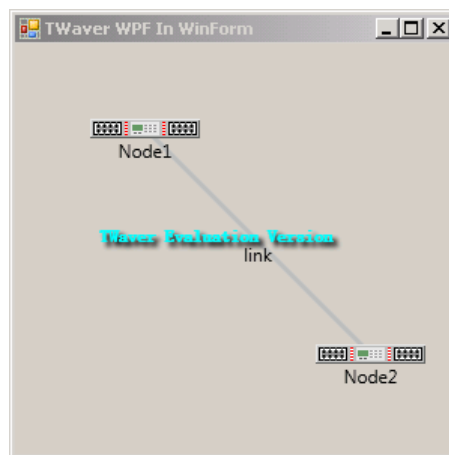
        box.Add(node2);

        Link link = new Link(node1, node2);
        link.Name = "link";
        box.Add(link);
    }

    private void RegisterImage(string name, double width, double height)
    {
        string uri = "Images/" + name + ".png";
        BitmapImage bitmapImage =
            new BitmapImage(new Uri(uri, UriKind.RelativeOrAbsolute));
        Utils.RegisterImage(name, bitmapImage, width, height);
    }
}

```

Run application



Integrate TWaver Silverlight with Microsoft Bing Maps Silverlight Control

This topic contains steps for integrating TWaver Silverlight with Microsoft Bing Maps Silverlight Control.

Setup Environment

Before start, make sure Microsoft Visual Studio 2008 SP1 and Microsoft Silverlight 3 Tools for Visual Studio 2008 SP1 are installed. Go to [Setup Environment](#) page for reference.

Create a Bing Maps Developer Account

In order to use the Bing Maps Silverlight Control, you need a Bing Maps Key to authenticate your application. To obtain Bing Maps Keys, first go to the [Bing Maps Accounts Center](#) and create a Bing Maps Developer Account. You'll get a Bing Maps Key associated with a URL, for example, key is Ag0BbWLMC3WIFteC_4E0_76NG7UxWa5-0irl9RrF-jVVinbmE77ymweCdDIR6iU5 and URL is <http://localhost:2731>. The URL is the address of your web site, and the key will be used later.

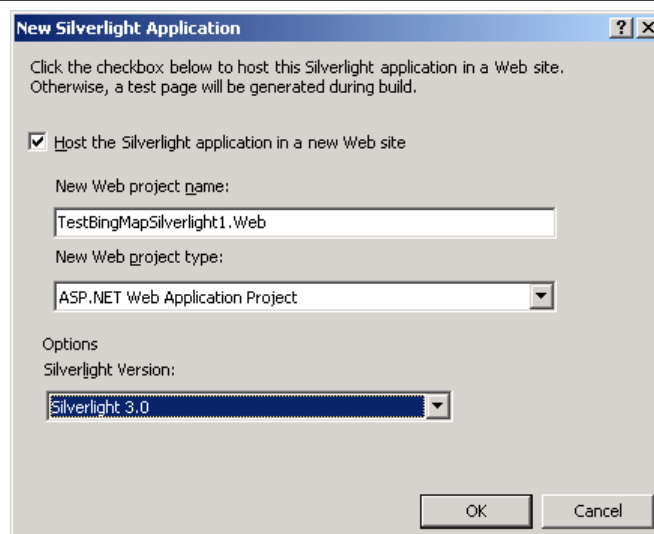
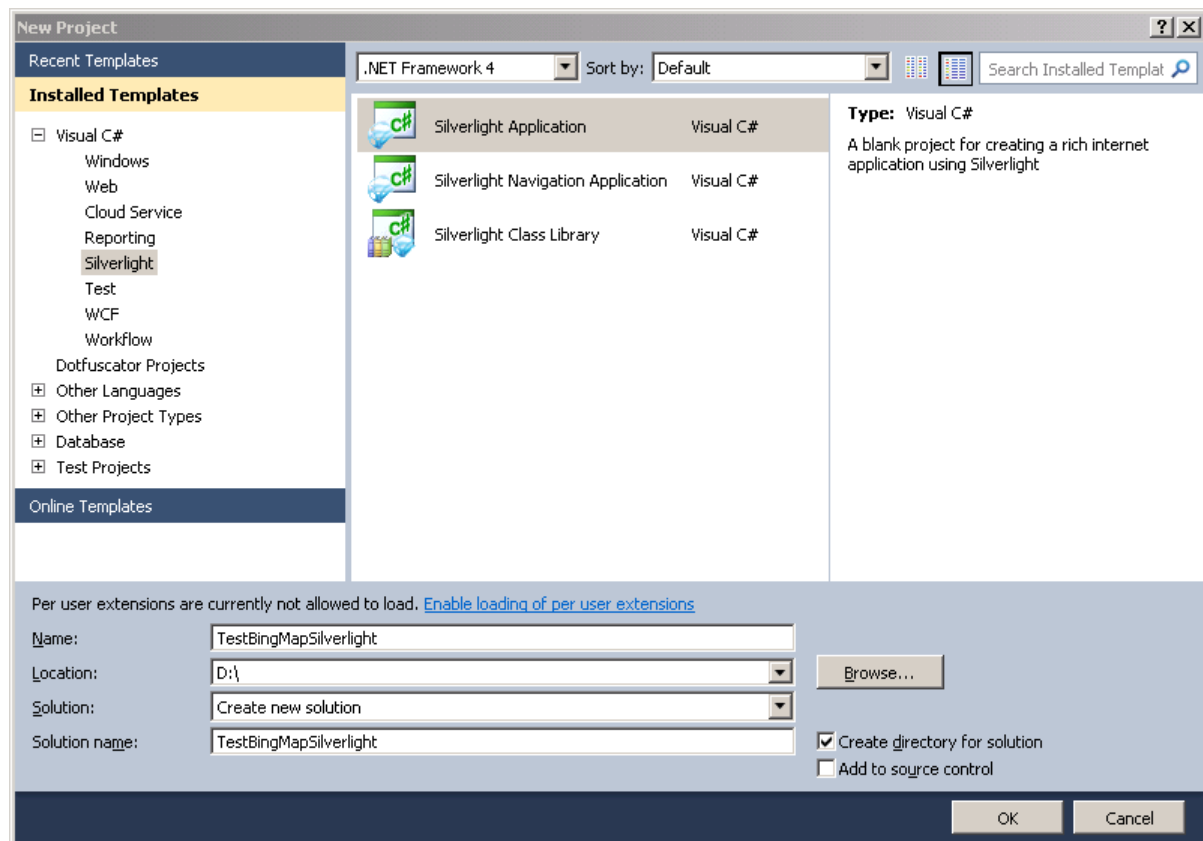
Download and install the Microsoft Bing Maps Silverlight Control

Download and install the [Microsoft Bing Maps Silverlight Control](#). The default installation path is C:\Program Files\Bing Maps Silverlight Control, and the assembly Microsoft.Maps.MapControl.Common.dll and Microsoft.Maps.MapControl.dll under folder C:\Program Files\Bing Maps Silverlight Control\V1\Libraries will be used later.

Creating a Basic Application

Create a Silverlight Project

1. Open Visual Studio 2008.
2. Select File from the main menu.
3. Select New, and then Project.
4. In the New Project dialog box, under the language Visual C#, select Silverlight.
5. Select Silverlight Application from the available templates, Input Name(Fro example: TestBingMapSilverlight) and Location, then click OK.
6. Make sure the Host the Silverlight application in a new Web site option is checked on the next New Silverlight Application dialog, then click OK.



Add reference to project

Add TWaver Silverlight library

1. Select the Silverlight project you just created, then select Project, then Add Reference from the Visual Studio main menu.
2. In the Add Reference dialog box, click the Browse tab.
3. Open the TWaver Silverlight library folder, select TWaver.Silverlight.dll file and click OK.

Add Bing Silverlight library

1. Select the Silverlight project you just created, then select Project, then Add Reference from the Visual Studio main menu.
2. In the Add Reference dialog box, click the Browse tab.

3. Open the Bing Silverlight library folder, select the Microsoft.Maps.MapControl.dll and the Microsoft.Maps.MapControl.Common.dll files and click OK.

Show TWaver Network on Bing Map

1. Add the map control assembly to MainPage.xaml

```
<UserControl x:Class="TestBingMapSilverlight.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:m="clr-namespace:Microsoft.Maps.MapControl;assembly=Microsoft.Maps.MapControl">

    <Grid x:Name="LayoutRoot">
    </Grid>
</UserControl>
```

2. Once the map control assembly is referenced, then you can add the map element <m:Map/> to your page. Put this element within the Grid, also authenticate your use of the map control by set the CredentialsProvider attribute to the Bing Maps Key.

```
<UserControl x:Class="TestBingMapSilverlight.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:m="clr-namespace:Microsoft.Maps.MapControl;assembly=Microsoft.Maps.MapControl">

    <Grid x:Name="LayoutRoot">
        <m:Map CredentialsProvider=
            "AkudmSZ5eNmBD_jqsPFIY7cpSXPxnpPu85Bc893NBjdQD9pgg9wphOI_-L_Ds4u"/>
    </Grid>
</UserControl>
```

3. Add buttons to navigate between Map Views.

```
<UserControl x:Class="TestBingMapSilverlight.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:m="clr-namespace:Microsoft.Maps.MapControl;assembly=Microsoft.Maps.MapControl">

    <Grid x:Name="LayoutRoot">
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="40" />
            <RowDefinition Height="20" />
        </Grid.RowDefinitions>
        <m:Map Name="myBingMap" Mode="AerialWithLabels" Grid.RowSpan="3" Padding="5"
            CredentialsProvider=
                "AkudmSZ5eNmBD_jqsPFIY7cpSXPxnpPu85Bc893NBjdQD9pgg9wphOI_-L_Ds4u"
            Center="39.3683,-95.2734,0.0000" ZoomLevel="4.000"/>
        <StackPanel Orientation="Horizontal" Opacity="0.7" Grid.Column="0" Grid.Row="1"
            HorizontalAlignment="Center">
            <Button x:Name="btnNorthAmerica" Click="ChangeMapView_Click"
                Tag="39.3683,-95.2734,0.0000" Margin="5">
                <TextBlock>North America</TextBlock>
            </Button>
            <Button x:Name="btnNewYork" Click="ChangeMapView_Click"
                Tag="40.7199,-74.0030,0.0000" Margin="5">
                <TextBlock>New York</TextBlock>
            </Button>
        </StackPanel>
    </Grid>
```



```

</StackPanel>
<StackPanel Orientation="Horizontal" Opacity="0.9" Grid.Column="0" Grid.Row="2"
    HorizontalAlignment="Center">
    <TextBlock Text="Latitude: " Padding="5" Foreground="White"/>
    <TextBox x:Name="txtLatitude" Text="" IsReadOnly="True" Background="LightGray"/>
    <TextBlock Text="Longitude: " Padding="5" Foreground="White" />
    <TextBox x:Name="txtLongitude" Text="" IsReadOnly="True" Background="LightGray"/>
</StackPanel>
</Grid>
</UserControl>

```

4. Add TWaver Newtork to Bing Map

```

using System;
using System.Globalization;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using Microsoft.Maps.MapControl;
using Microsoft.Maps.MapControl.Design;
using TWaver;
using TWaver.Network;

namespace TestBingMapSilverlight
{
    public partial class MainPage : UserControl
    {
        private const String LatitudeAndLongitude = "LatitudeAndLongitude";

        private LocationConverter locConverter = new LocationConverter();
        private Network network = new Network();

        public MainPage()
        {
            InitializeComponent();
            // Displays the current latitude and longitude as the map animates.
            myBingMap.ViewChangeOnFrame +=
                new EventHandler<MapEventArgs>(myBingMap_ViewChangeOnFrame);
            // The default animation level: navigate between different map locations.
            myBingMap.AnimationLevel = AnimationLevel.Full;
            // Init TWaver Network.
            initNetwork();
            // Add TWaver Network to Bing Map.
            myBingMap.Children.Add(network);
        }

        private void initNetwork()
        {
            network.Opacity = 0.9;

            Node nodeNorthAmerica = new Node();
            nodeNorthAmerica.Location = new System.Windows.Point(100, 100);
            nodeNorthAmerica.SetClient(LatitudeAndLongitude,
                locConverter.ConvertFrom("39.3683,-95.2734,0.0000"));
            nodeNorthAmerica.Name = "North America";
            nodeNorthAmerica.AlarmState.IncreaseNewAlarm(AlarmSeverity.Severities[4]);
            nodeNorthAmerica.SetStyle(Styles.LABEL_COLOR, Colors.Red);
            network.ElementBox.Add(nodeNorthAmerica);

            Node nodeNewYork = new Node();
            nodeNewYork.Location = new System.Windows.Point(200, 200);
            nodeNewYork.SetClient(LatitudeAndLongitude,
                locConverter.ConvertFrom("40.7199,-74.0030,0.0000"));

```

```

nodeNewYork.Name = "NewYork";
nodeNewYork.SetStyle(Styles.LABEL_COLOR, Colors.Red);
network.ElementBox.Add(nodeNewYork);

Link link = new Link(nodeNorthAmerica, nodeNewYork);
link.Name = "Link from North America to NewYork";
link.SetStyle(Styles.LABEL_COLOR, Colors.Red);
network.ElementBox.Add(link);
// Init location of all nodes according it's latitude and longitude.
InitNodeLocation();
}

private void InitNodeLocation()
{
    foreach (IElement element in network.ElementBox.Datas)
    {
        if (element is Node)
        {
            Location location = (Location)element.GetClient(LatitudeAndLongitude);
            Point point = myBingMap.LocationToViewportPoint(location);
            ((Node)element).CenterLocation = point;
        }
    }
}

private void myBingMap_ViewChangeOnFrame(object sender, MapEventArgs e)
{
    // Gets the map object that raised this event.
    Map map = sender as Map;
    // Determine if we have a valid map object.
    if (map != null)
    {
        // Gets the center of the current map view for this particular frame.
        Location mapCenter = map.Center;

        // Updates the latitude and longitude values, in real time,
        // as the map animates to the new location.
        txtLatitude.Text = string.Format(CultureInfo.InvariantCulture,
            "{0:F5}", mapCenter.Latitude);
        txtLongitude.Text = string.Format(CultureInfo.InvariantCulture,
            "{0:F5}", mapCenter.Longitude);

        InitNodeLocation();
    }
}

private void ChangeMapView_Click(object sender, RoutedEventArgs e)
{
    // Parse the information of the button's Tag property
    Location center = (Location)locConverter.ConvertFrom(((Button)sender).Tag);
    // Set the map view
    myBingMap.SetView(center, myBingMap.ZoomLevel);
}
}

```

5. Run your Application.

