

# TWaver .NET Training

2011

Serva Software LLC

# TWaver .NET Introduction

TWaver™

- What is TWaver?
- .NET Basics
  - The Relationship of C# to .NET
  - Silverlight & WPF ...
  - Visual Studio Basics
- TWaver .NET Basics
  - Hello TWaver!
  - MVC Pattern
  - Data Items/Data Models/Views
- Comparison with TWaver Java & Flex

# What is TWaver?

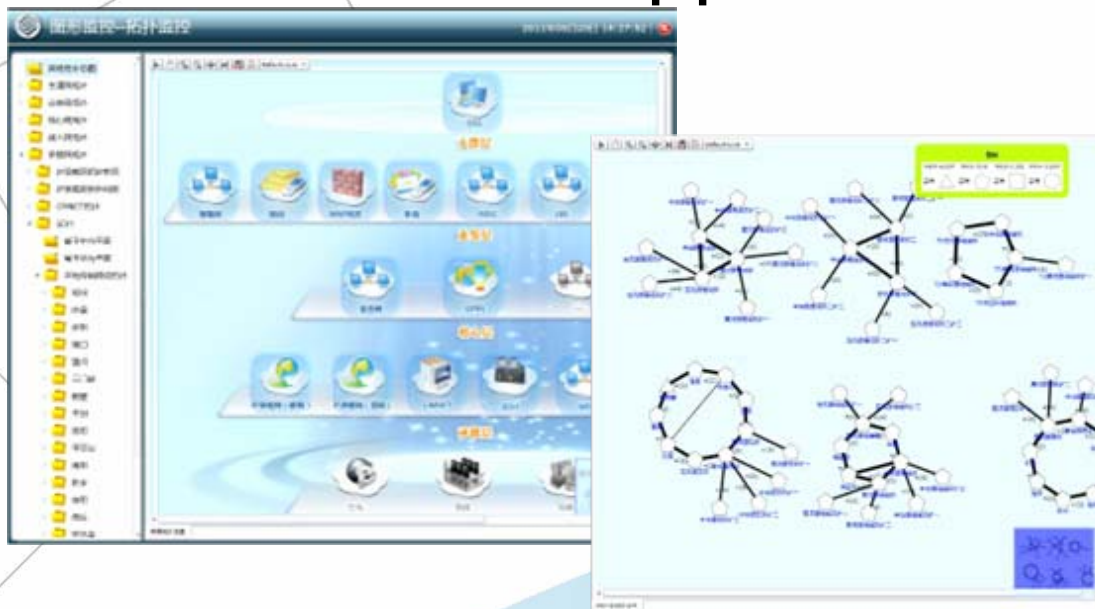
TWaver™

- High efficiency and lightweight graphical component library
- Multi-platform solutions
- TWaver .NET package contents
- TWaver license

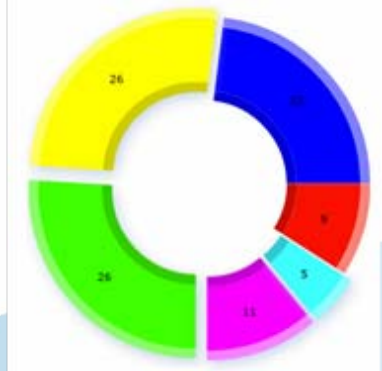
# TWaver is Graphical Component Library

- TWaver focus on graphical display of data
- TWaver is for developers, needs secondary development
- TWaver provides documentation, license, training and technical supports

TWaver™



- [-] TWaver Silverlight Demos
  - [x] Alarm Demos
    - [x] Alarm Statistics Demo
    - [x] Alarm Mapping Demo
    - [x] Alarm Propagation Demo
  - [-] Network Demos
    - [x] Topology Demos
    - [x] Equipment Demos
  - [x] Tree Demos
    - [x] Tree Layout Demo
    - [x] File Tree Demo

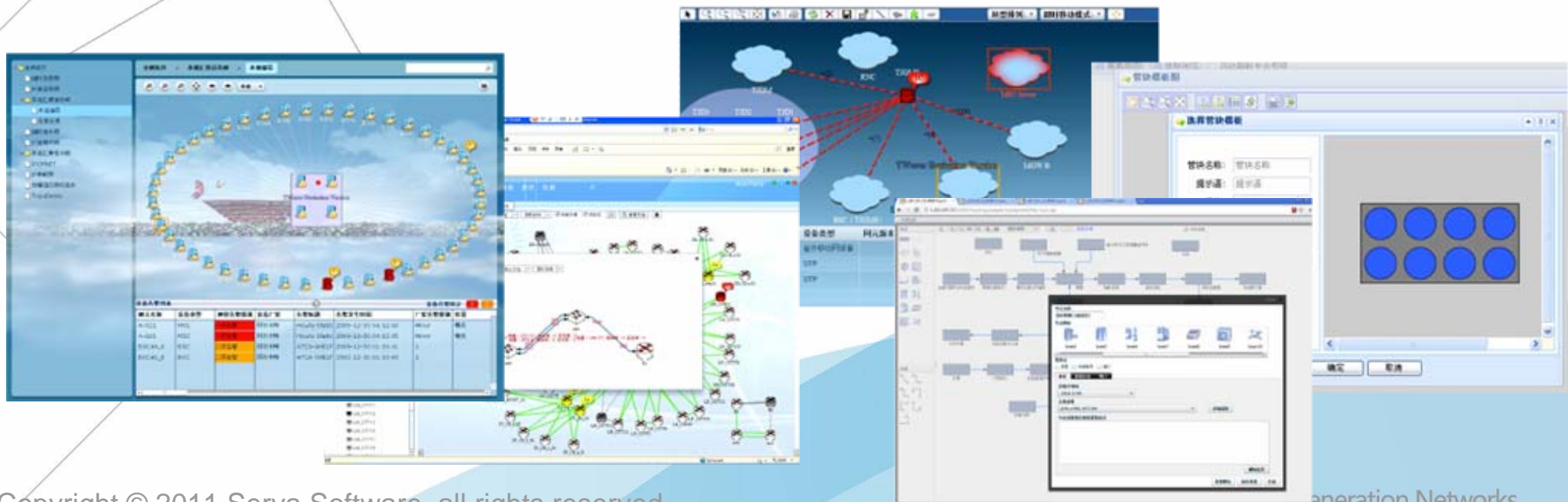


Mapping ID	Severity	Acked	Cleared	Raised Time
1	Indeterminate	<input type="checkbox"/>	<input type="checkbox"/>	7/29/2011 10:25:03 AM
2	Warning	<input type="checkbox"/>	<input type="checkbox"/>	7/29/2011 10:25:03 AM
3	Minor	<input type="checkbox"/>	<input type="checkbox"/>	7/29/2011 10:25:03 AM
4	Major	<input type="checkbox"/>	<input type="checkbox"/>	7/29/2011 10:25:03 AM
5	Critical	<input type="checkbox"/>	<input type="checkbox"/>	7/29/2011 10:25:03 AM
5	Indeterminate	<input checked="" type="checkbox"/>	<input type="checkbox"/>	7/29/2011 10:25:03 AM
4	Warning	<input checked="" type="checkbox"/>	<input type="checkbox"/>	7/29/2011 10:25:03 AM

# Focus on Telecom Network Management Components

- Provide telecommunications business models, equipment chassis, alarm transformation, etc.
- Have a large number of telecommunication application cases
- But is not limited in telecommunications industry

TWaver™

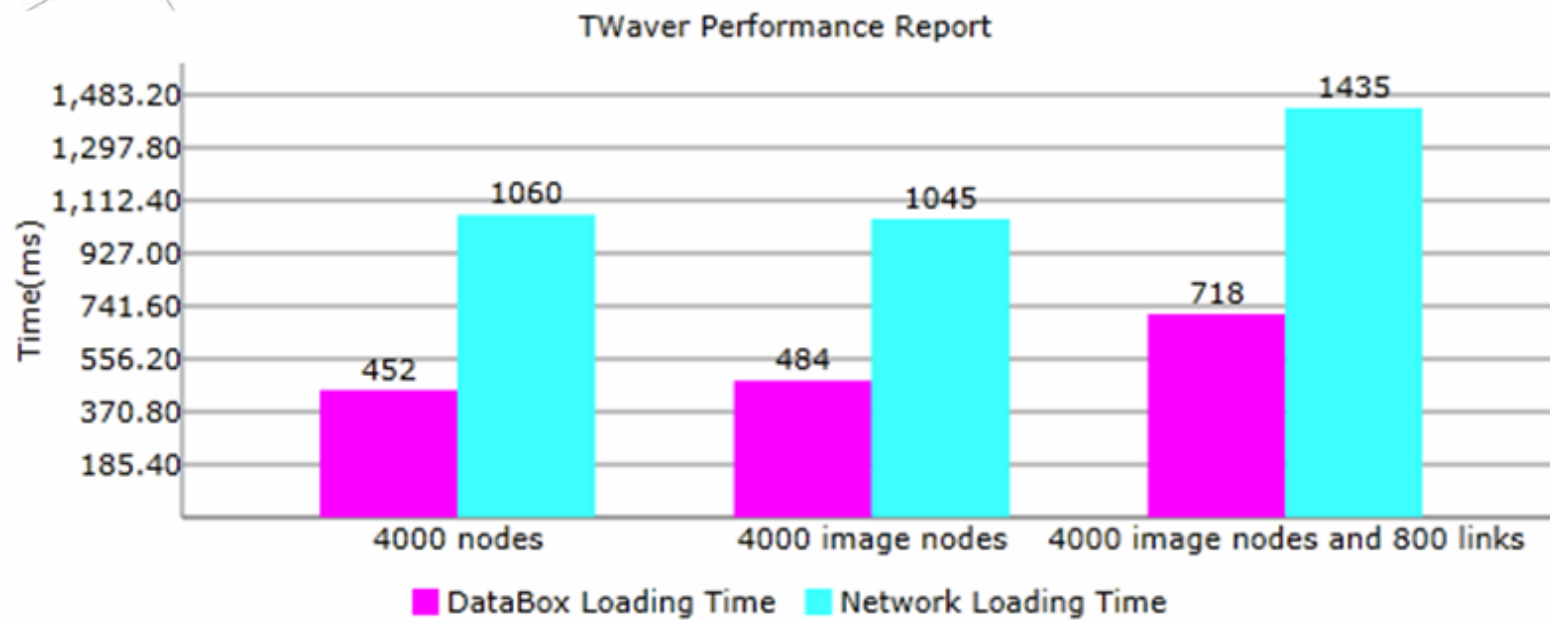




# High Efficiency and Lightweight

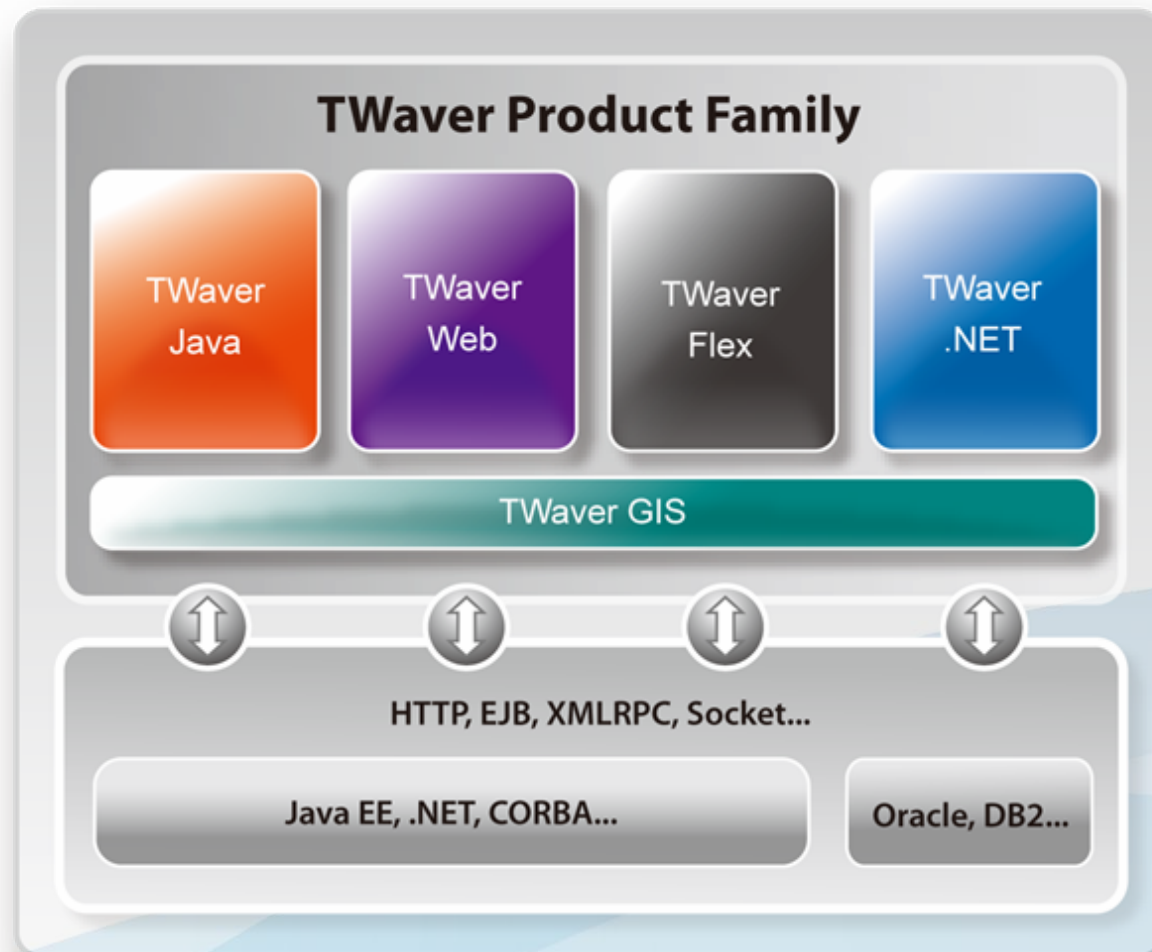
TWaver™

- TWaver.Silverlight.dll 900kb
- Less than 6000 nodes or links are suggested
- Load 4000 nodes and 800 links within 1.5s



# Multi-Platform Solutions

- Java
- Web
- Flex
- **.NET**
- HTML5...



# Customers

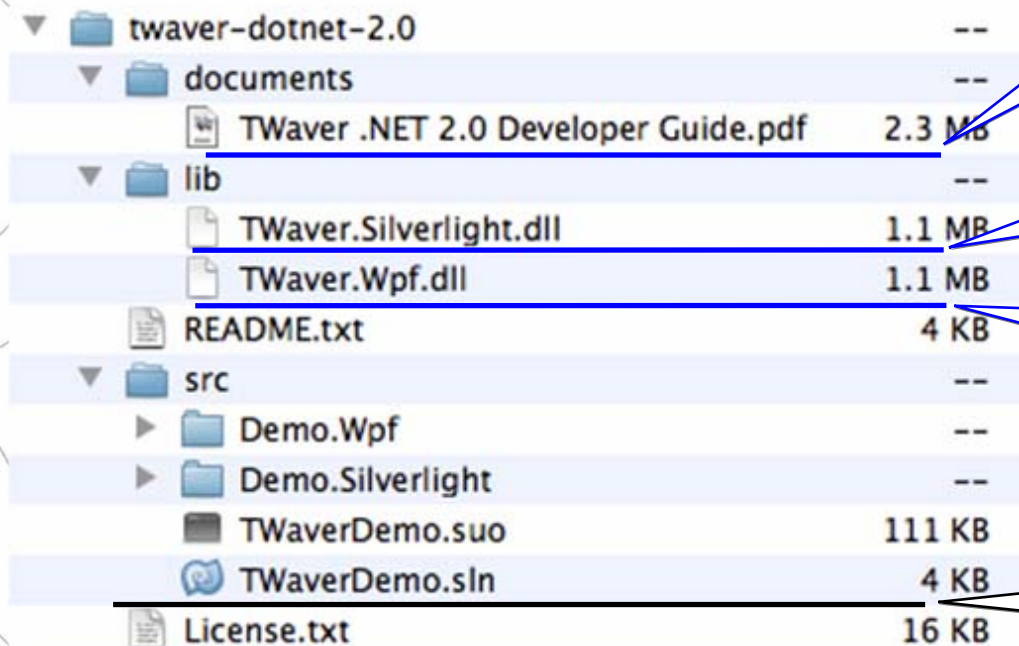
TM  
waver





# TWaver .NET Package Contents

TWaver™



▼ twaver-dotnet-2.0	--
▼ documents	--
TWaver .NET 2.0 Developer Guide.pdf	2.3 MB
▼ lib	--
TWaver.Silverlight.dll	1.1 MB
TWaver.Wpf.dll	1.1 MB
README.txt	4 KB
▼ src	--
▶ Demo.Wpf	--
▶ Demo.Silverlight	--
TWaverDemo.suo	111 KB
TWaverDemo.sln	4 KB
License.txt	16 KB

developer guider

TWaver.Silverlight.dll

TWaver.Wpf.dll

Silverlight and WPF  
demo sources

# TWaver License

TWaver™

- TWaver has three types of licenses: Evaluation License, Development License, and Runtime License. Click **Ctrl+Shift+L** to check license information on the interface of Silverlight or WPF.
- **Evaluation** - Used for prophecy on the initial stage or a period of technical selection, component interface will display watermark of "TWaver Evaluation Version" within 5 minutes
- **Development** - Grant you the right to use TWaver to develop your software product or project. The watermark will show after 2 hours.
- **Runtime** - Used for your final project/product deployment. No watermark

# How to Use License File

TWaver™

- license.xml is a plain-text format file, it contains license informations.

- How to use:

```
TWaver.Utills.ValidateLicense(new Uri(  
    "/PROJECT_NAME;component/license.xml",  
    UriKind.Relative));
```

# .NET Basics

Twaver™

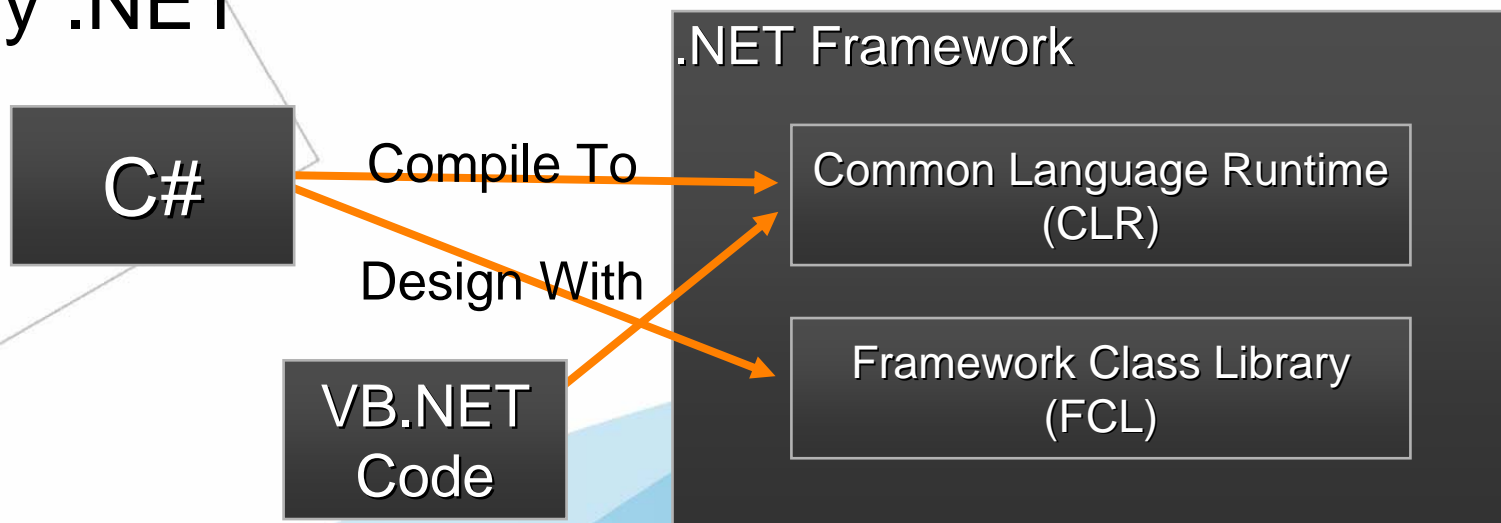
- C# & .NET
- C# Language
- Silverlight & WPF
- Others



# The Relationship of C# to .NET

TWaver™

- C# is specifically designed and targeted for use with Microsoft's .NET Framework
- C# is not itself part of .NET, some features are supported by .NET but not by C#, and some features of the C# language are not supported by .NET



# C# Language Features

*“C# is a multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines.” - from wikipedia*

- Get & Set
- Generics
- Delegates, Lambdas and Events

# get & set

```
namespace TWaver{  
    public class Data : IData {  
  
        ...  
  
        private string name;  
        public virtual string Name  
        {  
            get { return this.name; }  
            set  
            {  
                object oldValue = this.name;  
                this.name = value;  
                this.DispatchPropertyChangeEvent("Name", oldValue,  
value);  
            }  
        }  
        ...  
    }  
}
```

Example:

```
Data data = new Data();  
data.Name = "001";  
Console.WriteLine(data.Name);
```

TWaver™

# Generics

- Generic Classes

```
public class List<T> { }
```

```
public delegate TOutput Converter<TInput, TOutput>(TInput  
    from);
```

```
public interface IComparable<in T> {  
    int CompareTo(T other);  
}
```

- Using Generic

```
List<Node> nodes = new List<Node>();
```

```
DataBox<Element> box = new DataBox<Element>();
```

```
Tree<Element> tree = new Tree<Element>(box);
```



# Delegate, Lambdas and Events

TWaver™

## Define a delegate

```
delegate TOutput Converter<TInput, TOutput>(TInput from);
```

## Create a delegate instance

```
Converter<int, string> g = delegate(int input){  
    return "AA-" + input;  
};
```

## Create a delegate instance by Lambda expressions

```
Converter<int, string> g2 = (int input) =>{  
    return "AA-" + input;  
};
```

## Add event listener by Lambda expressions

```
tree.MouseLeftButtonDown += (object sender, MouseButtonEventArgs evt) =>{  
    Console.WriteLine("mouse click");  
};
```

# WPF & Silverlight

- WPF - Windows Presentation Foundation is a library to create the UI for smart client applications.
- Silverlight consists:
  - A core presentation framework (a subset of WPF), not supports flow and fixed documents and 3-D...
  - The .NET Framework for Silverlight (a subset of the .NET Framework)
  - Installers and updaters.

TM  
Tmaver

# WPF & Silverlight

- WPF applications run on a Windows system
- Silverlight uses a plugin model and is hosted in a web browser
- When you compile a WPF application, you get an executable assembly that contains XAML code in a binary form, named BAML, as a resource.
- With a Silverlight application, the compiler creates an XAP file that is a ZIP package containing the assembly and configuration.

TM  
waver

# Others

TM  
waver

- XAML & C#
- class & struct
- DispatcherObject, DependencyObject, Visual, UIElement, FrameworkElement, Control, ContentControl, ItemsControl, Panel, Application
- Shape, Path, Geometry, Transform ...



# TWaver Basics

TWaver™

- Hello TWaver
- MVC design pattern
- Data items and data models
- Views

# Hello TWaver

TWaver™

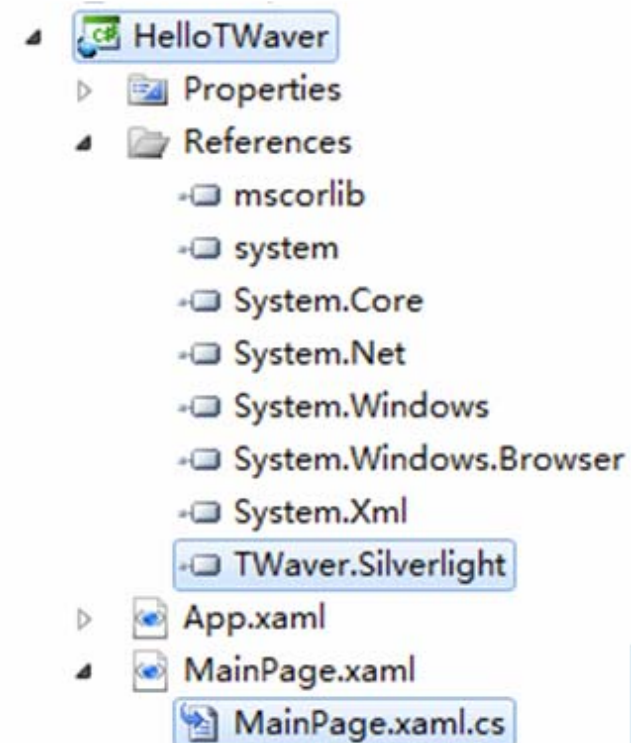
## TWaver .NET development environment

- TWaver.Silverlight.dll / TWaver.Wpf.dll
- Silverlight 4 Tools for Visual Studio 2010 / .NET Framework 4
- Visual Studio 2010
- Silverlight 4 / .NET Framework 4 Client Profile or .NET Framework 4

# Hello TWaver

- Create Silverlight Application
- Choose .NET Framework 4
- Add Reference - TWaver.Silverlight.dll

Refer to section "Setup Environment" from developer guide



TWaver™

# Hello TWaver

TWaver™

```
using TWaver;
using TWaver.NETwork;
namespace HelloTWaver{
    public partial class MainPage : UserControl {
        public MainPage() {
            InitializeComponent();

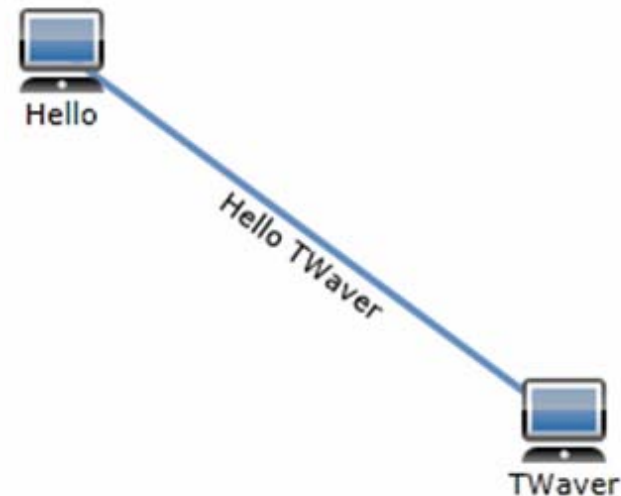
            ElementBox box = new ElementBox();

            Node node = new Node();
            node.Name = "Hello";
            node.SetLocation(10, 10);
            box.Add(node);

            Node node2 = new Node();
            node2.Name = "TWaver";
            node2.SetLocation(200, 150);
            box.Add(node2);

            Link link = new Link(node, node2);
            link.Name = "Hello TWaver";
            link.SetStyle(Styles.LINK_LABEL_ROTATABLE, true);
            box.Add(link);

            Network network = new Network(box);
            LayoutRoot.Children.Add(network);
        }
    }
}
```





# Configuration XAML

TWaver™

```
<navigation:Page x:Class="TWaverPPT.Demos.HelloTWaverFull"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="HelloTWaverFull Page">

  <Grid x:Name="LayoutRoot" Background="White" Margin="20" >
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="170" />
      <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
      <RowDefinition Height="*" />
      <RowDefinition Height="100" />
    </Grid.RowDefinitions>
  </Grid>
</navigation:Page>
```

# Adding Tree and Table

TWaver™

```
Network network = new Network(box);  
network.SetValue(System.Windows.Controls.Grid.ColumnProperty, 1);  
network.SetValue(System.Windows.Controls.Grid.RowProperty, 0);
```

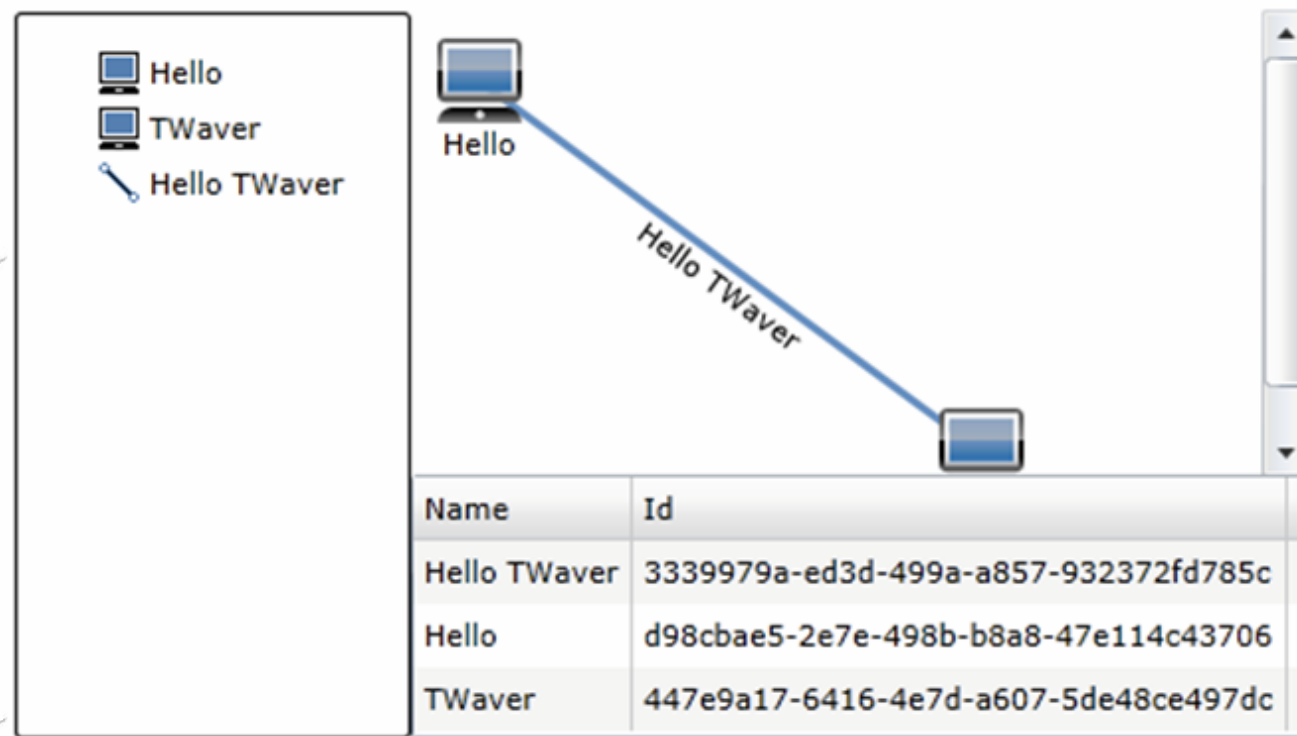
```
Tree<IElement> tree = new Tree<IElement>(box);  
tree.SetValue(System.Windows.Controls.Grid.ColumnProperty, 0);  
tree.SetValue(System.Windows.Controls.Grid.RowProperty, 0);  
tree.SetValue(System.Windows.Controls.Grid.RowSpanProperty, 2);
```

```
TWaver.Controls.Table<IElement> table = new TWaver.Controls.Table<IElement>(box);  
table.Columns.Add(new TableTextColumn(Consts.PROPERTY_TYPE_ACCESSOR,  
"Name", "Name"));  
table.Columns.Add(new TableTextColumn(Consts.PROPERTY_TYPE_ACCESSOR,  
"ID", "Id"));  
table.SetValue(System.Windows.Controls.Grid.ColumnProperty, 1);  
table.SetValue(System.Windows.Controls.Grid.RowProperty, 1);
```

```
LayoutRoot.Children.Add(network);LayoutRoot.Children.Add(tree);  
LayoutRoot.Children.Add(table);
```

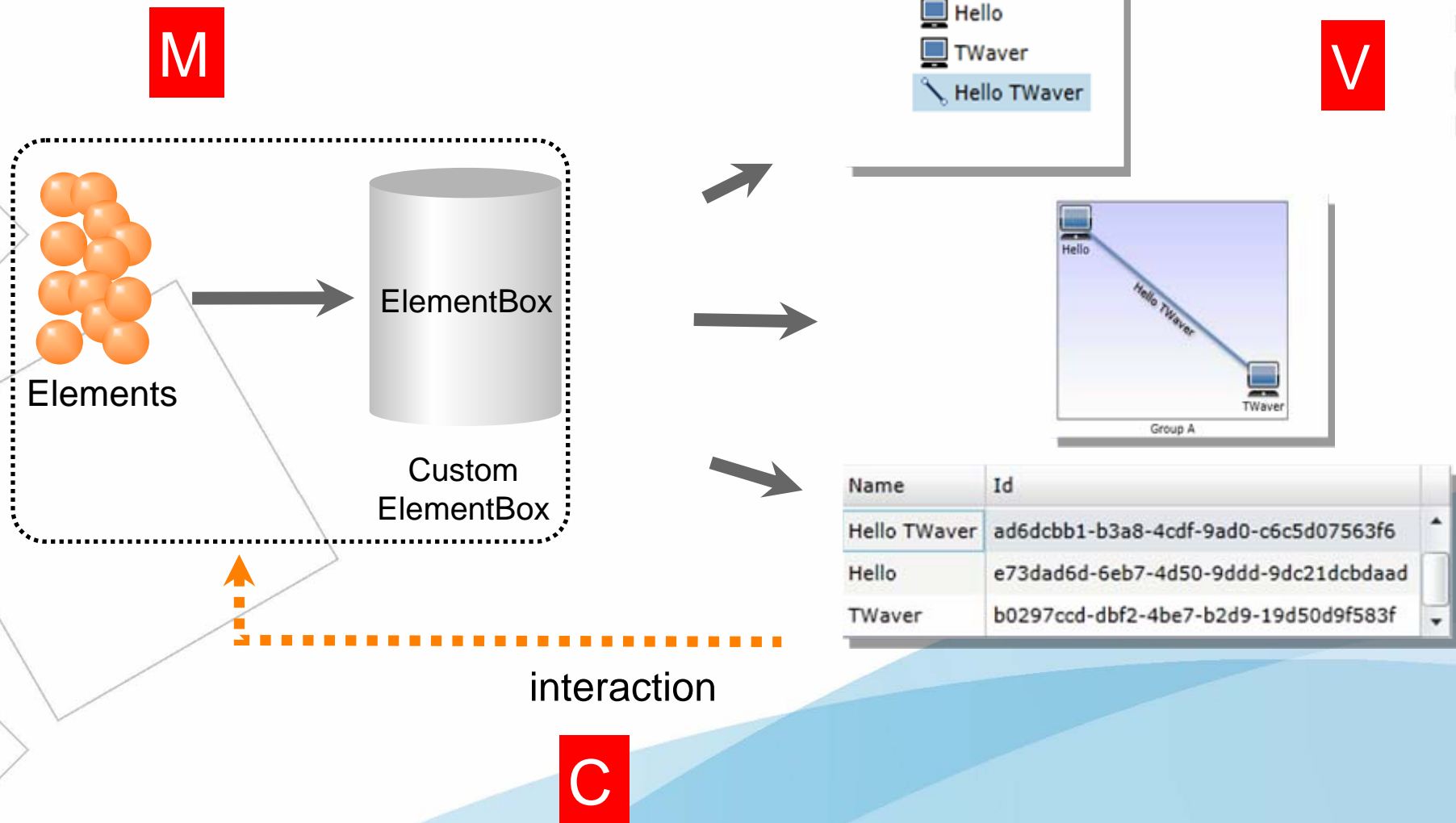
# Hello TWaver

TWaver™



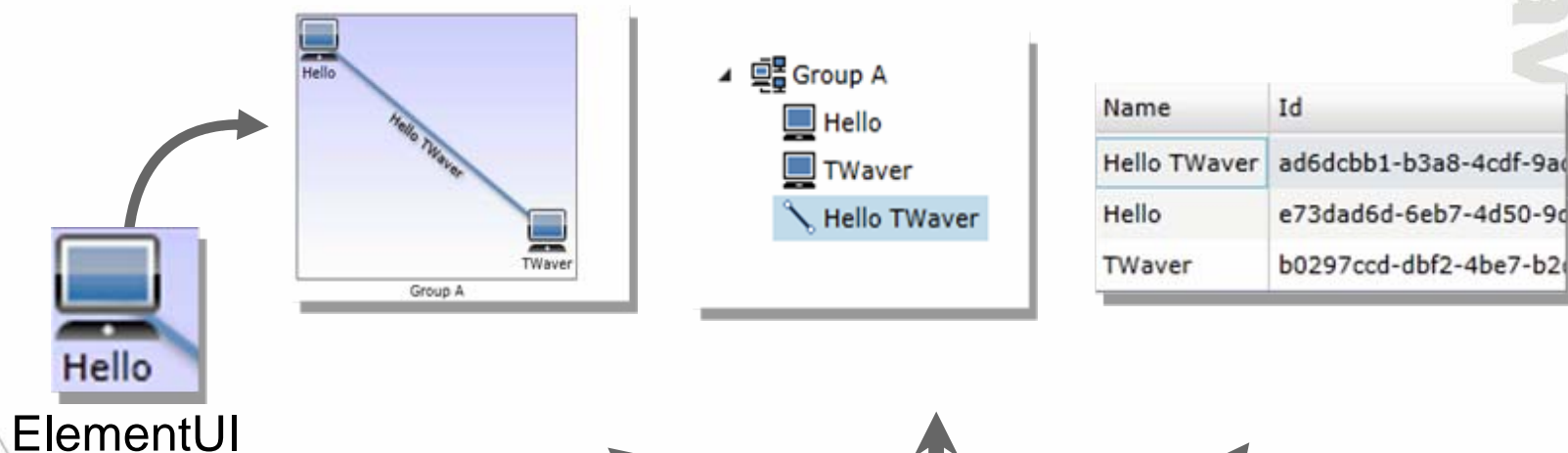
# MVC Design Pattern

TWaver™



# Model & Views

Views(V)



ElementUI

Element

Model(M)

Node  
Link  
Group  
...



ElementBo  
X



# Event-Driven

Twaver™

**Element**

```
node.SetStyle(Styles.INNER_COLOR,  
Utils.CreateColor(0xFFFF0000))
```

```
DispatchPropertyChangeEvent
```

**ElementBox**

```
HandleDataPropertyChange
```

```
DispatchEvent
```

**View**

network, tree, chart ...

```
HandleElementPropertyChange
```

```
invalidate element UI
```

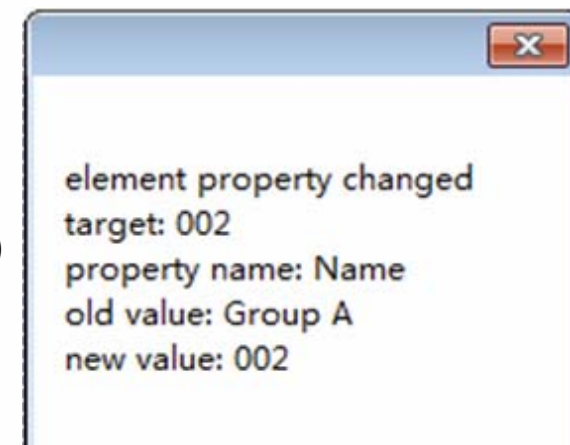


# Event Listeners

TWaver™

```
box.DataPropertyChange += (PropertyChangedEvent<IElement> evt) => {  
    MessageBox.Show("element property changed\ntarget: " + evt.Source +  
        "\nproperty name: " + evt.PropertyName +  
        "\nold value: " + evt.OldValue +  
        "\nnew value: " + evt.NewValue);};
```

```
<Button Content="Change Name" Click="Button_Click" />  
  
private void Button_Click(object sender, RoutedEventArgs e)  
{  
    box.Datas[0].Name = "002";  
}
```



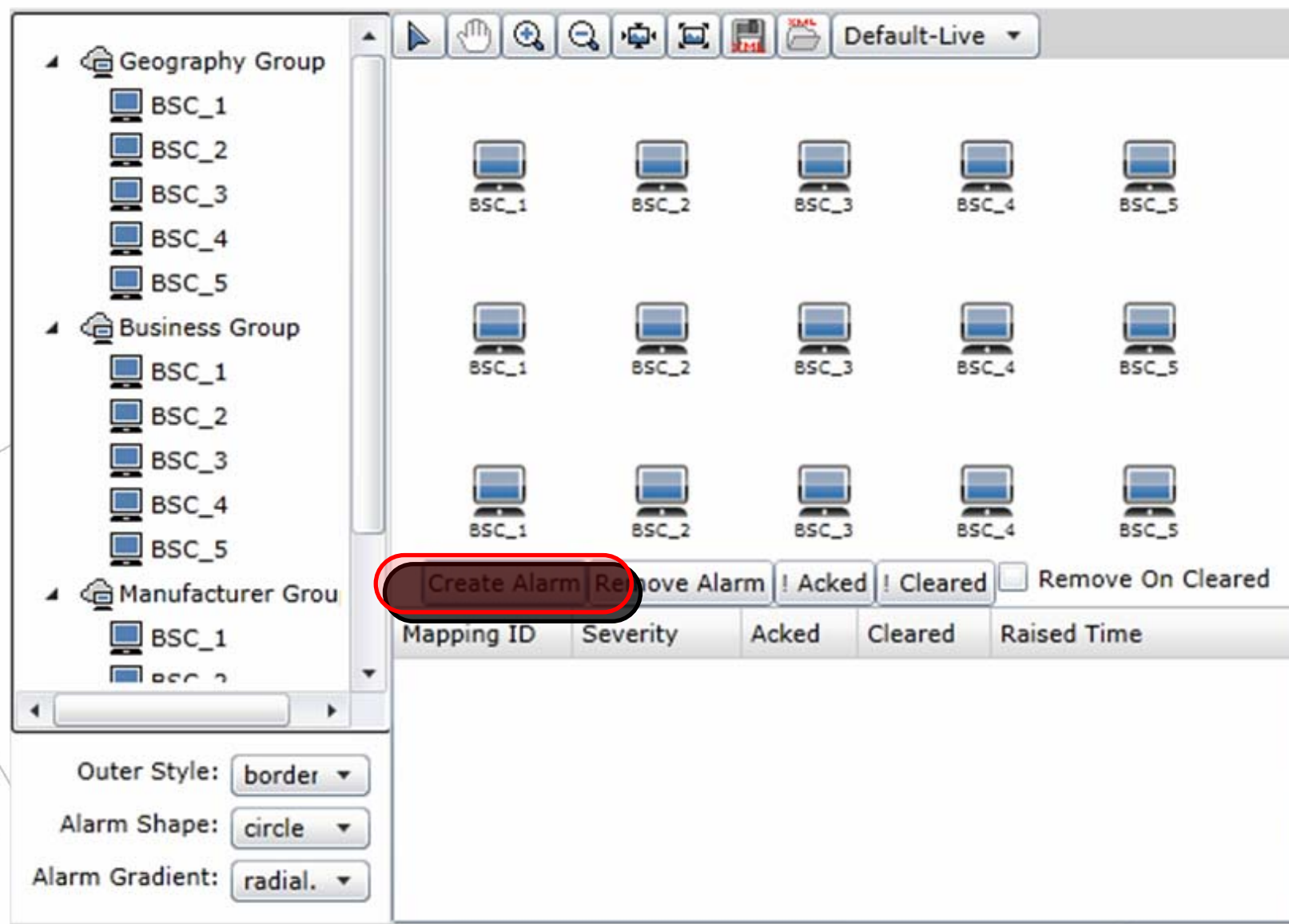
# Event Listeners

TWaver™

Target	Listener
DataBox	PriorDataBoxChange / DataBoxChange / DataPropertyChange / HierarchyChange / PropertyChange
ElementBox	extends DataBox<IElement> IndexChange
AlarmBox	extends DataBox<IAlarm>
LayerBox	extends DataBox<ILayer>
SelectionModel	SelectionChange
Network	Interaction / PropertyChange
AlarmSeverity	AlarmSeverityChange

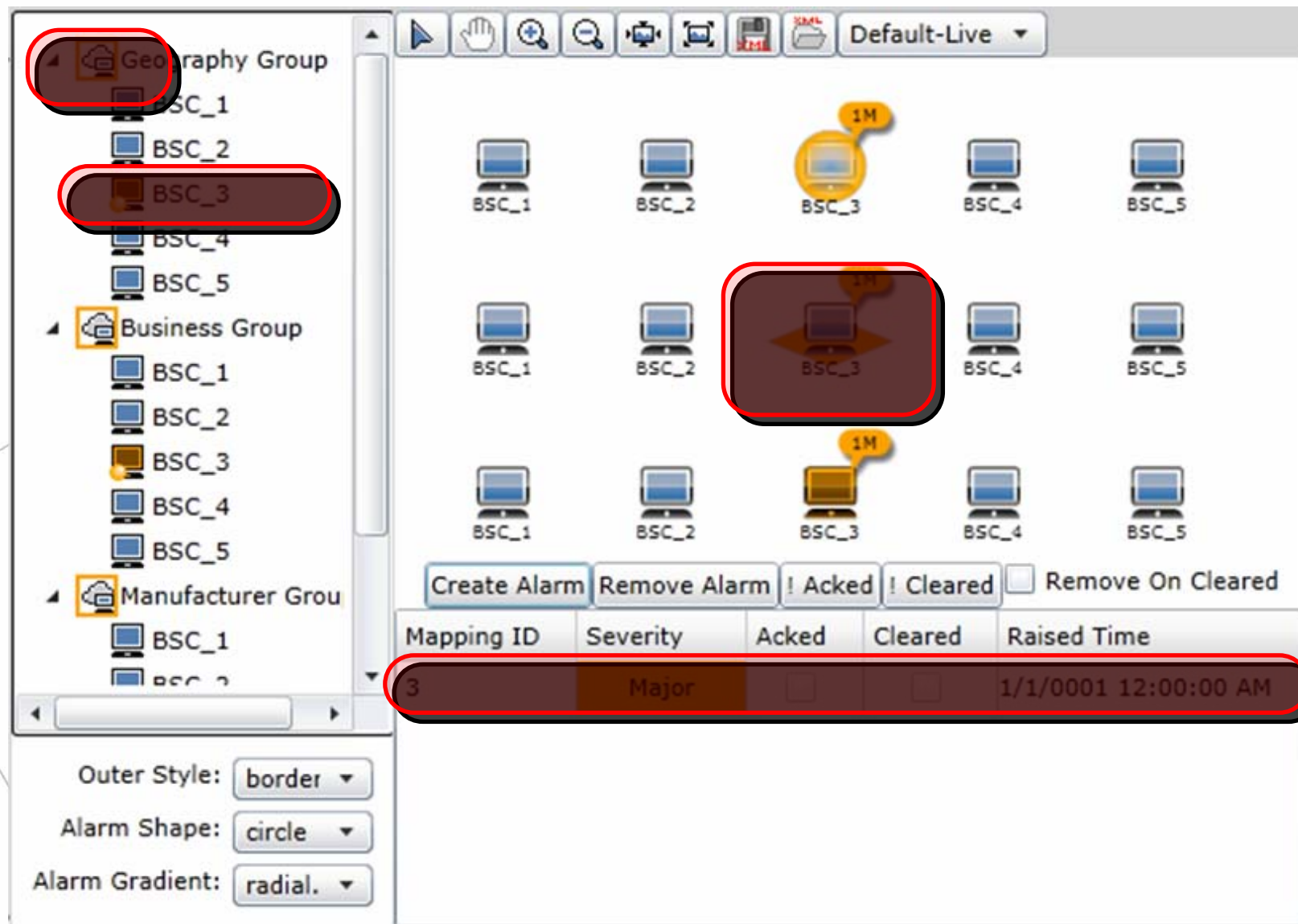
# Views Synchronize Update

TWaver™



# Views Synchronize Update

TWaver™



The screenshot displays the TWaver software interface. On the left, a hierarchical tree structure is shown with three main groups: 'Geography Group', 'Business Group', and 'Manufacturer Group'. Each group contains five 'BSC' (Base Station Controller) items, labeled BSC\_1 through BSC\_5. The 'Geography Group' and 'BSC\_3' are highlighted with red boxes. The main area on the right shows a grid of BSC icons. The 'BSC\_3' icon is highlighted with a red box and a yellow '1M' label. Below the grid, there are buttons for 'Create Alarm', 'Remove Alarm', 'Acked', 'Cleared', and 'Remove On Cleared'. At the bottom, a table displays alarm information.

Mapping ID	Severity	Acked	Cleared	Raised Time
3	Major	<input type="checkbox"/>	<input type="checkbox"/>	1/1/0001 12:00:00 AM

At the bottom left, there are settings for 'Outer Style' (border), 'Alarm Shape' (circle), and 'Alarm Gradient' (radial).



# Data Item & Data Models

**Data Item** : The basic unit of data

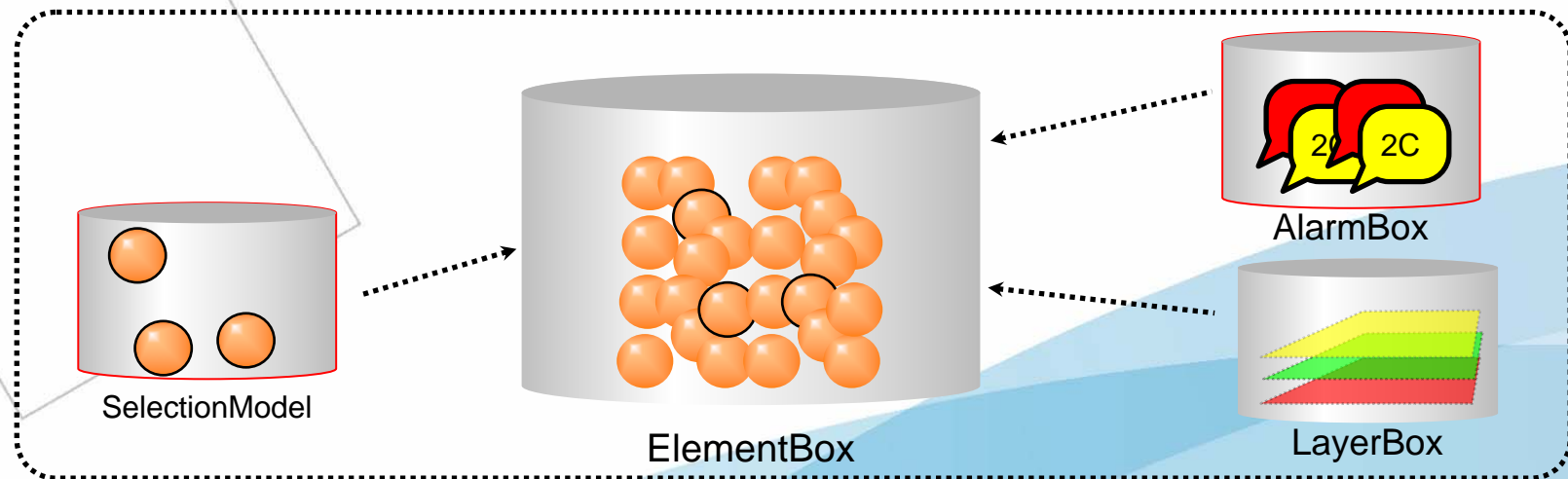
**Data Model** : the data management container, provides add/remove/clear operations, can monitor the property change of data

Data Items	Data Models
IData	DataBox
IElement	ElementBox
IAlarm	AlarmBox
ILayer	LayerBox

# The Relationship Among Data Models

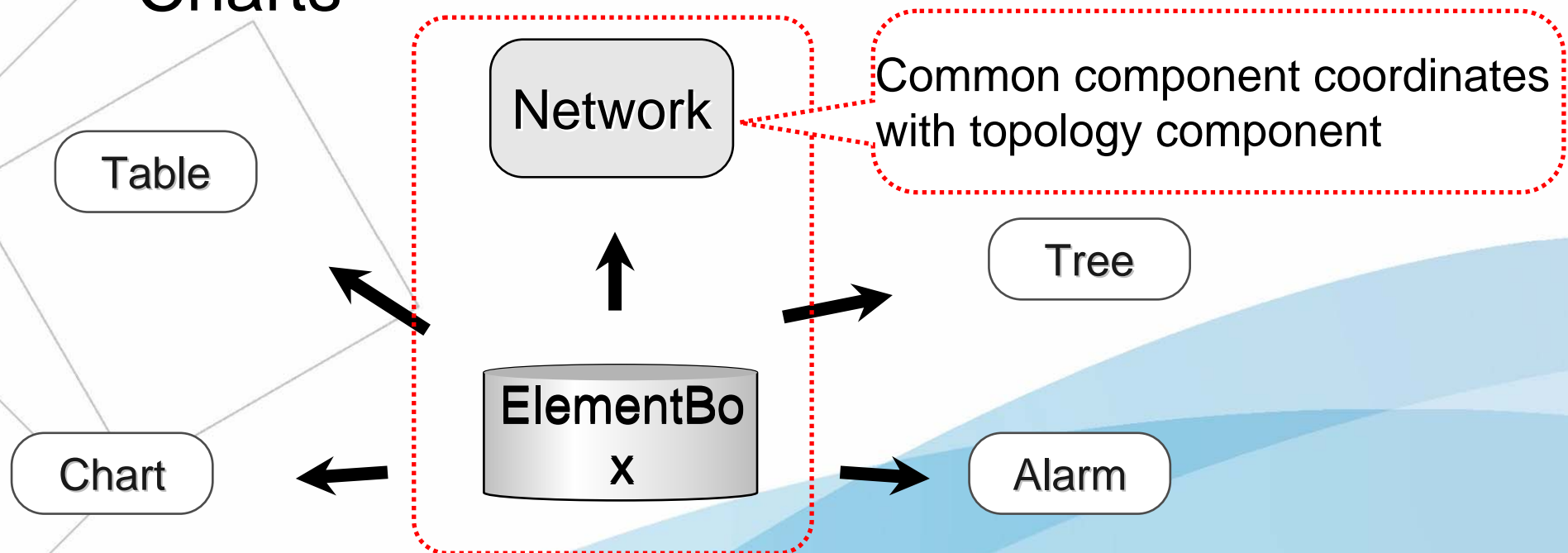
TWaver™

- ElementBox is a data container for managing network elements, it includes a alarm box, a layer box and a selection model



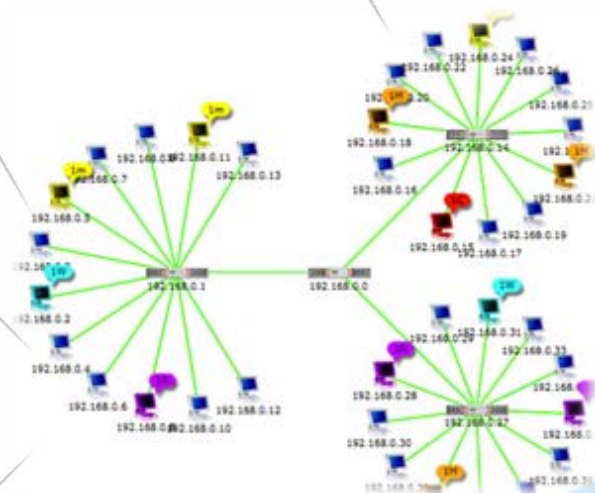
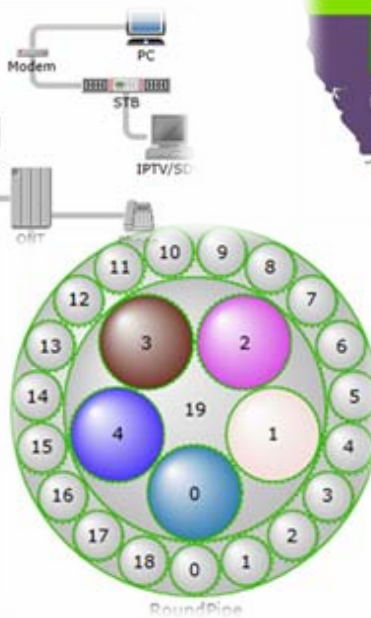
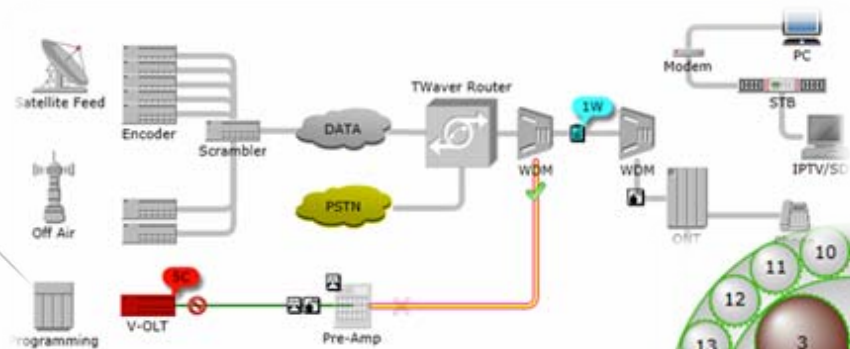
# View Types

- **Topology component - Network**
- **Common components - Tree, Table, Charts**

































# Network

Twaver™

















# Tree

- ▲ Folder Label Folder
  - label
- ▲ Folder Icons Folder
  -     
- ▲ Folder Message Folder
  -  Memory Usage 63%, 190M/512MB
- ▲ Folder Composite Folder
  - label     message
  - label message    
  -     message label
  - message     label
  -     label message
  - message label    

- ▲ TWaver Silverlight Demos
  - ▲ Alarm Demos
    - ☒ Alarm Statistics Demo
    - ☒ Alarm Mapping Demo
    - ☒ Alarm Propagation Demo
  - ▶ Network Demos
    - ☒ Topology Demos
  - ▲ Equipment Demos
    - ☐ EMS Demo
    - ☐ Chassis Demo
  - ▲ Tree Demos
    - ☐ Tree Layout Demo
    - ☐ File Tree Demo

- ▲ TWaver Silverlight Demos
  - ▲ Alarm Demos
    -  Alarm Statistics Demo
    -  Alarm Mapping Demo
    -  Alarm Propagation Demo
  - ▶ Network Demos
  - ▲ Chart Demos
    -  Pie Chart Demo NEW
    -  Bar Chart Demo NEW
    -  Line Chart Demo NEW
    -  Bubble Chart Demo NEW
    -  Radar Chart Demo NEW
    -  Dial Chart Demo NEW
  - Editor Demos
    -  Grid Editor Demo
    -  Pipe Editor Demo
    -  Topology Editor Demo







TWaver™



# Table

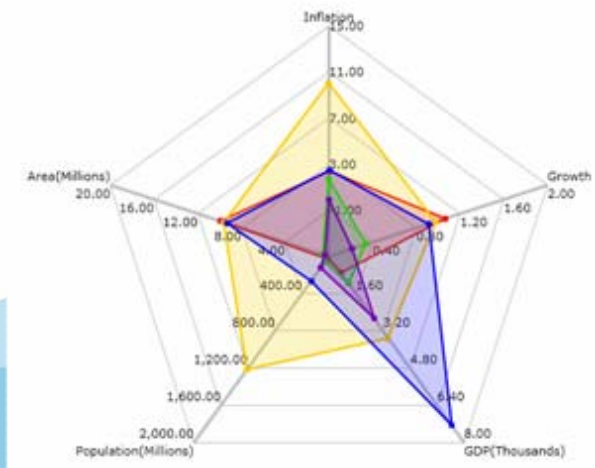
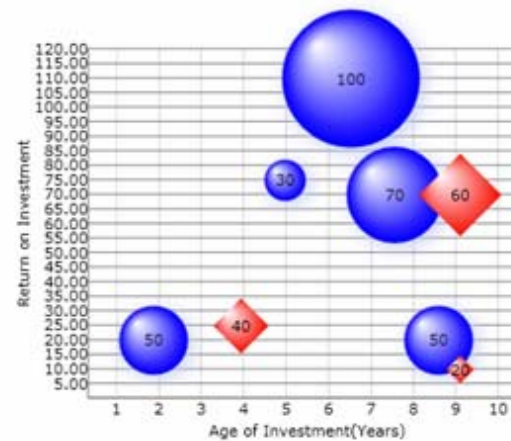
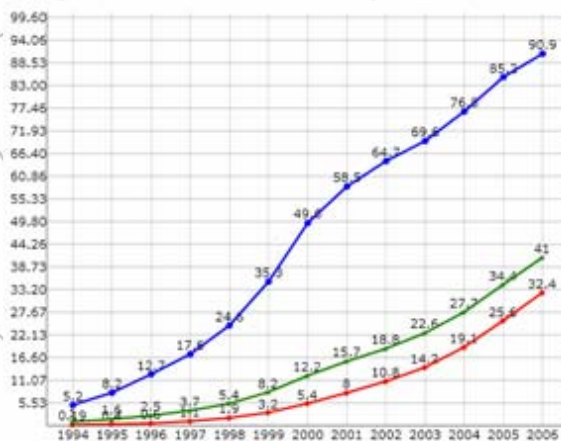
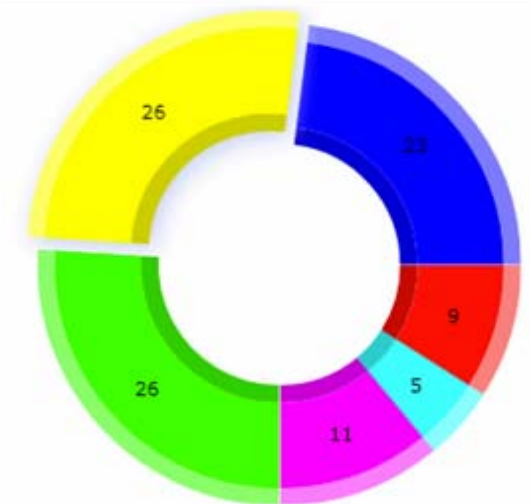
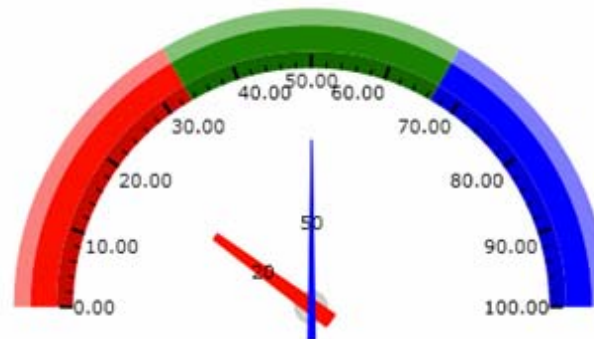
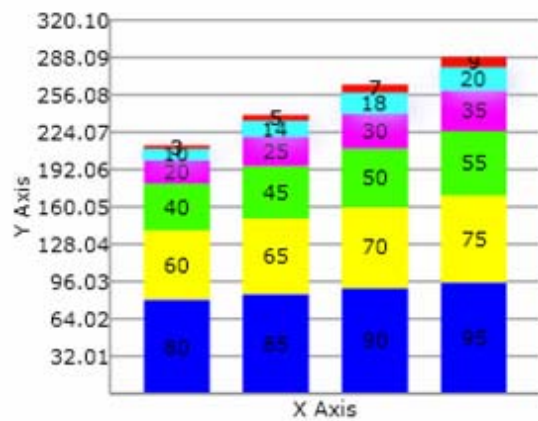
State	Employed	Unemploy	Male	Female
Michigan	4166196	374341	4512781	4782516
Texas	7687338	590269	8433346	8688674
Kentucky	1970934	148125	2195130	2356394
Ohio	4524383	324867	4816445	5164442

Mapping ID	Severity	Acked	Cleared	Raised Time	627	14862394
1	Indeterminate	<input type="checkbox"/>	<input type="checkbox"/>	8/22/2011 3:04:47 PM	3	5878369
2	Warning	<input type="checkbox"/>	<input type="checkbox"/>	8/22/2011 3:04:47 PM	4	2503774
3	Minor	<input type="checkbox"/>	<input type="checkbox"/>	8/22/2011 3:04:47 PM		
4	Major	<input type="checkbox"/>	<input type="checkbox"/>	8/22/2011 3:04:47 PM		
5	Critical	<input type="checkbox"/>	<input type="checkbox"/>	8/22/2011 3:04:47 PM		

Icon	ID	Alarm Color	Tree Label	Network Label
	13f965c5-0f35-48ef	#FF000000	boy1	Sam
	233b4926-c8a0-4cc	#FF000000	boy2	Paul
	8013ae51-7c54-427	#FF000000	girl1	
	cafa8abb-74c1-45c	#FF000000	girl2	
	cb57a4a0-3375-471	#FF000000	girl3	
	d7d3e832-aa04-4af	#FF000000	boy3	Eric

TWaver™

# Chart



TM  
Tmaver

# Comparison with TWaver Java

TWaver™

- There are many similarities between TWaver .NET and TWaver Java
- Class comparison
- Element properties comparison
- Network component comparison

# Classes Comparison

TWaver™

TWaver .NET	TWaver Java	Description
ElementBox	TDataBox	Element container
AlarmBox	AlarmModel	Alarm container
LayerBox	LayerModel	Layer container
Network	TNetwork	Topology component
Tree	TTree	Tree component
Table	TElementTable	Table component



# Element Properties Comparison

TWaver™

TWaver .NET	TWaver Java	Description
node.Name = "001"	node.setName("001")	properties
node.SetStyle	node.putClientProperty	styles
node.SetClient	node.putUserProperty	client properties



# Network Using Comparison

TWaver .NET	TWaver Java	Description
AlarmLabelFunction	alarmLabelGenerator	alarm bubble label
VisibleFunction	visibleFilter	visible filter
MovableFunction	movableFilter	movable filter
EditableFunction	elementLabelEditableFilter	label editable filter
LabelFunction	elementLabelGenerator	label text generator
ToolTipFunction	elementToolTipTextGenerator	tooltip text generator
InnerColorFunction	elementBodyColorGenerator	body color generator
OuterColorFunction	elementOutlineColorGenerator	outline color generator
SelectColorFunction	elementSelectColorGenerator	selection color generator
AlarmFillColorFunction	alarmColorGenerator	alarm bubble color



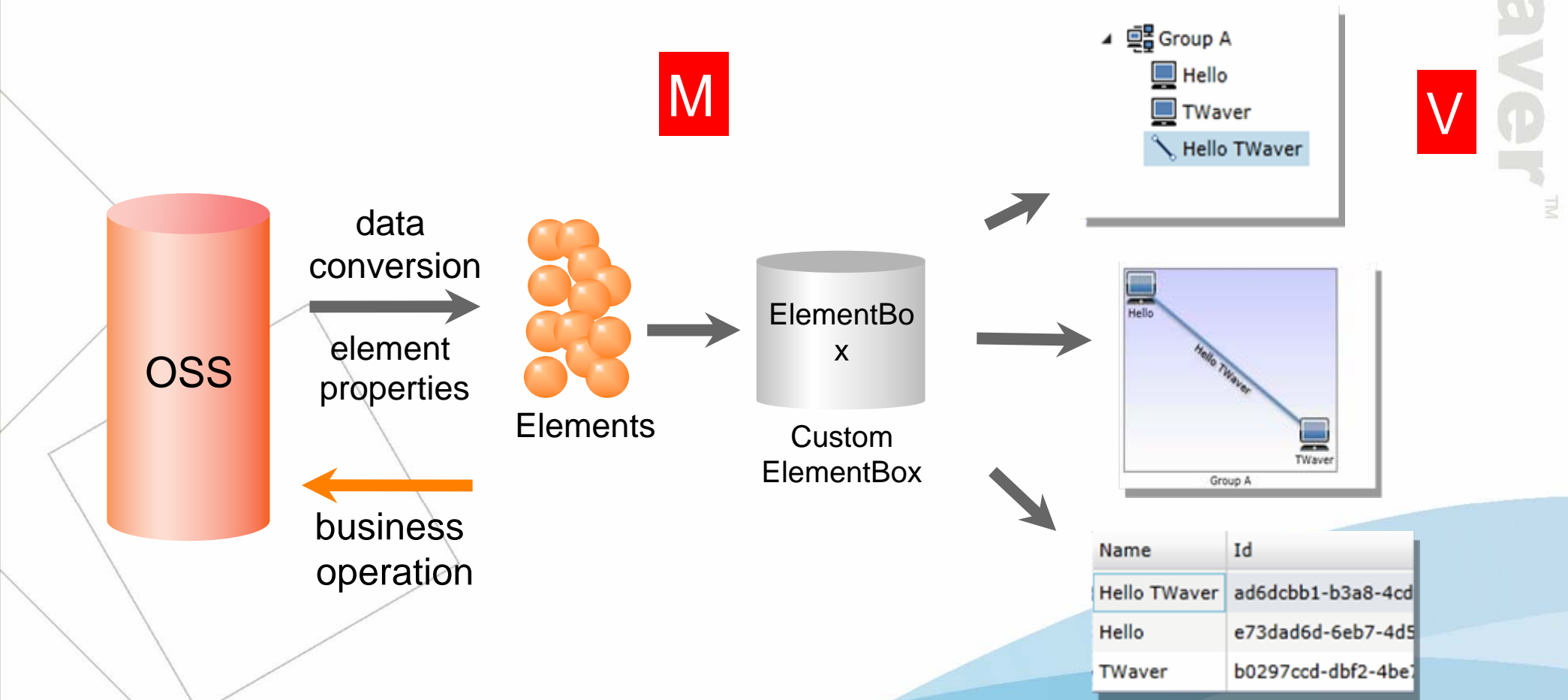
- Home - ServaSoftware.com
- Email - [tw-service@servasoft.com](mailto:tw-service@servasoft.com)

# TWaver .NET Core Components

TWaver™

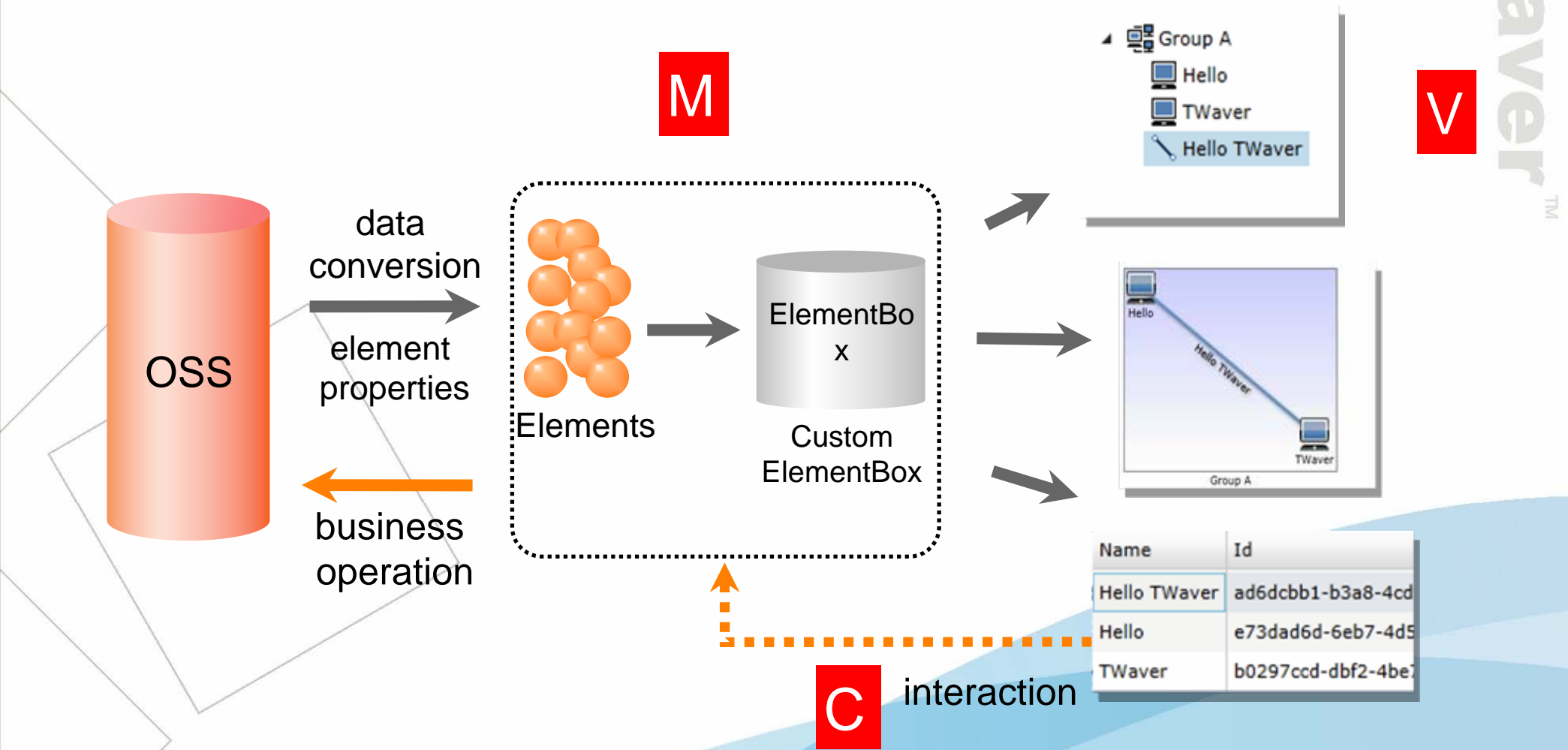
- Development process
- Using DataBox & ElementBox
- Predefined elements
- Using Network

# TWaver .NET Development Process



TWaver™

# TWaver .NET Development Process



TWaver™



# TWaver .NET Development Process

TWaver™

```
ElementBox box = new ElementBox();
//data acquisitionArray
devices = GetDevicesFromOSS();
Array relationships = GetDevicesRelationshipFromOSS();
//data translate
TranslateToTWaverNode(box, devices, relationships);
Network network = new Network(box);
this.LayoutRoot.Children.Add(network);
AutoLayouter layouter = new AutoLayouter(network);
layouter.IsAnimated = false;
layouter.DoLayout(Constants.LAYOUT_SYMMETRY);
//add interaction
network.Interaction += (InteractionEvent evt) =>{
    if (evt.Kind != InteractionEvent.DOUBLE_CLICK_ELEMENT) {
        return;
    }
    IElement element = evt.Element;
    ShowInputDialog("Rename", element.Name, (string result) => {
        element.Name = result;
    });
};
```

# TWaver .NET Development Process

TWaver™

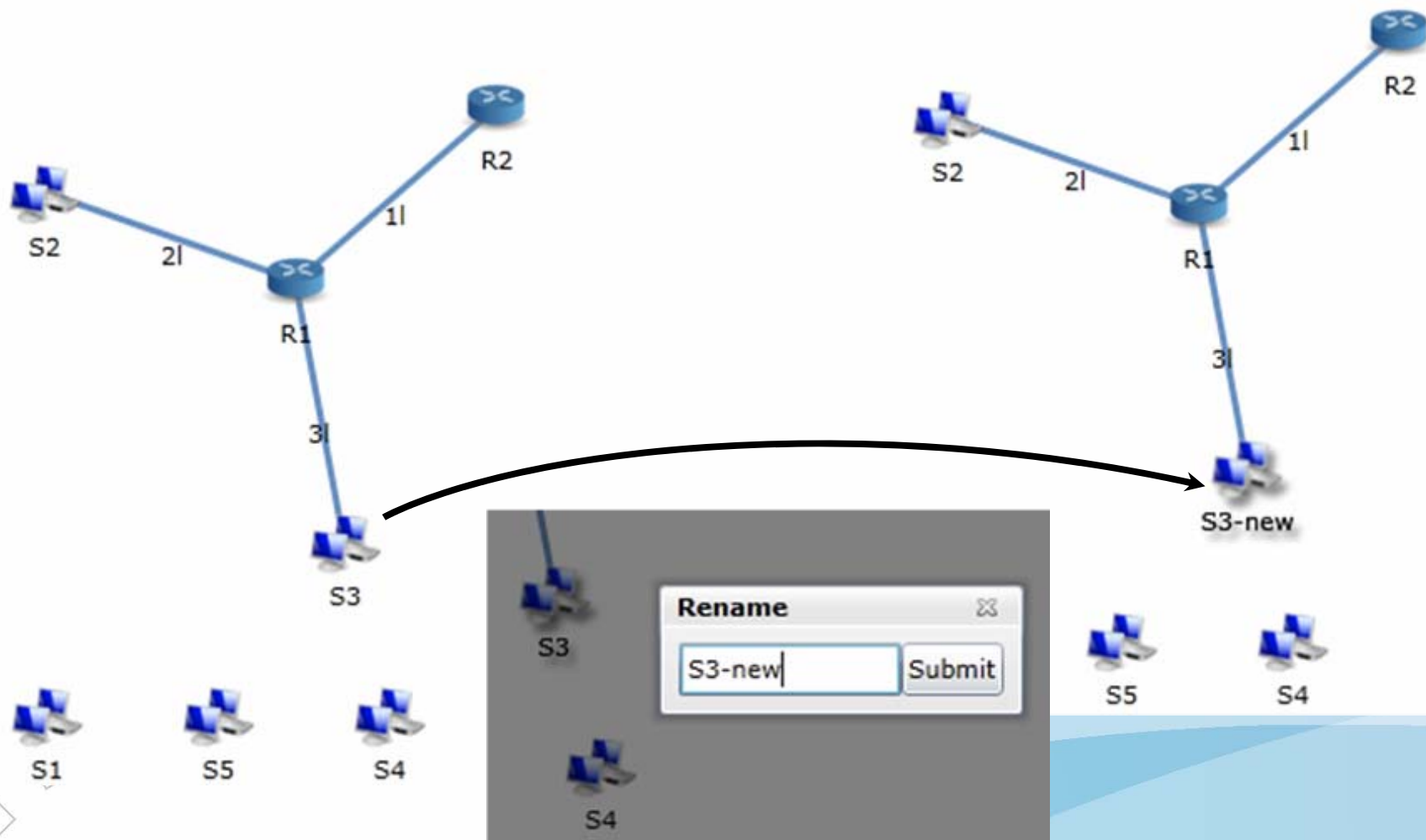
```
private Array GetDevicesFromOSS() {
    return new object[] { new { name = "R1", type = "router" },
        new { name = "R2", type = "router" },
        new { name = "S1", type = "device" },
        new { name = "S2", type = "device" },
        new { name = "S3", type = "device" },
        new { name = "S4", type = "device" },
        new { name = "S5", type = "device" } };
}

private Array GetDevicesRelationshipFromOSS() {
    return new object[] { new { name = "1I", from = "R1", to = "R2" },
        new { name = "2I", from = "R1", to = "S2" },
        new { name = "3I", from = "R1", to = "S3" } };
}

private void TranslateToTWaverNode(ElementBox box, Array devices, Array relationships) {
    foreach (dynamic device in devices) {
        Node node = new Node(device.name);
        node.Name = device.name;
        node.Image = device.type;
        box.Add(node);
    }
    foreach (dynamic relationship in relationships) {
        Link link = new Link((Node)box.GetDataByID(relationship.from), (Node)box.GetDataByID(relationship.to));
        link.Name = relationship.name;
        box.Add(link);
    }
}
```

# TWaver .NET Development Process

TWaver™

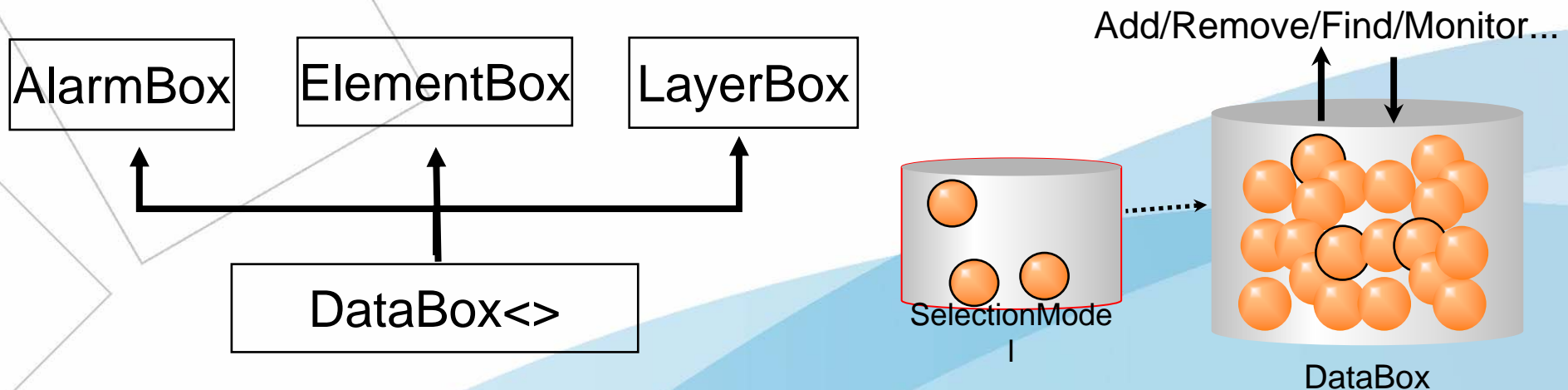


# Using DataBox

DataBox - Data container for managing data items.

- Provides basic operations like add, delete, edit
- Monitors elements changes and other events
- It's the base class for ElementBox, AlarmBox, LayerBox
- It contains selection model

DataBox#**public** SelectionModel SelectionModel



# Basic Operations

Twaver™

- Add data

**public virtual void** Add(TData data)

**public virtual void** Add(TData data, int index)

- Delete data

**public virtual void** Remove(TData data)

**public virtual void** RemoveSelection()

**public virtual void** RemoveByID(Object id)

- Clear data

**public virtual void** Clear()



# Getting Elements

TWaver™

- Get data

**public virtual** TData GetDataByID(Object id)

**public virtual** IList<TData> Datas()

**public virtual** IList<TData> ToDatas(Predicate<TData> match)

**public virtual** IList<TData> Roots()

**public virtual** IList<TData> GetSiblings(TData data)

- Traverse data

**public virtual void** ForEach(Action<TData> callbackFunction):

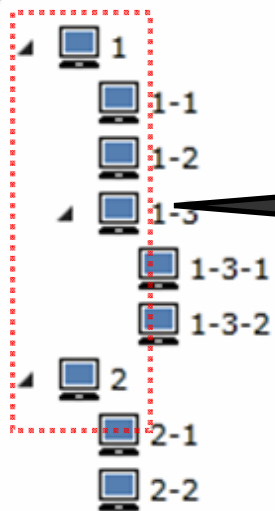
**public virtual void** ForEachByDepthFirst(Action<TData> callbackFunction,  
TData data = **null**):

**public virtual void** ForEachByBreadthFirst(Action<TData>  
callbackFunction, TData data = **null**)

# Hierarchy of Elements

- Element hierarchy is built by parent-child relationships
- Move\*\*\*methods of DataBox are used to adjust element index

TWaver™



node 1 ,2 at the root  
level (Roots)

- MoveDown(TData)
- MoveSelectionDown()
- MoveSelectionDown(TWaver.SelectionModel<TData>)
- MoveSelectionToBottom()
- MoveSelectionToBottom(TWaver.SelectionModel<TData>)
- MoveSelectionToTop()
- MoveSelectionToTop(TWaver.SelectionModel<TData>)
- MoveSelectionUp()
- MoveSelectionUp(TWaver.SelectionModel<TData>)
- MoveTo(TData, int)
- MoveToBottom(TData)
- MoveToTop(TData)
- MoveUp(TData)

# Quick Finder

## QuickFinder:

To find elements by name or other property of element, for example:

```
QuickFinder<IData> finder = new QuickFinder<IData>(box, "age");  
IList<IData> elements = finder.Find("PC");
```

## Create a quick finder:

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

## Quick finder types:

Consts.PROPERTY\_TYPE\_ACCESSOR — find by accessor

propertyConsts.PROPERTY\_TYPE\_CLIENT — find by client

propertyConsts.PROPERTY\_TYPE\_STYLE — find by style property

TM  
TWAVER

# Quick Finder Example

TWaver™

```
DataBox<IData> box = new DataBox<IData>();  
for (int i = 0; i < 30; i++)  
{  
    Node node = new Node();  
    node.Name = "Node-" + i;  
    node.SetClient("age", i % 10);  
    box.Add(node);  
}  
QuickFinder<IData> finder = new QuickFinder<IData>(box, "age",  
    Consts.PROPERTY_TYPE_CLIENT);  
IList<IData> elements = finder.Find(8);  
foreach (IData node in elements)  
{  
    Trace(node.Name + "'s age is " + node.GetClient("age"));  
}
```

trace:

Node-8's age is 8

Node-18's age is 8

Node-28's age is 8

# Listeners in DataBox

Add listener - DataBox.DataPropertyChange += listener

Delete listener - DataBox.DataPropertyChange -= listener

box.DataPropertyChange += new

```
EventListener<PropertyChangeEvent<IData>>((PropertyChangeEvent<IData> evt) =>{  
    MessageBox.Show(evt.Source.Name + "'s property changed");  
});
```

Listener	Description
DataBoxChange	detect element's add, remove, clear
PriorDataBoxChange	before dataBox change
PropertyChange	detect dataBox's property change
DataPropertyChange	detect element's property change
HierarchyChange	detect dataBox's sequence change



# Listener Example

```
DataBox<IData> box = new DataBox<IData>();
```

```
box.DataPropertyChange += new  
EventListener<PropertyChangeEvent<IData>>((PropertyChangeEvent<IData> evt) =>{  
    Trace(evt.Source.Name + "'s property changed");  
});  
box.DataBoxChange += new  
EventListener<DataBoxChangeEvent<IData>>((DataBoxChangeEvent<IData> evt) => {  
    Trace("data " + evt.Kind + ": " + evt.Data.Name);  
});
```

```
Data data = new Data();  
data.Name = "001";  
//add data  
box.Add(data);  
//modified data property  
data.Name = "002";  
//remove data  
box.Remove(data);
```

```
output:  
data add: 001  
002's property changed  
data remove: 002
```

# Import & Export DataBox

Twaver™

- DataBox can both import and export xml
- The export xml contains properties of all element and DataBox' s own properties
- Import and export by XMLSerializer

# Import & Export DataBox

TWaver™

- XMLSerializer - import & export

- **public** XMLSerializer(DataBox<TData> dataBox, SerializationSettings settings, Predicate<TData> filterFunction))
- **public** String Serialize():
- **public void** Deserialize(string xmlString, IData rootParent)

- SerializationSettings - import & export setting

- **public void** RegisterProperty(string property, string type, bool cdata)
- **public void** RegisterStyle(string styleProp, string type, bool cdata)
- **public void** RegisterClient(string clientProp, string type, bool cdata)

# DataBox Export Example

```
ElementBox box = network.ElementBox;
```

```
SubNetwork parent = new SubNetwork();  
parent.Name = "Parent";  
network.ElementBox.Add(parent);
```

```
Node node1 = new Node();  
node1.Location = new Point(50, 50);  
node1.Name = "Node1";  
node1.SetClient("age", 27);  
node1.Parent = parent;  
network.ElementBox.Add(node1);
```

```
SerializationSettings settings = new SerializationSettings();  
settings.RegisterProperty("ID", null, false);  
settings.RegisterClient("age", Consts.TYPE_INT, false);  
settings.RegisterProperty("name", Consts.TYPE_STRING, true);  
settings.RegisterProperty("parent", Consts.TYPE_DATA, false);  
serializer = new XMLSerializer<IElement>(box, settings);
```

```
textBox.Text = serializer.Serialize();
```

```
<?xml version="1.0" encoding="utf-16"?>  
<TWaver V="2.0" P="SILVERLIGHT">  
  <DataBox Type="TWaver.ElementBox">  
    <LayerBox>  
      <Layer Name="default" IsVisible="True"  
IsEditable="True" IsMovable="True" />  
    </LayerBox>  
  </DataBox>  
  <Data Type="TWaver.SubNetwork" Ref="0">  
    <P N="Name"><![CDATA[Parent]]></P>  
  </Data>  
  <Data Type="TWaver.Node" Ref="1">  
    <C N="age">27</C>  
    <P N="Name"><![CDATA[Node1]]></P>  
    <P N="Parent" Ref="0" />  
    <P N="Location" X="50" Y="50" />  
  </Data>  
</TWaver>
```

# DataBox Import Example

TWaver™

```
SerializationSettings settings = new SerializationSettings();  
settings.RegisterProperty("ID", null, false);  
settings.RegisterClient("age", Consts.TYPE_INT, false);  
settings.RegisterProperty("name", Consts.TYPE_STRING, true);  
settings.RegisterProperty("parent", Consts.TYPE_DATA, false);  
serializer = new XMLSerializer<IElement>(box, settings);
```

```
textBox.Text = serializer.Serialize();
```

```
string xml = textBox.Text;  
box.Clear();  
serializer.Deserialize(xml);
```

```
textBox.Text = "box count: " + box.Count;  
textBox.Text += "\nthe first node's name: " + box.Datas[0].Name;  
textBox.Text += "\nthe second node's name: " + box.Datas[1].Name;
```

output:  
box count: 2  
the first node's name: Parent  
the second node's name: Node1



# Using ElementBox

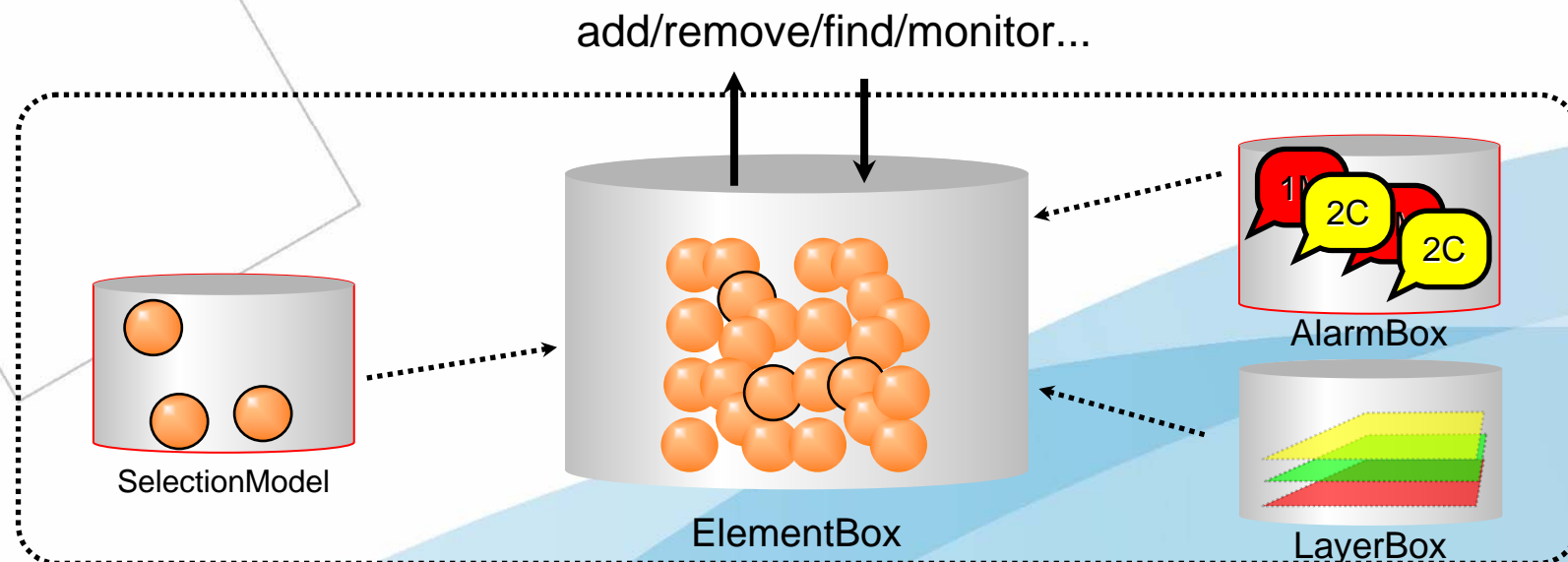
**ElementBox** - the data container for topology elements management

Inherit from DataBox<IElement>, retain the functions of DataBox, increase layer box and alarm box

ElementBox#

public LayerBox LayerBox

public AlarmBox AlarmBox



# Using ElementBox

- Increase LayerBox、AlarmBox

```
public LayerBox LayerBox
```

```
public AlarmBox AlarmBox
```

- Traverse elements by layer

```
public void ForEachByLayer(Action<IElement> callbackFunction, ILayer layer)
```

```
public void ForEachByLayerReverse(Action<IElement> callbackFunction, ILayer layer)
```

- Increase layer change listener

```
public event EventHandler<IndexChangedEvent> IndexChange
```

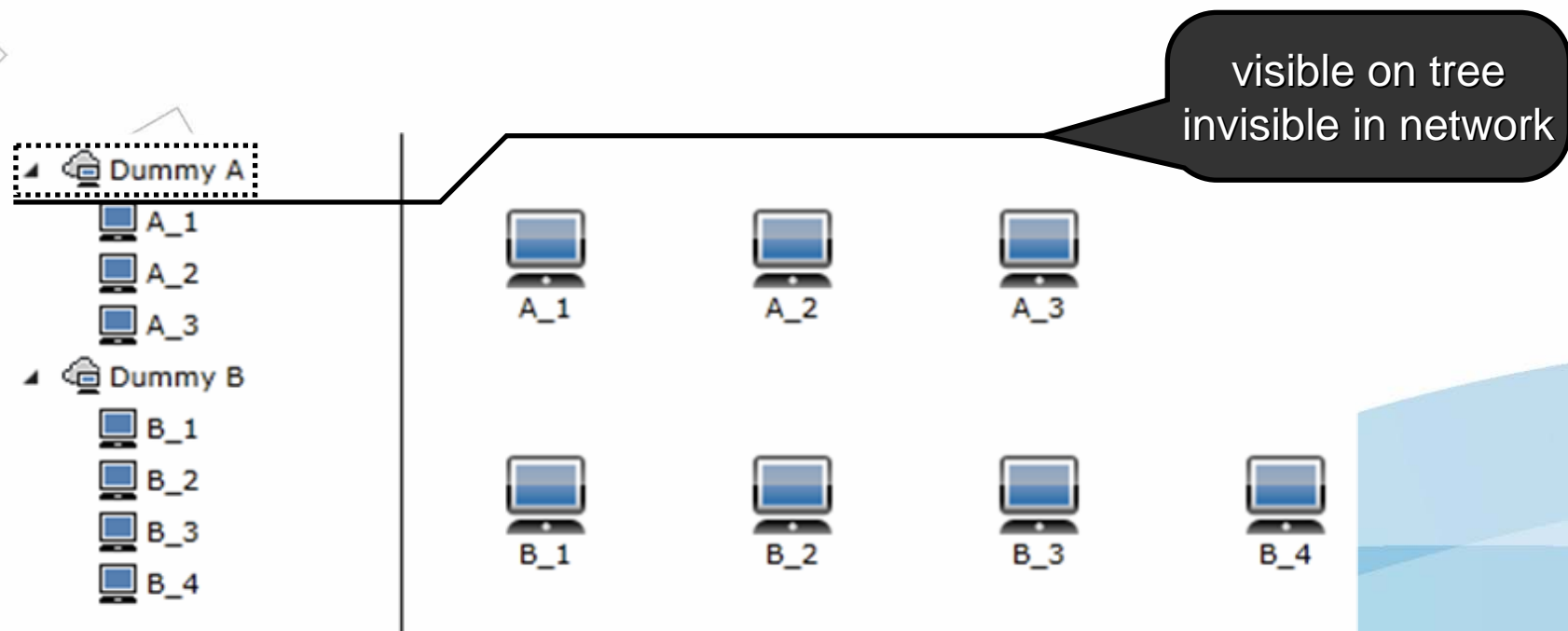
# Elements

## Data

- | -- Alarm
- | -- Layer
- | -- Element
  - | -- Dummy
  - | -- Node
    - | -- Follower
    - | -- Group
    - | -- Grid
    - | -- ShapeNode
    - | -- Bus
    - | -- SubNetwork
  - | -- Link
    - | -- ShapeLink
    - | -- LinkSubNetwork

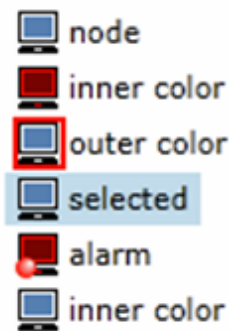
# Dummy

- Dummy is invisible on network, but visible on Tree and TableDummy can be used to reorganize tree structure and avoid to affect network



# Node

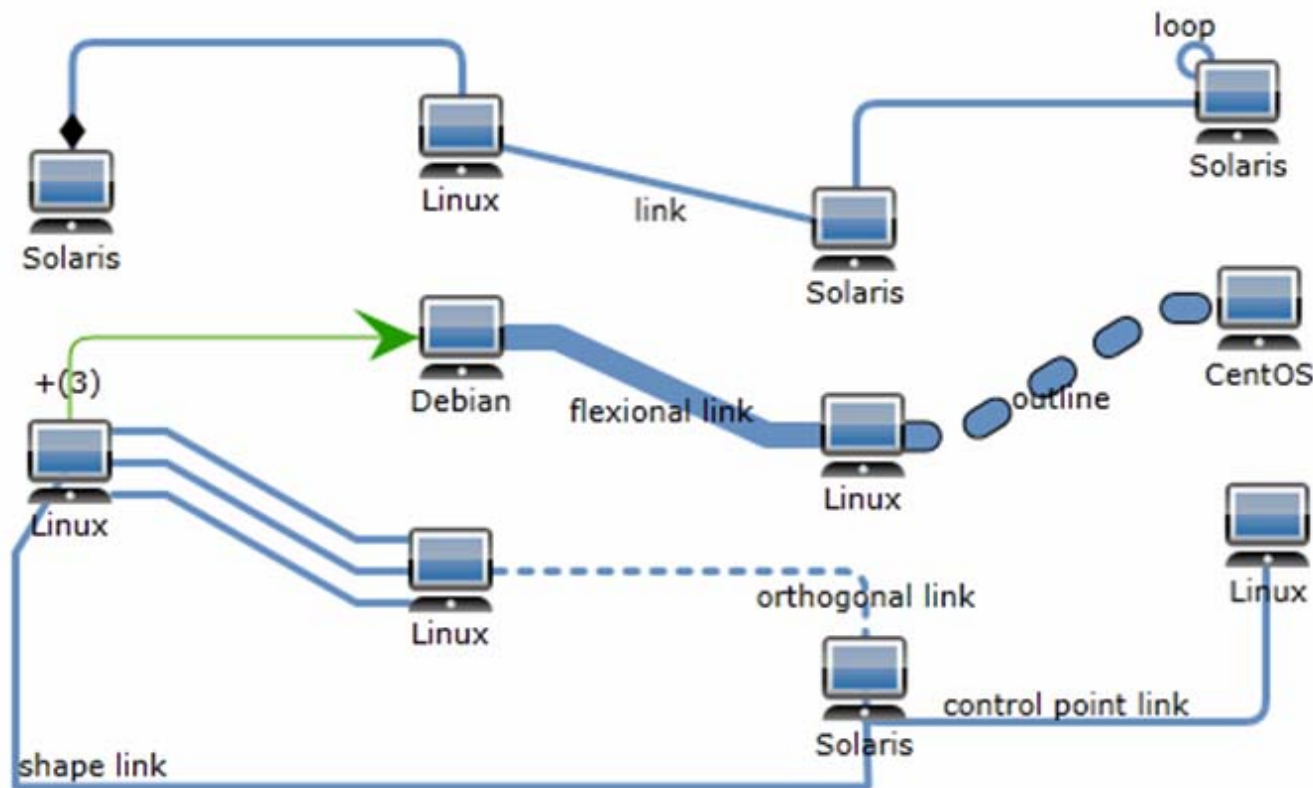
- Basic node type, can set image, outline, render color ...
- TWaver.Node is the base class for other elements





# Link

- Link, linking two nodes, it has many types and styles, it provides arrows, loop, flowing etc.

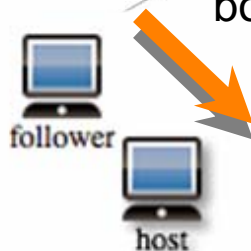


# Follower

- Follower, it can set a host node, follower will follow it when host is moving.

```
Follower node = new Follower();  
node.Location = new Point(100, 100);  
node.Name = "follower";  
box.Add(node);
```

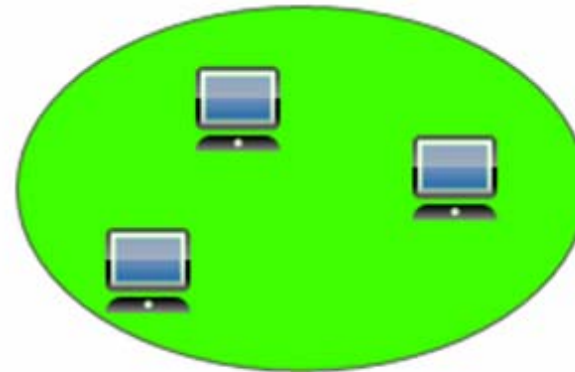
```
Node host = new Node();  
host.Location = new Point(140, 140);  
node.Host = host;  
host.Name = "host";  
box.Add(host);
```



host move, and follower move, too.

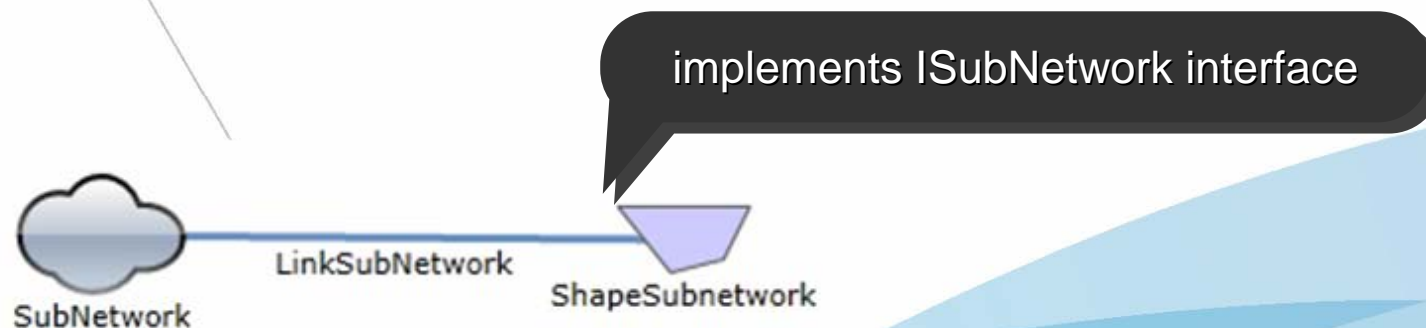
# Group

- Group, contains other nodes, it can close or open, it has many types of shapes



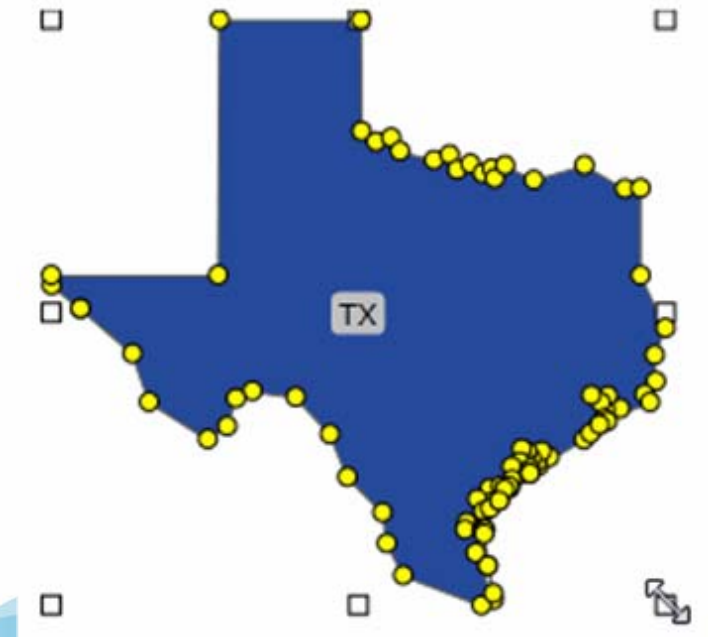
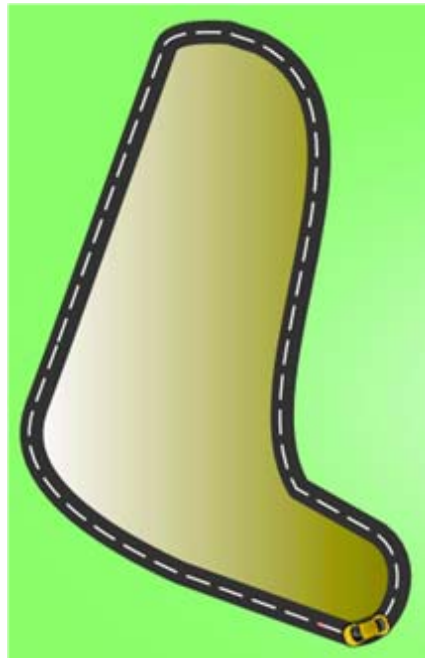
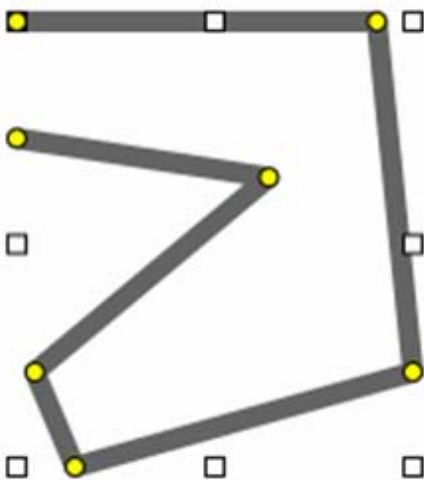
# SubNetwork

- Subnetwork, you can drop in or out by double click, when you enter into a subnetwork, the children of the subnetwork will be displayed, Subnetwork has its own background



# ShapeNode

- ShapeNode, surrounded by a number of control points.

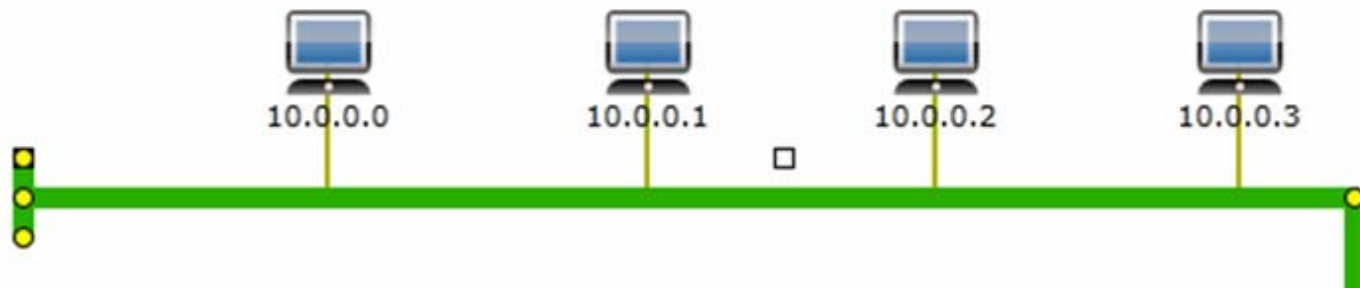




# Bus

TWaver™

- Inherited from ShapeNode, connected by a set of control points into line, the links that connected to it are orthogonal distribution



# Grid

- Grid, displays child elements in rows and columns, one grid can embed into an other grid, can be used for creating equipment chassis

```
TWaver.Grid grid = new TWaver.Grid();
grid.Name = "Grid B";
grid.SetStyle(Styles.GRID_BORDER, 5);
grid.SetStyle(Styles.GRID_COLUMN_COUNT, 4);
grid.SetStyle(Styles.GRID_COLUMN_PERCENTS, new double[]
{ 0.15, 0.2, 0.25, 0.4 });
grid.SetStyle(Styles.GRID_ROW_COUNT, 2);
grid.SetStyle(Styles.GRID_ROW_PERCENTS, new double[]
{ 0.6, 0.4 });
grid.SetStyle(Styles.GRID_FILL_COLOR,
Utils.CreateColor(0xFFCC6600));
grid.SetStyle(Styles.GRID_DEEP, 11);
grid.SetStyle(Styles.GRID_CELL_DEEP, -4);
grid.SetStyle(Styles.GRID_PADDING, 0);
grid.Width = 250;
grid.Height = 106;
```



Grid A



Grid B



# Element Properties

twaver.Consts#

```
public const string PROPERTY_TYPE_ACCESSOR =  
"accessor"; public const string PROPERTY_TYPE_CLIENT = "client";  
public const string PROPERTY_TYPE_STYLE = "style";
```

Element

**accessor** - name, id ...

node.Name = "001";

**style** - style properties

node.SetStyle(Styles.INNER\_COLOR,  
Utils.CreateColor(0xFFCC6600));

**client** - client properties

node.SetClient("age", 27);

# Using Network

- Network hierarchy
- Network background
- Network interactions
- Network filters
- Network generators
- Network auto layout

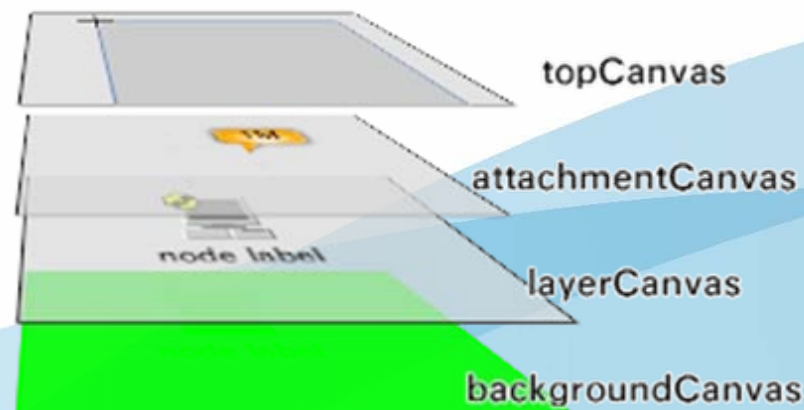
TM  
T  
W  
a  
v  
e  
r  
™

# Network Hierarchy

ScrollView

NetworkCanvas

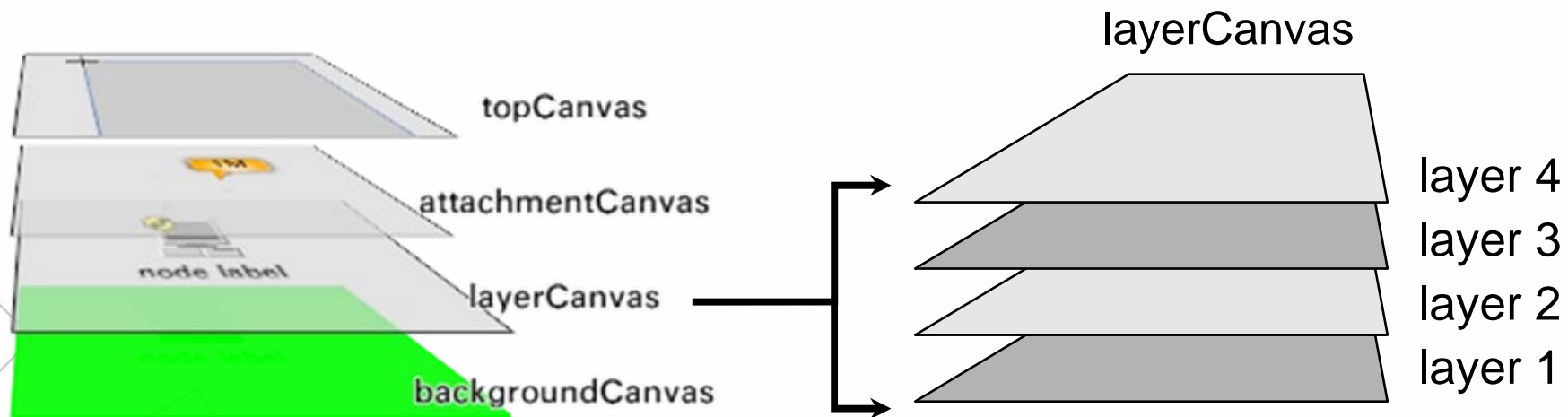
- RootCanvas — main canvas, contains all topology components
  - TopCanvas — can be used for painting selection rectangle
  - AttachmentCanvas — contains top attachments, like alarm bubble
  - LayerCanvas — contains all nodes and links
    - Layer n
    - Layer ...
    - Default layer
  - BottomCanvas — bottom canvas, painting shading under the background
  - BackgroundCanvas — color background or image background





# Layer Management

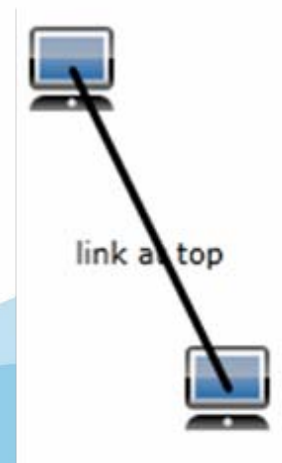
TM  
waver



```
ElementBox box = network.ElementBox;
LayerBox layerBox = box.LayerBox;
```

```
Layer layerTop = new Layer("top", "top");
layerBox.Add(layerTop, layerBox.Count);
```

```
Node node = new Node();
box.Add(node);
Node node2 = new Node();
box.Add(node2);
Link link = new Link(node, node2);
link.LayerID = layerTop.ID;
link.Name = "link at top";
link.SetStyle(Styles.LINK_COLOR, Utils.CreateColor(0xFF000000));
box.Add(link);
```



# Adding Background

TWaver™

- Can set the background to ElementBox and subnetworks, supports color, image or gradient background.

```
Utils.RegisterPNGImage("background", new  
    Uri("/TWaverPPT;component/images/shanghai.png", UriKind.Relative));  
box.SetStyle(Styles.BACKGROUND_TYPE, Consts.BACKGROUND_TYPE_IMAGE);  
box.SetStyle(Styles.BACKGROUND_IMAGE, "background");
```

- .NET component also supports its own background styles

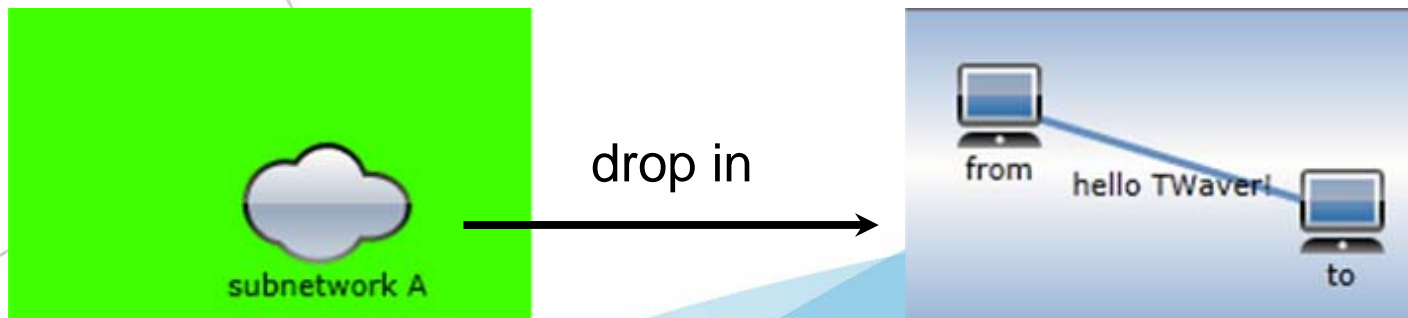
```
network.Background = new SolidColorBrush(Color.FromArgb(255, 0, 255,  
0));
```

# Background Example

TWaver™

```
Network network = new Network();network.Background = new  
SolidColorBrush(Color.FromArgb(255, 0, 255, 0));
```

```
SubNetwork subnetworkA = AddSubnetwork(box, "subnetwork A");  
subnetworkA.Location = new Point(100, 100);  
subnetworkA.SetStyle(Styles.BACKGROUND_COLOR, Utils.CreateColor(0xFF9CB5D8));  
subnetworkA.SetStyle(Styles.BACKGROUND_GRADIENT_COLOR,  
Utils.CreateColor(0xFFFFFFFF));  
subnetworkA.SetStyle(Styles.BACKGROUND_GRADIENT,  
Consts.GRADIENT_SPREAD_VERTICAL);
```



# Network Interaction Mode

- **InteractionHandler** - single listener
- **Interaction Mode** - several input handlers combined into an interaction mode

Network#

```
public IList<InteractionHandler> InteractionHandlers
```

- **Predefined Modes** - Network provides several predefined interaction modes, such as the default mode, edit mode, create link mode ...

Network#

```
public void SetDefaultInteractionHandlers(bool lazyMode)
```

```
public void SetEditInteractionHandlers(bool lazyMode)
```

```
public void SetCreateLinkInteractionHandlers(Type type)
```

...

# Custom Interaction

Interaction handler class

```
public class HighlightInputHandler : BasicInteractionHandler {  
    public HighlightInputHandler(Network network) : base(network) { }  
    override public void InstallListeners(){  
        this.NETwork.NETworkCanvas.MouseMove += HandleMouseMove;  
    }  
    override public void UninstallListeners(){  
        this.NETwork.NETworkCanvas.MouseMove -= HandleMouseMove;  
    }  
    ...  
}
```

} Setting interaction mode

```
network.InteractionHandlers = new InteractionHandler[] {  
    new SelectInteractionHandler(network),  
    new MoveInteractionHandler(network, false),  
    new DefaultInteractionHandler(network),  
    new HighlightInputHandler(network) };
```



mouse over, node  
highlight



```
public class HighlightInputHandler : BasicInteractionHandler {
    public HighlightInputHandler(Network network) : base(network){ }
    override public void InstallListeners(){
        this.NETwork.NETworkCanvas.MouseMove += HandleMouseMove;
    }
    override public void UninstallListeners(){
        this.NETwork.NETworkCanvas.MouseMove -= HandleMouseMove;
    }
    private void HandleMouseMove(object sender, MouseEventArgs e){
        IElement element = network.GetElement(e);
        if (element != null){
            highlight(element);
        } else{
            reset();
        }
    }
    private IElement highlightElement;
    private Color? highLightColor = Utils.CreateColor(0xFFFF8888);
    private Color? oldColor = null;

    private void highlight(IElement element){
        if (element == null || element == highlightElement) {
            return;
        }
        reset();
        highlightElement = element;
        oldColor = (Color?)element.GetStyle(Styles.INNER_COLOR);
        network.Cursor = Cursors.Hand;
        element.SetStyle(Styles.INNER_COLOR, highLightColor);
    }
    private void reset(){
        if (highlightElement != null) {
            if (oldColor != null) {
                highlightElement.SetStyle(Styles.INNER_COLOR, oldColor);
            } else{
                highlightElement.SetStyle(Styles.INNER_COLOR, null);
            }
            highlightElement = null;
            network.Cursor = null;
        }
    }
}
```

# Network Filters

Filters are used for global control, such as control elements to show or hide, move or not move

Network#

//Visible filter

```
public Predicate<IElement> VisibleFunction
```

//Movable filter

```
public Predicate<IElement> MovableFunction
```

//Editable filter

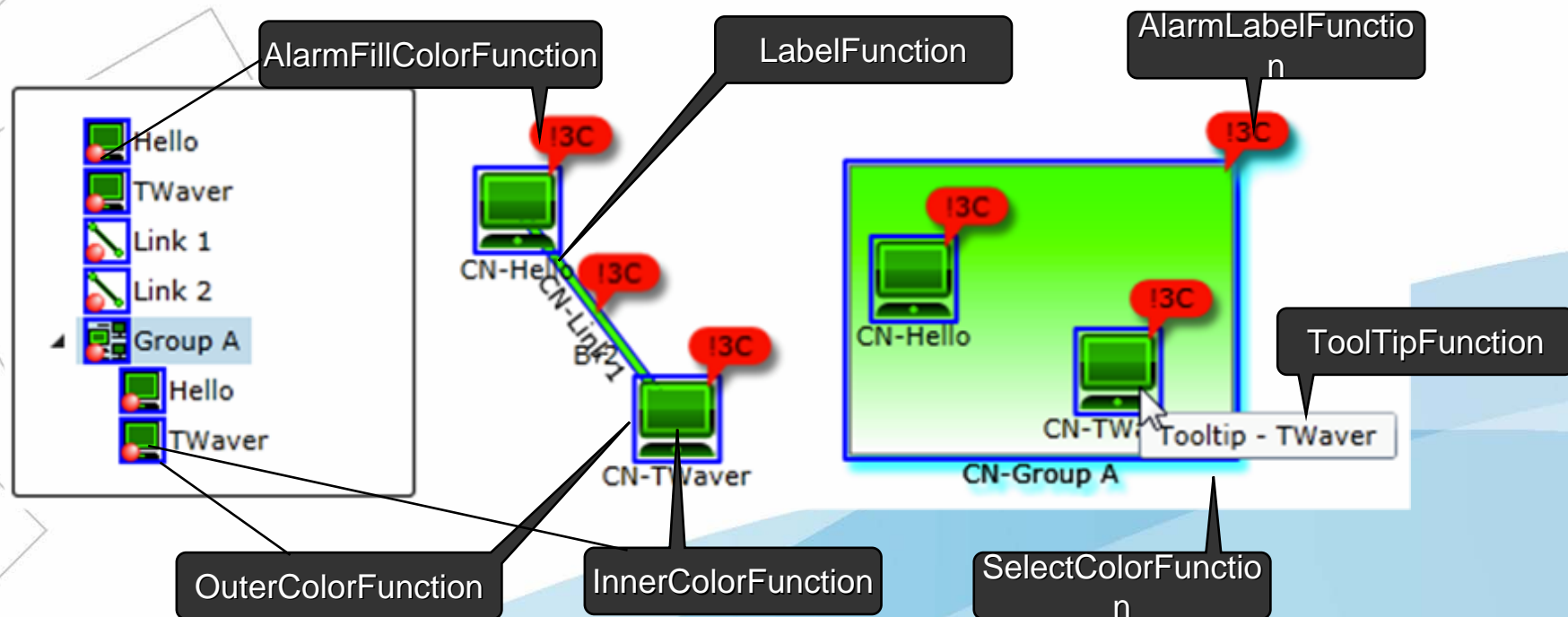
```
public Predicate<IElement> EditableFunction
```

```
network.VisibleFunction = (IElement element) =>{  
    return !(element is Link);  
};
```

# Network Style Functions

## \*\*\*Function

Style functions can have the general control of element styles

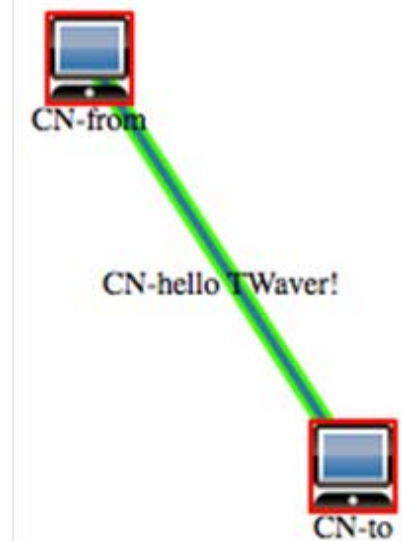


# Style Function Example

Example for adding prefix “CN-”  
to element label, and  
customizing outline color

```
network.LabelFunction = (IElement element) => {  
    return "CN-" + element.Name;  
};
```

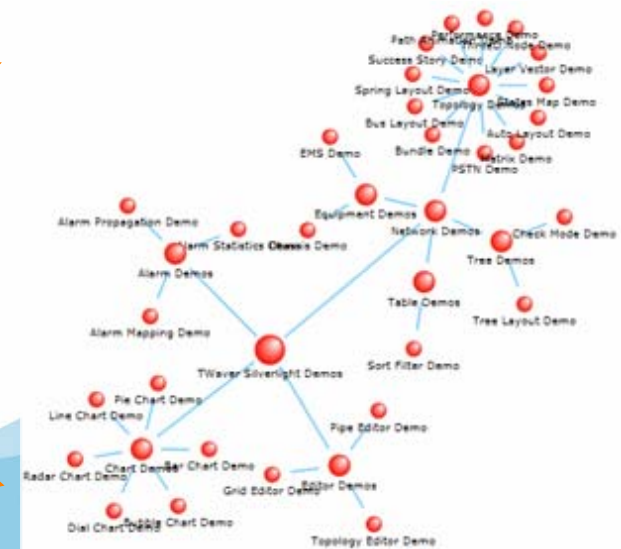
```
network.OuterColorFunction = (IData data) => {  
    return Color.FromArgb(255, 0, 0, 255);  
};
```



# Network Auto Layouts

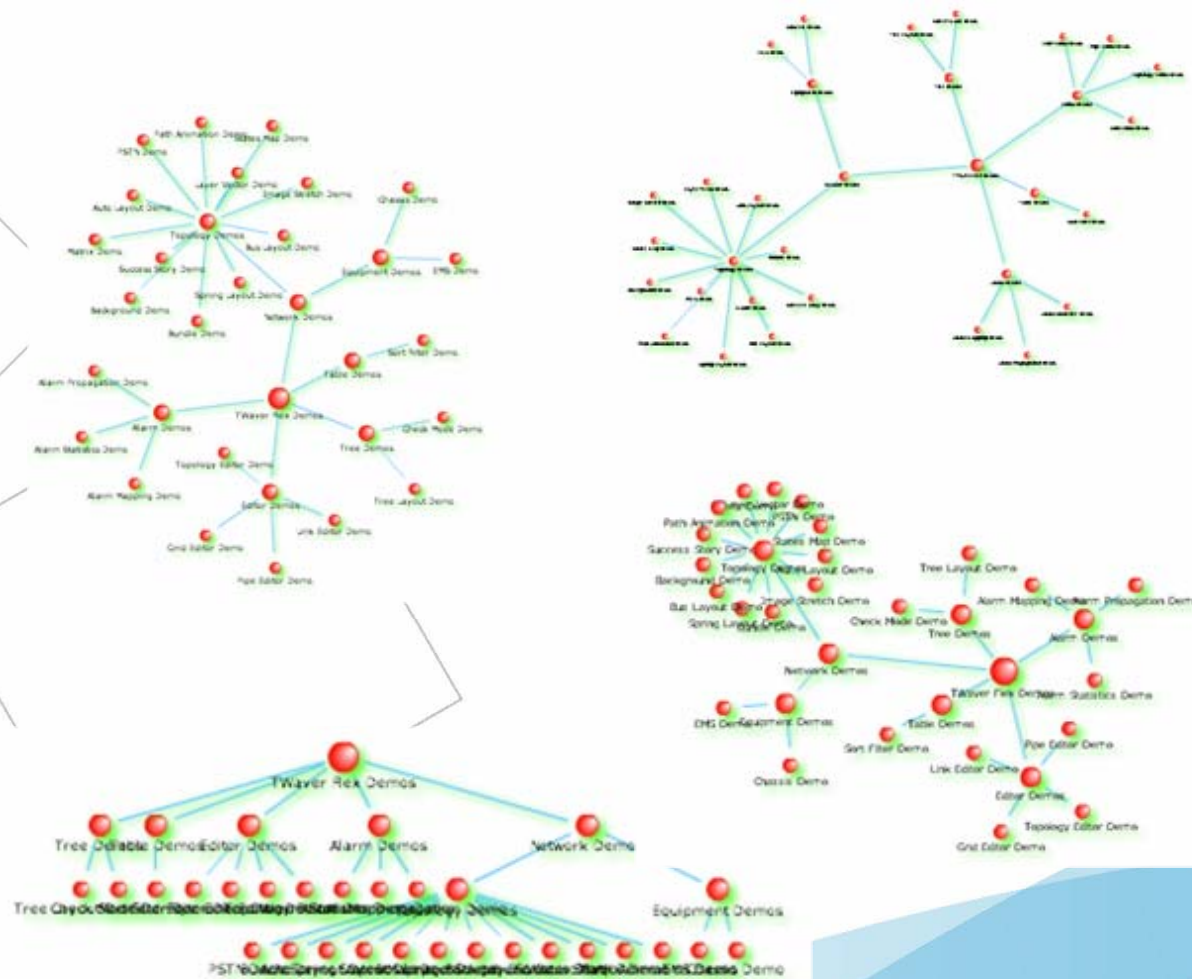
TWaver™

- Position nodes and links automatically, make them spread out in a good condition.





# Default Layouts





- Home - ServaSoftware.com
- Email - [tw-service@servasoft.com](mailto:tw-service@servasoft.com)

# Common Components & Alarms

TM  
waver

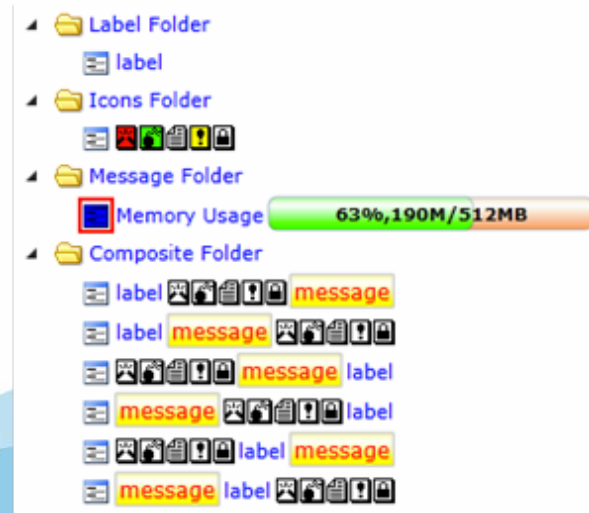
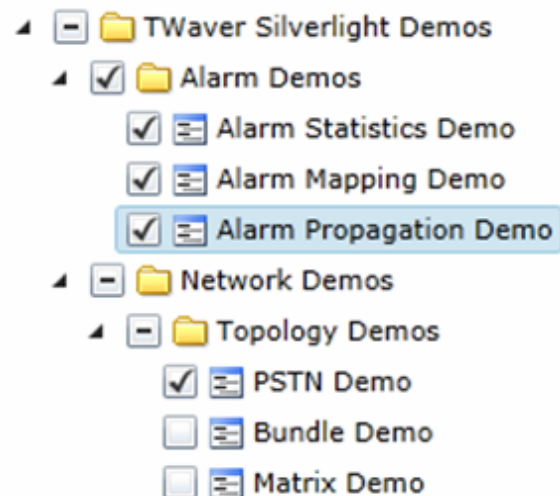
- **Common components**
  - Tree / Table / Chart
- **Alarm**
  - Alarm / AlarmSeverity / AlarmBox / AlarmState / AlarmStateStatistics / Alarm views

# Using Tree

TWaver™

twaver.controls.Tree - provides a hierarchical view of elements contained in a DataBox

```
Tree<IElement> tree = new Tree<IElement>(box);
```



# Creating a Tree

```
DataBox<Data> box = new DataBox<Data>();
```

```
Group group = new Group();  
box.Add(group);  
SubNetwork subnetwork = new SubNetwork();  
box.Add(subnetwork);
```

```
Node node = new Node();  
node.Name = "Node1";  
node.Parent = group;  
node.AlarmState.IncreaseNewAlarm(AlarmSeverity.CRITICAL);  
box.Add(node);  
node = new Node();  
node.Name = "Node2";  
node.Parent = group;  
box.Add(node);
```

```
node = new Node();  
node.Name = "Node3";  
node.Parent = subnetwork;  
box.Add(node);
```

```
tree = new TWaver.Controls.Tree<Data>();  
this.LayoutRoot.Children.Add(tree);  
tree.DataBox = box;  
tree.ExpandAll();
```



sets element's parent-child relationship

TWaver™



# Customizing Tree Node

- **Icon**

`data.Icon = "iconName";`

- **Label**

`data.Name = "001";`

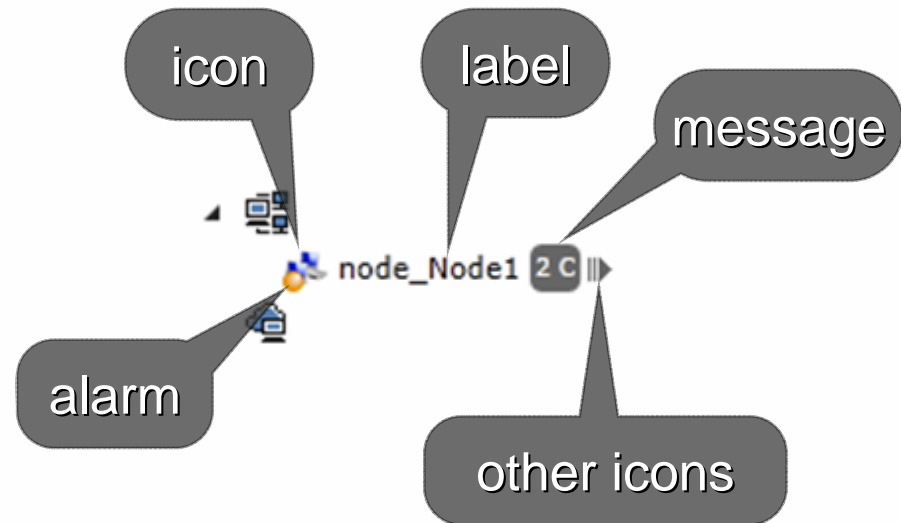
`element.SetStyle(Styles.TREE_LABEL, "002");`

- **Other styles**

`element.SetStyle(Styles.TREE_LABEL_***, ***);`

`element.SetStyle(Style.TREE_MESSAGE_***, ***);`

`element.SetStyle(Style.TREE_ICONS_***, ***);`

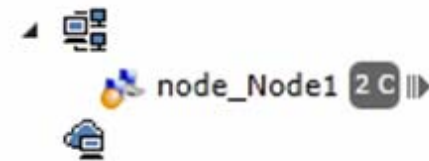


# Tree Node Style Example

TWaver™

```
ImageSource image = new BitmapImage(new  
Uri("/TWaverPPT;component/images/out.png", UriKind.Relative));  
Utils.RegisterImageByImageSource("outIcon", image, 9, 9);  
Utils.RegisterPNGImage("deviceIcon", new  
Uri("/TWaverPPT;component/images/device.png", UriKind.Relative));  
...
```

```
Node node = new Node();  
node.Name = "Node1";  
node.Icon = "deviceIcon";  
node.SetStyle(Styles.TREE_LABEL, "node_" + node.Name);  
node.SetStyle(Styles.TREE_ICONS_NAMES, new string[] { "outIcon" });  
node.SetStyle(Styles.TREE_LAYOUT_GAP, 3);  
node.SetStyle(Styles.TREE_MESSAGE, "2 C");  
node.SetStyle(Styles.TREE_MESSAGE_FILL_COLOR, Utils.CreateColor(0xFF666666));  
node.SetStyle(Styles.TREE_MESSAGE_GRADIENT, Consts.GRADIENT_NONE);  
node.SetStyle(Styles.TREE_MESSAGE_SIZE, 9);  
node.SetStyle(Styles.TREE_MESSAGE_COLOR, Utils.CreateColor(0xFFFFFFFF));  
node.SetStyle(Styles.TREE_LAYOUT,  
Consts.TREE_LAYOUT_LABEL_MESSAGE_ICONS);  
node.SetStyle(Styles.TREE_ALARM_FILL_COLOR, Utils.CreateColor(0xFFE08000));
```



# Node Hierarchy and Order

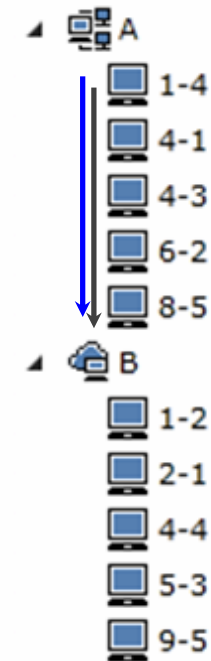
Twaver™

- Parent-child hierarchy  
`node.Parent = parent;`  
`node.AddChild(child);`
- Element order  
`box.Move***(node);`
- Sort by setting comparer  
`tree.CompareFunction = compareFunction;`

# Tree Sort Example

```
int i = 0;
while (i++ < 5)
{
    Node node = new Node();
    node.Name = "" + Utils.RandomInt(10) + "-" + i;
    node.Parent = group;
    box.Add(node);
}

tree.CompareFunction = (Data data1, Data data2) =>
{
    return Comparer<string>.Default.Compare(data1.Name, data2.Name);
};
```

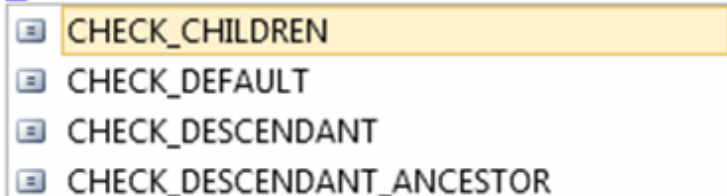


# Tree Check Mode

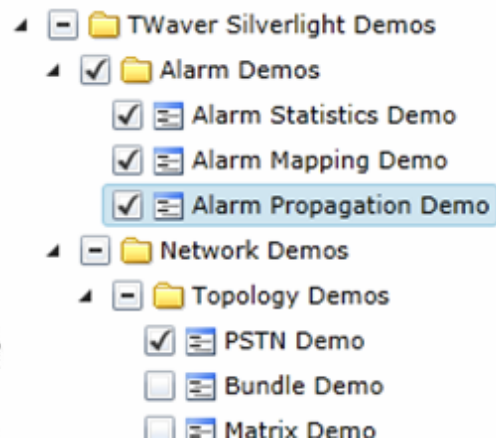
TWaver™

- Tree supplies check mode, setting as follow:
- *tree.CheckMode = Tree<Data>.CHECK\_\*\*\**

```
tree.CheckMode = Tree<Data>.check_
```



- ☒ CHECK\_CHILDREN
- ☐ CHECK\_DEFAULT
- ☐ CHECK\_DESCENDANT
- ☐ CHECK\_DESCENDANT\_ANCESTOR



Tree has four types of check modes:  
CheckModeDemo



# Use Table

Twaver™

- Bind with DataBox, list all data that in the DataBox, each row is one data
- Support the display for three types of data properties: accessor、style、client
- Data property change and the table update automatically
- Table inherited from System.Windows.Controls.DataGrid.

# Create a Table

TWaver™

TWaver.Controls.Table<> - List all data that in the DataBox, each row is one data

```
Table<Data> table = new Table<Data>(box);
```

Name	Is Single	Color
1	<input type="checkbox"/>	#FF8EC80E
2	<input checked="" type="checkbox"/>	#FF9BF5D4
3	<input checked="" type="checkbox"/>	#FFDE7CD3
4	<input type="checkbox"/>	#FF9FD331
5	<input checked="" type="checkbox"/>	#FF5875B6

Mapping ID	Severity	Acked	Cleared
1	Indeterminate	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2	Warning	<input type="checkbox"/>	<input type="checkbox"/>
3	Minor	<input type="checkbox"/>	<input type="checkbox"/>
4	Major	<input type="checkbox"/>	<input type="checkbox"/>
5	Critical	<input type="checkbox"/>	<input type="checkbox"/>

# Create Table

```
DataBox<Data> box = new DataBox<Data>();
```

```
int i = 0;
while (i++ < 10)
{
    Data data = new Data();
    data.Name = "" + i;
    data.SetClient("age", Utils.RandomInt(30));
    data.SetClient("isSingle", Utils.RandomBool());
    data.SetClient("color", Utils.RandomColor());
    box.Add(data);
}
```

```
Table<Data> table = new Table<Data>(box);
```

```
DataGridColumn column = new TableTextColumn(Consts.PROPERTY_TYPE_ACCESSOR,
"Name", "Name");
table.Columns.Add(column);
column = new TableCheckBoxColumn(Consts.PROPERTY_TYPE_CLIENT, "isSingle", "Is
Single");
table.Columns.Add(column);
column = new TableColorColumn(Consts.PROPERTY_TYPE_CLIENT, "color", "Color");
table.Columns.Add(column);
```

Name	Is Single	Color
1	<input type="checkbox"/>	#FF8EC80E
2	<input checked="" type="checkbox"/>	#FF9BF5D4
3	<input checked="" type="checkbox"/>	#FFDE7CD3
4	<input type="checkbox"/>	#FF9FD331
5	<input checked="" type="checkbox"/>	#FF5875B6

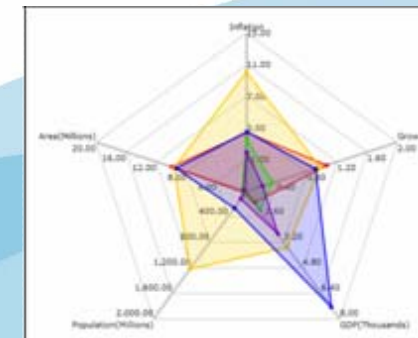
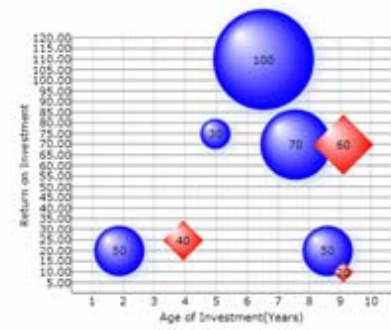
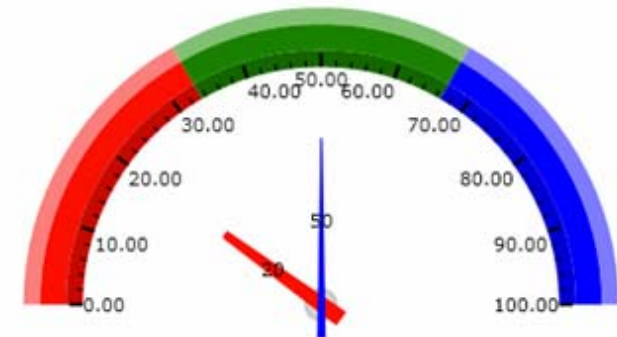
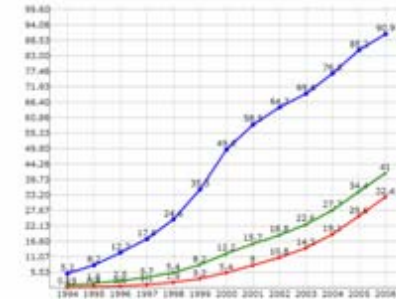
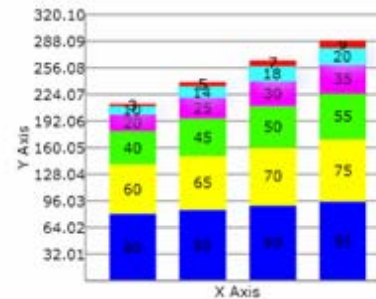
# Table Rows Order

- Table has three types of orders - reverse, forward and roots
- `table.IterateMode = Consts. ITERATE_MODE_***;`
  - `Consts.ITERATE_MODE_REVERSE`
  - `Consts.ITERATE_MODE_FORWARD`
  - `Consts.ITERATE_MODE_ROOTS`
- Adjust row order by method of `dataBox.Move***(element)`

# Use Charts

TM  
Twaiver

- BarChart
- LineChart
- PieChart
- DialChart
- BubbleChart
- RadarChart



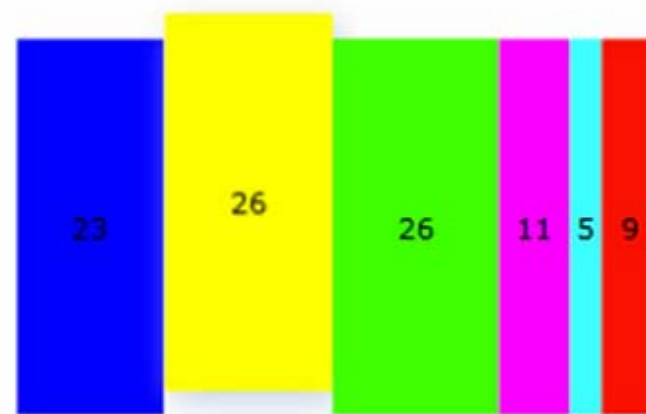
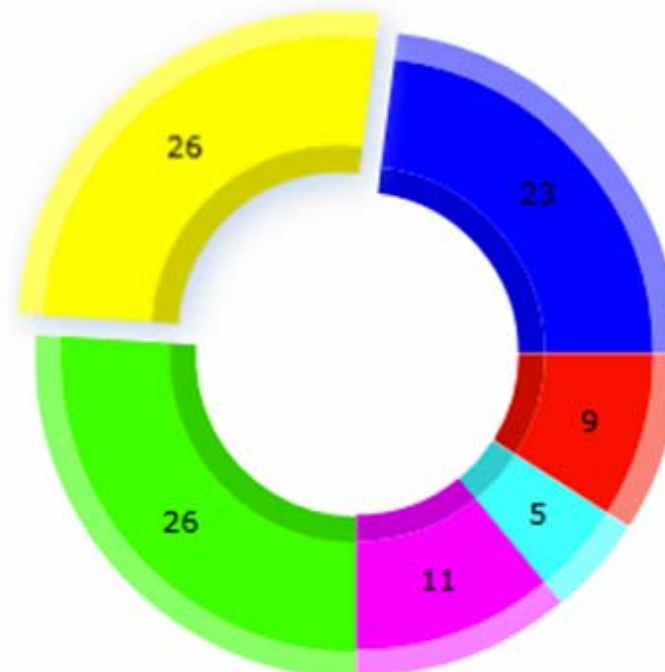
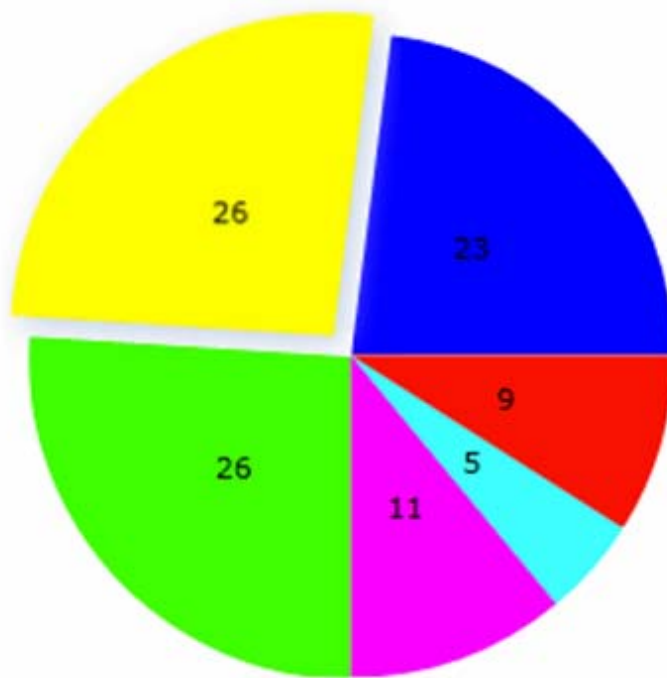


# ChartValue & ChartValues

TWaver™

- Chart associated with the DataBox, graphically display the chart values of elements in the DataBox
- There are two types of chart values:  
***CHART\_VALUE & CHART\_VALUES***
- setting chart values as follows:
- *element.SetStyle(Styles.CHART\_VALUE, 27d);*
- *element.SetStyle(Styles.CHART\_VALUES, new double[]{27, 37, 48});*

# PieChart



Twaver™

# PieChart

```
public PieChartTest() {
    InitializeComponent();

    DataBox<Data> box = new DataBox<Data>();
    PieChart<Data> chart = new PieChart<Data>(box);

    createData(box, "A", 10, Utils.CreateColor(0xFF4CB743))
    createData(box, "B", 20, Utils.CreateColor(0xFFD6EAC9))
    createData(box, "C", 40, Utils.CreateColor(0xFF77DD77))

    chart.Type = Consts.PIECHART_TYPE_OVALDONUT;
    chart.ValueTextFunction = (IData item, double value) =>
    {
        return item.Name + " - " + (int)(value / chart.Sum * 100) + "%";
    };

    this.LayoutRoot.Children.Add(chart);
}

private void createData(DataBox<Data> box, string name, double value, Color color){
    Data data = new Data();
    data.SetClient(Styles.CHART_VALUE, value);
    data.SetClient(Styles.CHART_COLOR, color);
    data.SetClient(Styles.CHART_VALUE_COLOR, Utils.CreateColor(0xFF555555));
    data.SetClient(Styles.CHART_VALUE_FONT, new FontFamily("heivetica"));
    data.Name = name;
    box.Add(data);
}
```



TWaver™

# LineChart

```

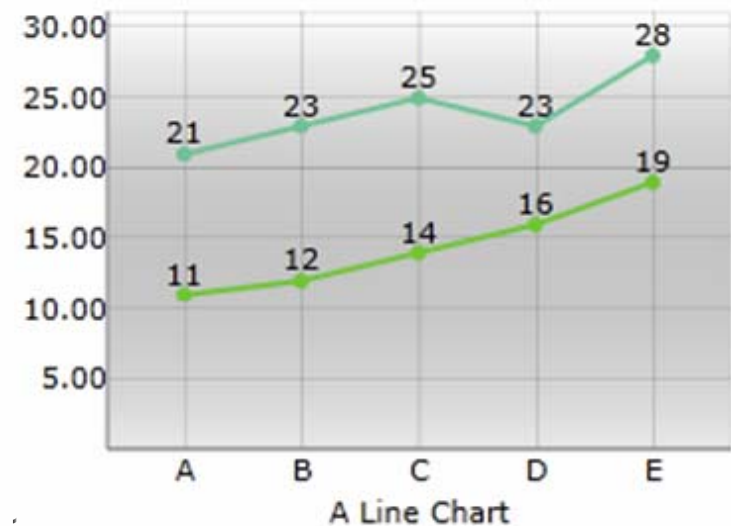
public LineChartTest()
{
    InitializeComponent();

    DataBox<Data> box = new DataBox<Data>();
    LineChart<Data> chart = new LineChart<Data>(box);

    Element data = new Element();
    data.SetStyle(Styles.CHART_VALUES, new double[] {
19d });
    box.Add(data);
    data = new Element();
    data.SetStyle(Styles.CHART_VALUES, new double[] { 21d, 23d, 25d, 23d,
28d });
    box.Add(data);

    chart.XScaleTexts = new string[] { "A", "B", "C", "D", "E" };
    chart.LowerLimit = 0d;
    chart.YScaleValueGap = 5d;
    chart.XAxisText = "A Line Chart";
    chart.BackgroundVisible = true;
    this.LayoutRoot.Children.Add(chart);
}

```

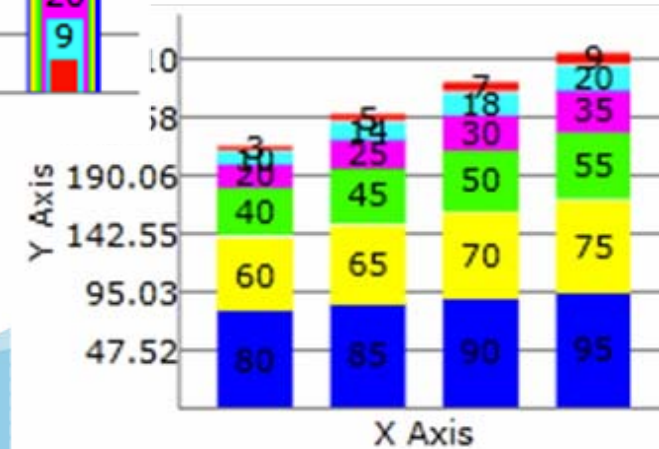
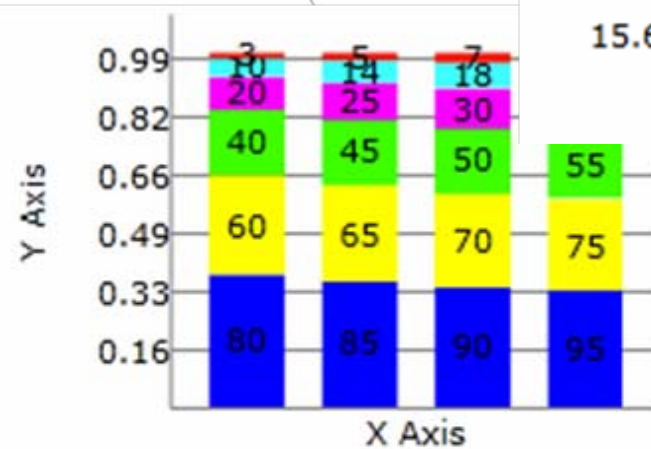
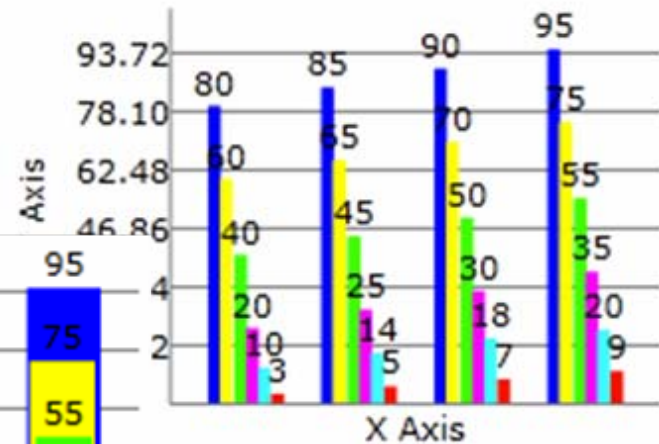
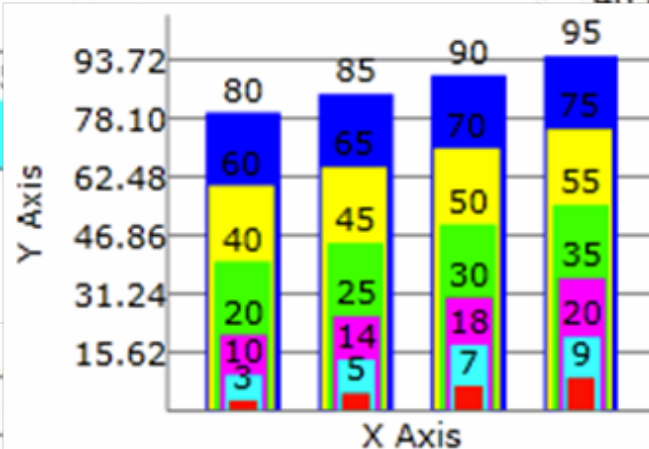
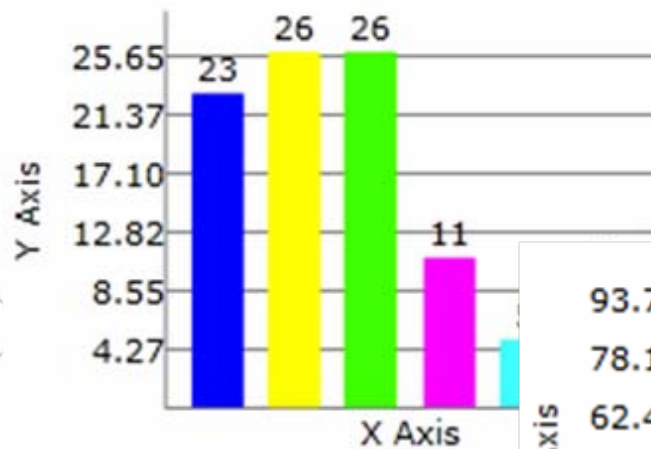


TWaver™



# BarChart

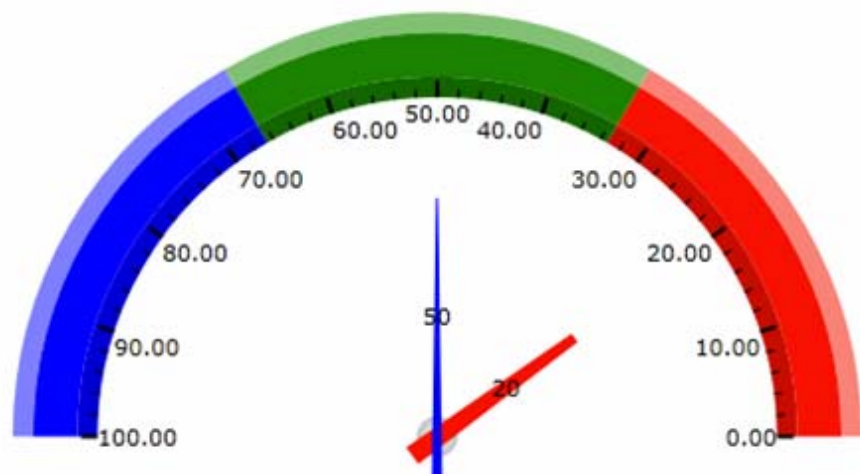
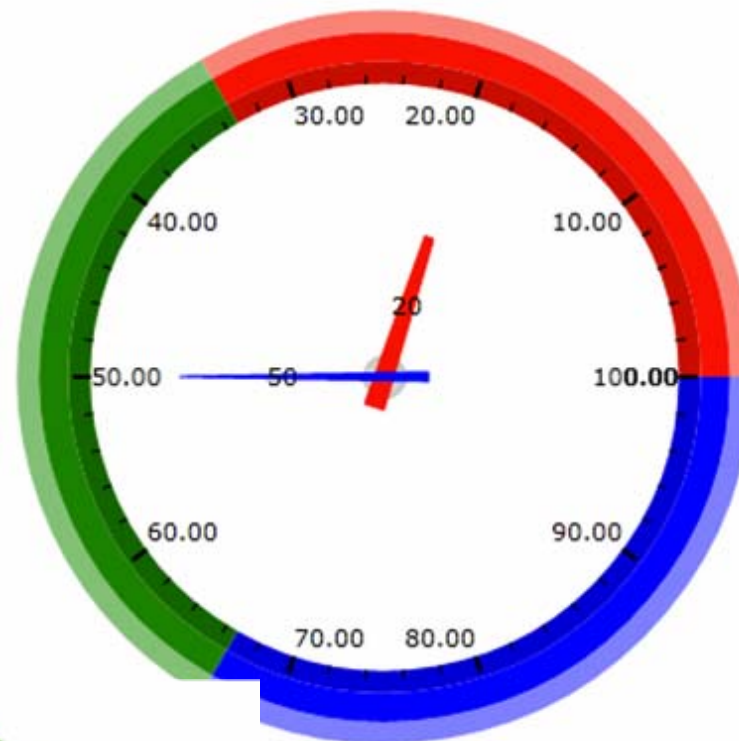
Twaver™





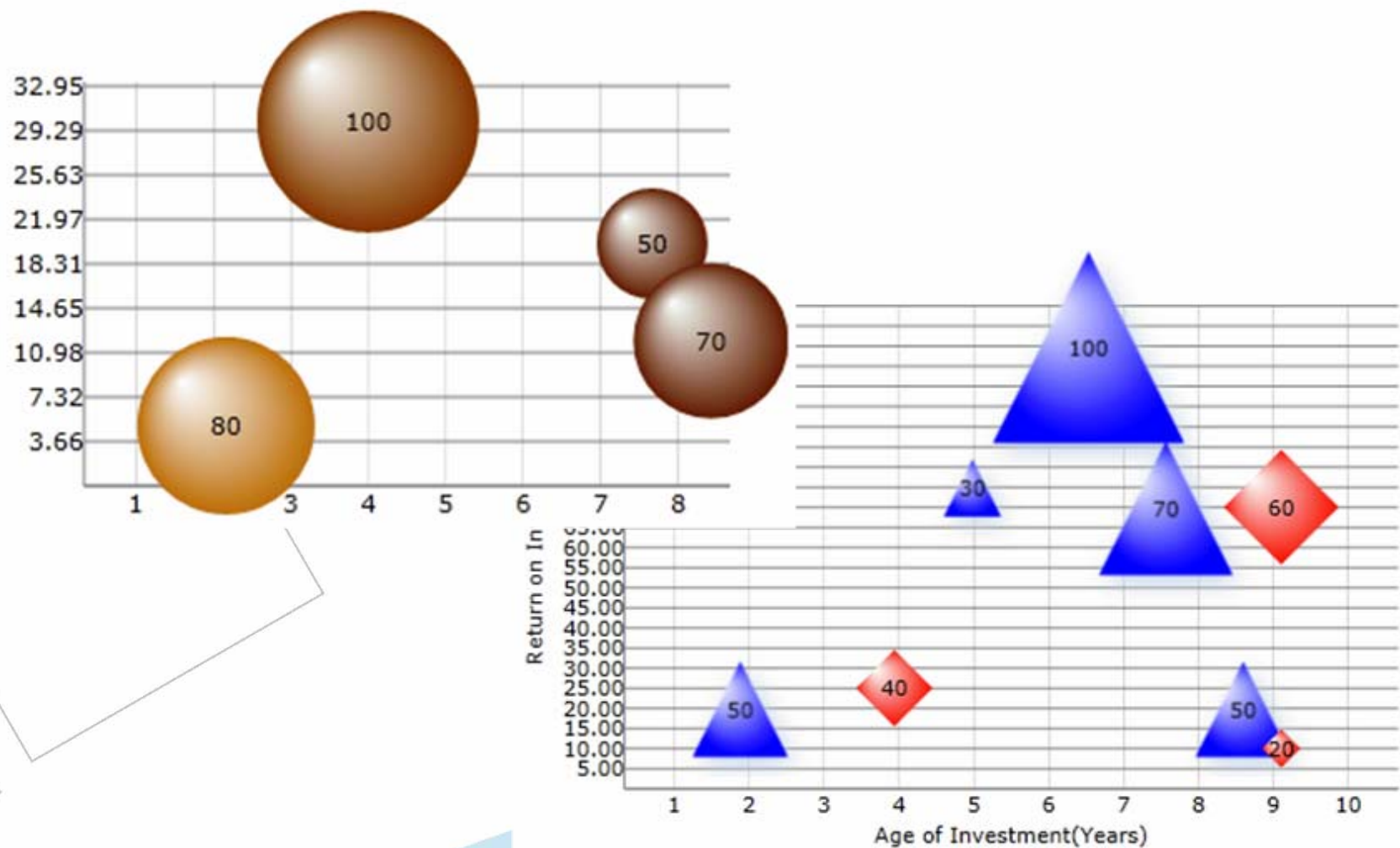
# DialChart

Twaver™



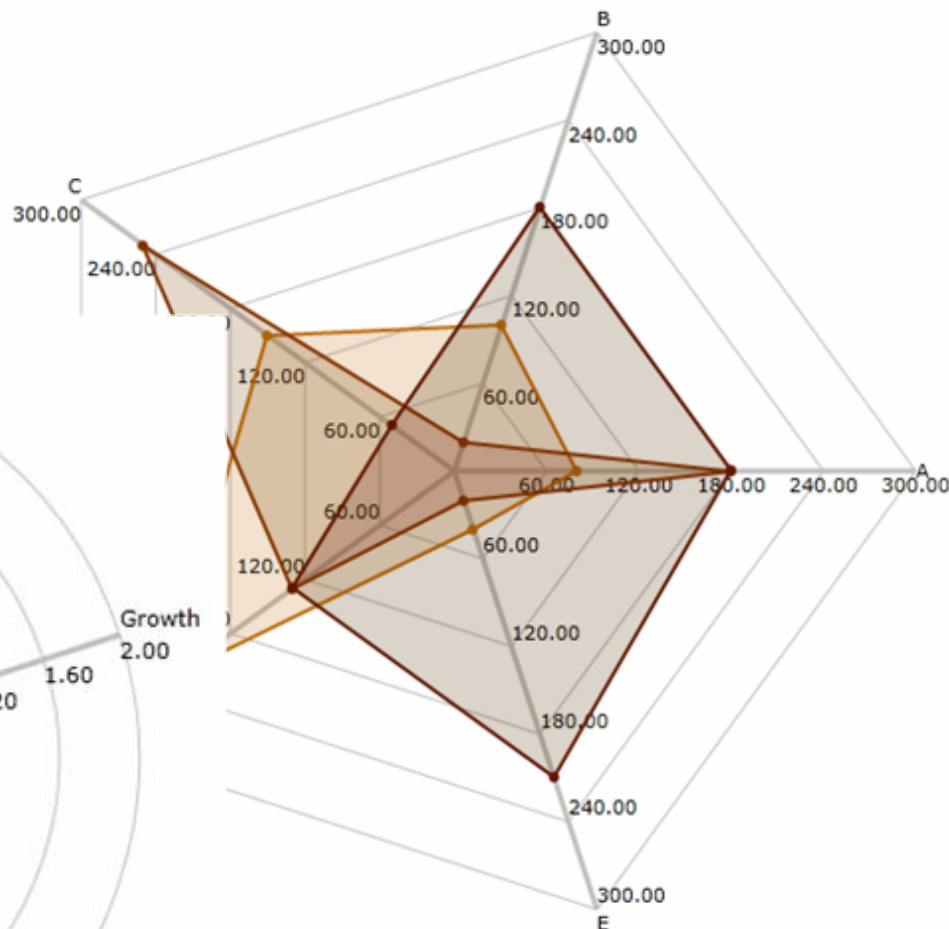
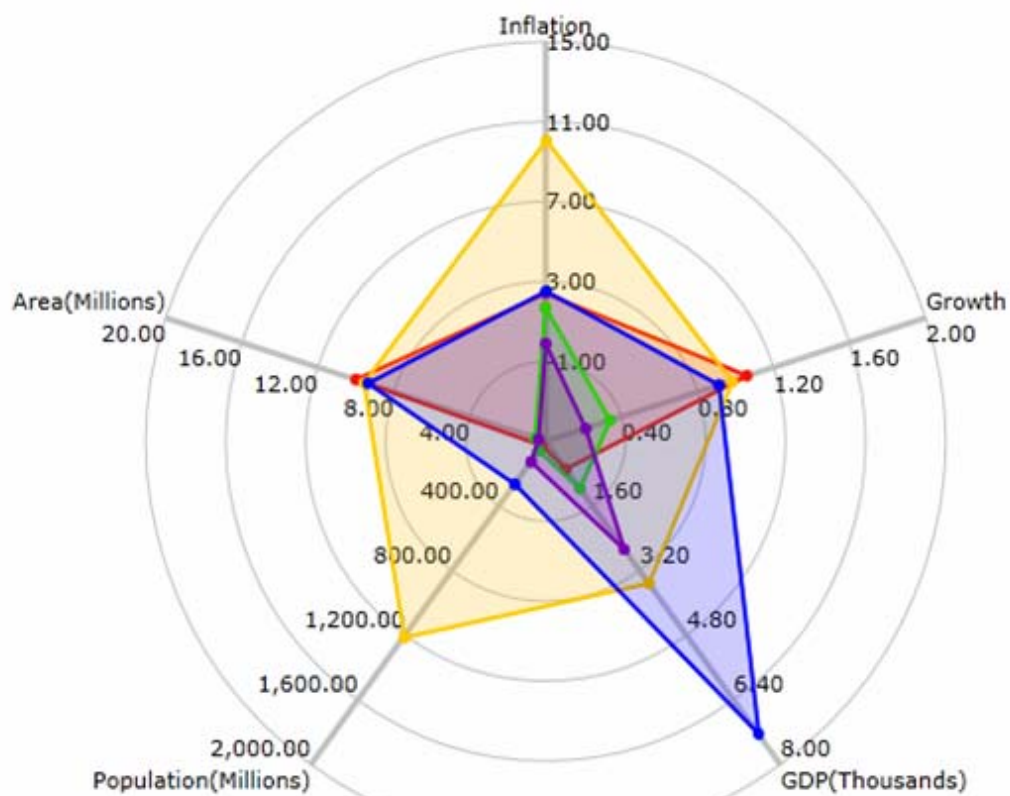
# BubbleChart

TWaver™



# RadarChart

Twaver™



# Using Alarm

Twaver™

- Alarm - describes the element or device operation status
- Alarm/AlarmSeverity
- AlarmBox
- AlarmState/AlarmStateStatistics
- Alarm display



# AlarmSeverity

TWaver™

- *TWaver. AlarmSeverity*
- **Alarm severity** - Reflect the urgency of the alarm contains name, nickName, value, color and other properties
- **Default alarm severities** - There are six alarm severities predefined.

*CRITICAL, MAJOR, MINOR, WARNING, INDETERMINATE, CLEARED*

- **Custom alarm severities**

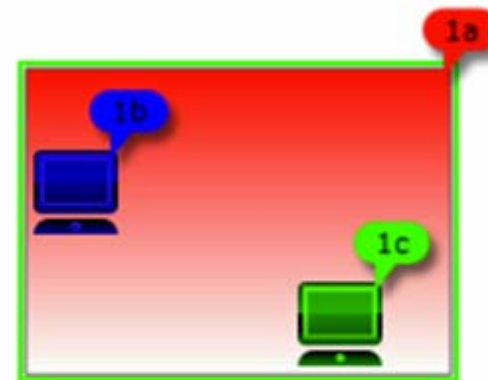
```
AlarmSeverity.Clear();
```

```
AlarmSeverity.Register(1, "a", "a", Utils.CreateColor(0xFFFF0000), "AAA");
```



# Custom Alarm Severity Example

```
public CustomAlarmSeverityTest() {  
    ...  
    AlarmSeverity.Clear();  
    AlarmSeverity a = AlarmSeverity.Register(1, "a", "a", Utils.CreateColor(0xFFFF0000), "AAA");  
    AlarmSeverity b = AlarmSeverity.Register(2, "b", "b", Utils.CreateColor(0xFF0000FF), "BBB");  
    AlarmSeverity c = AlarmSeverity.Register(3, "c", "c", Utils.CreateColor(0xFF00FF00), "CCC");  
  
    ElementBox box = new ElementBox();  
  
    Group node1 = new Group();  
    node1.IsExpanded = true;  
    Node node2 = new Node();  
    node2.SetLocation(100, 100);  
    Node node3 = new Node();  
    node3.SetLocation(200, 150);  
    node3.Parent = node1;  
    node2.Parent = node1;  
    box.Add(node1);  
    box.Add(node2);  
    box.Add(node3);  
  
    AddAlarm(box.AlarmBox, node1, a);  
    AddAlarm(box.AlarmBox, node2, b);  
    AddAlarm(box.AlarmBox, node3, c);  
    ...  
}  
  
private void AddAlarm(AlarmBox alarmBox, Element element, AlarmSeverity severity) {  
    Alarm alarm = new Alarm(null, element.ID, severity);  
    alarmBox.Add(alarm);  
}
```



# Alarm Object

TWaver™

- TWaver.Alarm - the basic data type of alarm.
- Alarm properties:  
ID, ElementID, AlarmSeverity, IsAcked, IsCleared  
and other properties by method:  
alarm.SetClient(key, value).

# AlarmBox

Twaver™

- AlarmBox - Data container for managing alarm objects.
- Basic Operations  
Add/Remove/Clear ...
- Quick finder and listeners



AlarmBox

# AlarmState

TWaver™

- TWaver. AlarmState - describes the current alarm status of elements or devices
- Including all statistical information about new alarms, acknowledged alarms and propagated alarms.
- The specific alarm objects(TWaver.Alarm) are not included

# AlarmStateStatistics

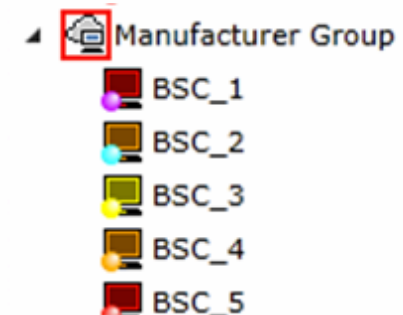
Twaver™

- AlarmStateStatistics - Counts the alarm states of all elements in the ElementBox.
- Includes all quantities of alarms for each alarm severity
- Includes the numbers of all new alarms or acknowledged alarms.
- Supports filter, can statistics for specifying elements



# Alarm Display

- Alarm ballon
- Alarm color - fill or outline
- Alarm table
- Alarm statistics charts



Mapping ID	Severity	Acked	Cleared	Raised Time
1	Indeterminate	<input type="checkbox"/>	<input type="checkbox"/>	2011-08-30
2	Warning	<input type="checkbox"/>	<input type="checkbox"/>	2011-08-30
3	Minor	<input type="checkbox"/>	<input type="checkbox"/>	2011-08-30
4	Major	<input type="checkbox"/>	<input type="checkbox"/>	2011-08-30
5	Critical	<input type="checkbox"/>	<input type="checkbox"/>	2011-08-30

Name	New	Acked	Total
Critical	3	2	5
Major	5	1	6
Minor	1	3	4
Warning	8	1	9
Indeterminate	0	7	7
Cleared	2	1	3
Total	19	15	34

# Thank you

- Home - ServaSoftware.com
- Email - [tw-service@servasoft.com](mailto:tw-service@servasoft.com)