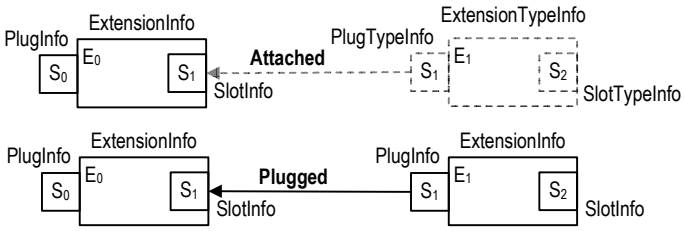
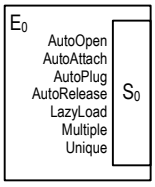
Legend

Attach & Plug

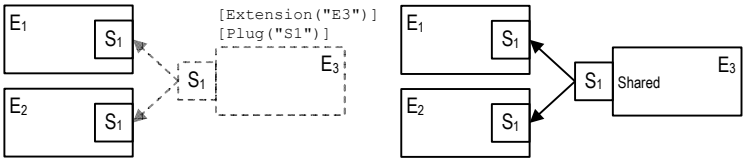


Slot Properties



Shared extension

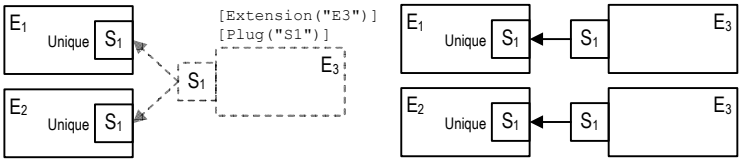
```
[Extension("E1")]
[Slot("S1")]
```




```
[Extension("E2")]
[Slot("S1")]
```

Unique extension (single plug)

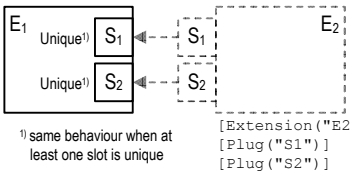
```
[Extension("E1")]
[Slot("S1", UniqueExtension=true)]
```




```
[Extension("E2")]
[Slot("S1", UniqueExtension=true)]
```

Unique extension (multiple plugs)

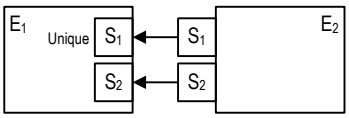
```
[Extension("E1")]
[Slot("S1", UniqueExtension=true)]
[Slot("S2")]
```



1) same behaviour when at least one slot is unique

```
[Extension("E2")]
[Plug("S1")]
[Plug("S2")]
```

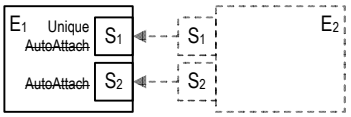
a) single unique extension (default)



Plux.NET Runtime calls...

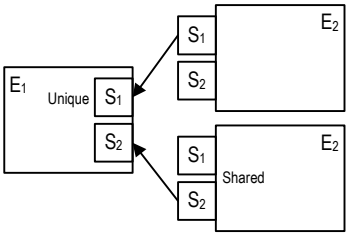
```
Runtime.Repository
  .ExtensionInfos["E1"]
  .AttachInSlots()
```

```
[Extension("E1", OnCreated="E1_Created")]
[Slot("S1",
  UniqueExtension=true,
  AutoAttach=false)]
[Slot("S2", AutoAttach=false)]
```



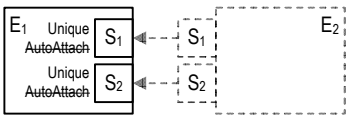

```
[Extension("E2")]
[Plug("S1")]
[Plug("S2")]
```

b) unique and shared extension (manual attach)



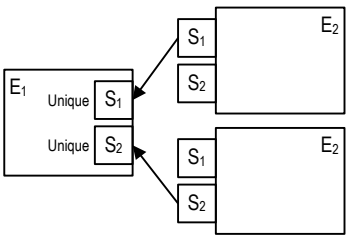

```
void E1_Created(object s, ExtensionEventArgs eea) {
  eea.ExtensionInfo.SlotInfos["S1"].Attach();
  eea.ExtensionInfo.SlotInfos["S2"].Attach();
}
```

```
[Extension("E1", OnCreated="E1_Created")]
[Slot("S1", UniqueExtension=true,
  AutoAttach=false)]
[Slot("S2", UniqueExtension=true,
  AutoAttach=false)]
```



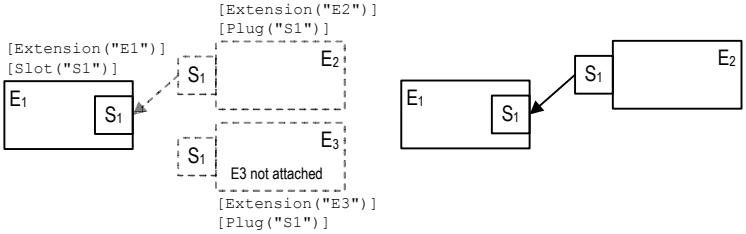

```
[Extension("E2")]
[Plug("S1")]
[Plug("S2")]
```

c) two unique extensions (manual attach)

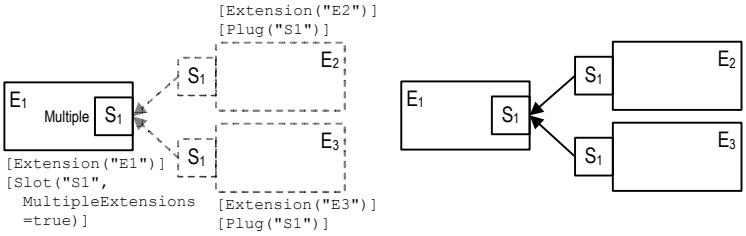



```
void E1_Created(object s, ExtensionEventArgs eea) {
  eea.ExtensionInfo.SlotInfos["S1"].Attach();
  eea.ExtensionInfo.SlotInfos["S2"].Attach();
}
```

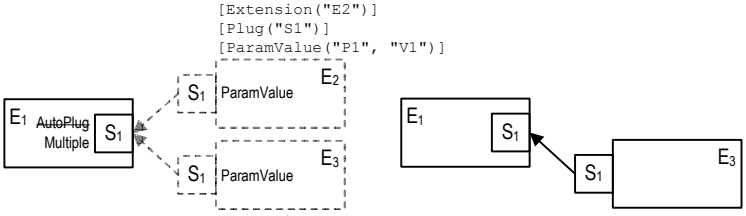
Single contributor



Multiple contributors

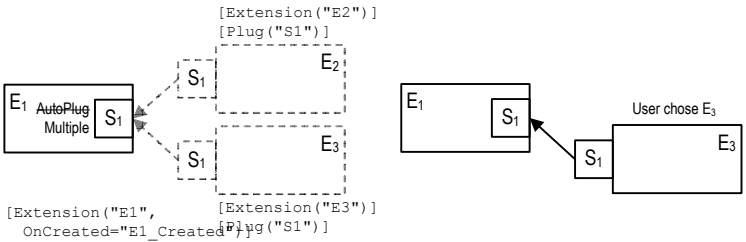


Choose contributor depending on parameter (manual plugging)




```
[Extension("E1")]
[Slot("S1",
  AutoPlug=false,
  MultipleExtensions=true,
  OnAttach="S1_Attached")]

void S1_Attached(object s, AttachEventArgs aea) {
  if((string) ara.GetParamValue("P1") == "V2")
    aea.PlugTypeInfo.ExtensionTypeInfo.
      GetSharedExtension(true).PlugInSlots();
}
```

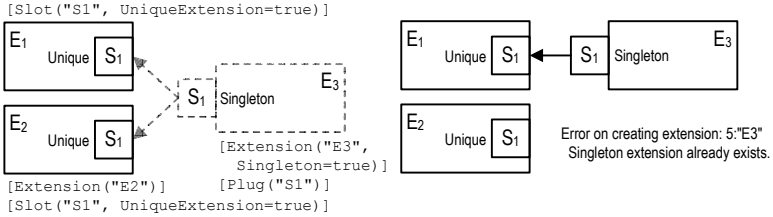
Choose contributors w/ user interaction (manual plugging)




```
[Extension("E1",
  OnCreated="E1_Created")]
[Slot("S1",
  AutoPlug=false,
  MultipleExtensions=true)]

ExtensionInfo me = null;
void E1_Created(object s, ExtensionEventArgs eea) {
  me = eea.ExtensionInfo;
  Runtime.TaskQueue.QueueEmptied += ViewAvailableExtensions;
}
void ViewAvailableExtensions() {
  Runtime.TaskQueue.QueueEmptied -= ViewAvailableExtensions;
  foreach(PlugTypeInfo pti in me.SlotInfos[0].AttachedPlugTypeInfos)
    Console.WriteLine(pti.ExtensionTypeInfo.Name);
}
void Choose(ExtensionTypeInfo eti) {
  me.SlotInfos[0].Plug(eti.GetSharedExtension(true).PlugInfos[0]);
}
```

Singleton extension (2nd unique fails)

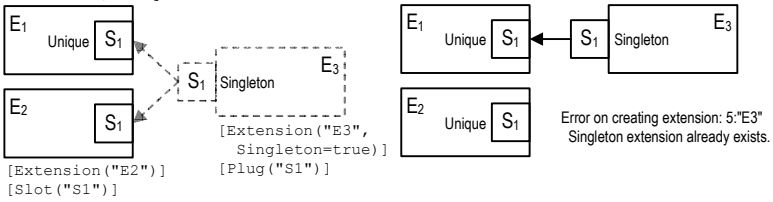
```
[Extension("E1")]
[Slot("S1", UniqueExtension=true)]
```



Patterns\Singleton1.cs

Singleton extension (unique first, shared fails)

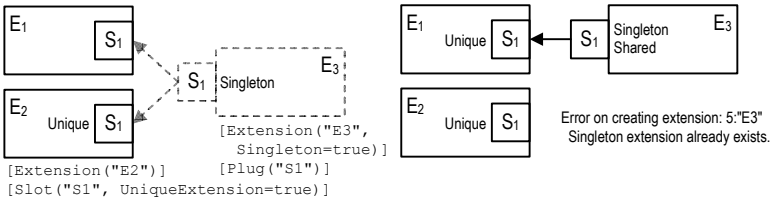
```
[Extension("E1")]
[Slot("S1", UniqueExtension=true)]
```



Patterns\Singleton2.cs

Singleton extension (shared first, unique fails)

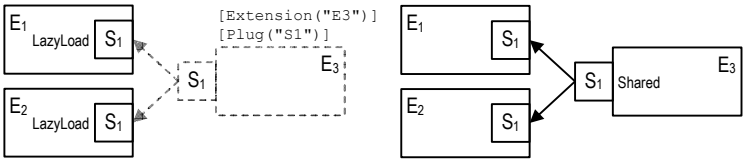
```
[Extension("E1")]
[Slot("S1")]
```



Patterns\Singleton3.cs
w/ manual plugging Patterns\Singleton4.cs

Lazy loading (shared extension)

```
[Extension("E1")]
[Slot("S1", LazyLoad=true)]
```



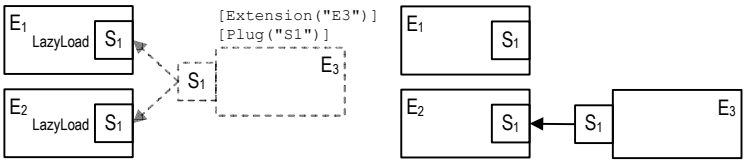

```
[Extension("E2")]
[Slot("S1", LazyLoad=true, OnAttached="S1_Attach")]

public void S1_Attach(object s, AttachEventArgs aea) {
  aea.PlugTypeInfo.ExtensionTypeInfo.CreateSharedInstance();
}
```

Shared extensions are plugged automatically when created.

Patterns\LazyLoading1.cs

Lazy loading (unique extension)

```
[Extension("E1")]
[Slot("S1", LazyLoad=true)]
```



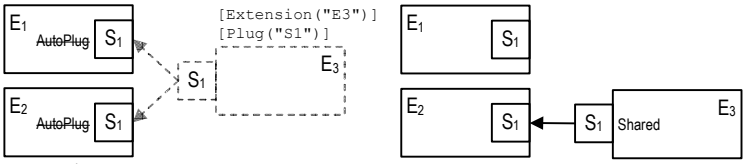

```
[Extension("E2")]
[Slot("S1", LazyLoad=true, OnAttached="S1_Attach")]

public void S1_Attached(object s, AttachEventArgs aea) {
  aea.SlotInfo.ExtensionInfo.PlugInSlots(
    aea.PlugTypeInfo.ExtensionTypeInfo.CreateUniqueExtension());
}
```

Unique extensions are plugged manually after creation.

Patterns\LazyLoading2.cs

Manual plugging

```
[Extension("E1")]
[Slot("S1", AutoPlug=false)]
```



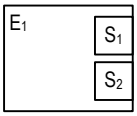

```
[Extension("E2")]
[Slot("S1", AutoPlug=false, OnAttached="S1_Attach")]

public void S1_Attach(object s, AttachEventArgs aea) {
  aea.SlotInfo.ExtensionInfo.PlugInSlots(
    aea.PlugTypeInfo.ExtensionTypeInfo.GetSharedExtension(true));
}
```

Shared extensions are plugged automatically when created.

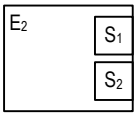
Patterns\ManualPlugging1.cs

Manual Attaching

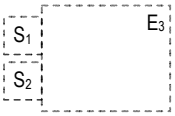
```
[Extension("E1")]
[Slot("S1",
  AutoAttach=false,
  Multiple=true)]
[Slot("S2",
  AutoAttach=false,
  Multiple=true)]
```



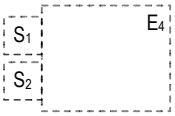

```
[Extension("E2")]
[Slot("S1",
  AutoAttach=false,
  Multiple=true)]
[Slot("S2",
  AutoAttach=false,
  Multiple=true)]
```



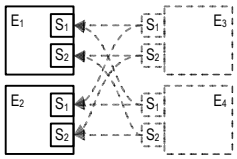

```
[Extension("E3")]
[Plug("S1")]
[Plug("S2")]
```




```
[Extension("E4")]
[Plug("S1")]
[Plug("S2")]
```



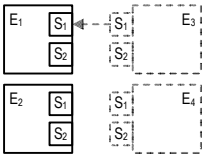
with AutoAttach=true




```
ExtensionInfo e1 = Runtime.Repository.ExtensionInfo["E1"]
ExtensionInfo e2 = Runtime.Repository.ExtensionInfo["E2"]
ExtensionTypeInfo t3 = Runtime.Repository.ExtensionTypeInfos["E3"];
ExtensionTypeInfo t4 = Runtime.Repository.ExtensionTypeInfos["E4"];
```

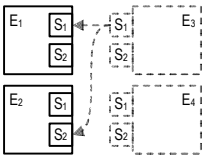
Attach single plug into single slot

```
e1.SlotInfos["S1"].Attach(t3.PlugTypeInfos["S1"]);
   - alternatively -
t3.PlugTypeInfos["S1"].Attach(e1.SlotInfos["S1"]);
```



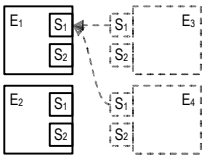
Attach single plug

```
t3.PlugTypeInfos["S1"].Attach();
```



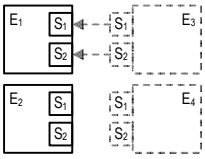
Attach to single slot

```
e1.SlotInfos["S1"].Attach();
```



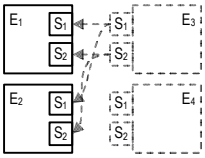
Attach extension to extension

```
t3.AttachPlugs(e3);
   - alternatively -
e3.AttachInSlots(t3);
```



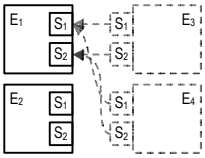
Attach extension

```
t3.AttachPlugs();
```



Attach to extension

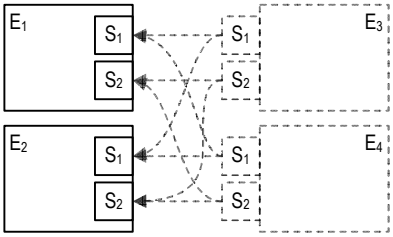
```
e1.AttachInSlots();
```



Manual plugging

```
[Extension("E1")]
[Slot("S1",
  AutoPlug=false,
  Multiple=true)]
[Slot("S2",
  AutoPlug=false,
  Multiple=true)]
```

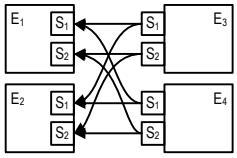
```
[Extension("E3")]
[Plug("S1")]
[Plug("S2")]
```




```
[Extension("E2")]
[Slot("S1",
  AutoPlug=false,
  Multiple=true)]
[Slot("S2",
  AutoPlug=false,
  Multiple=true)]
```

```
[Extension("E4")]
[Plug("S1")]
[Plug("S2")]
```

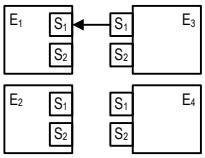
with AutoPlug=true




```
ExtensionInfo e1 = Runtime.Repository.ExtensionInfo["E1"]
ExtensionInfo e2 = Runtime.Repository.ExtensionInfo["E2"]
ExtensionTypeInfo t3 = Runtime.Repository.ExtensionTypeInfos["E3"];
ExtensionTypeInfo t4 = Runtime.Repository.ExtensionTypeInfos["E4"];
ExtensionInfo e3 = t3.GetSharedExtension(true);
ExtensionInfo e4 = t4.GetSharedExtension(true);
```

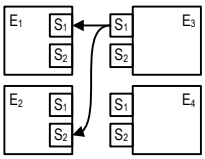
Plug single plug into single slot

```
e1.SlotInfos["S1"].Plug(e3.PlugInfos["S1"]);
   - alternatively -
e3.PlugInfos["S1"].Plug(e1.SlotInfos["S1"]);
```



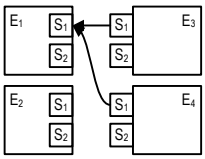
Plug single plug

```
e3.PlugInfos["S1"].Plug();
```



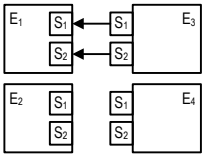
Plug into single slot

```
e1.SlotInfos["S1"].Plug();
```



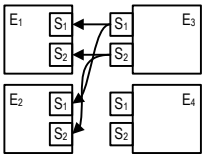
Plug extension into extension

```
e1.PlugPlugs(e3);
   - alternatively -
e3.PlugInSlots(e1);
```



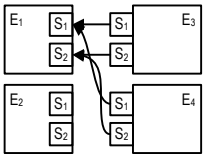
Plug extension

```
e3.PlugPlugs();
```

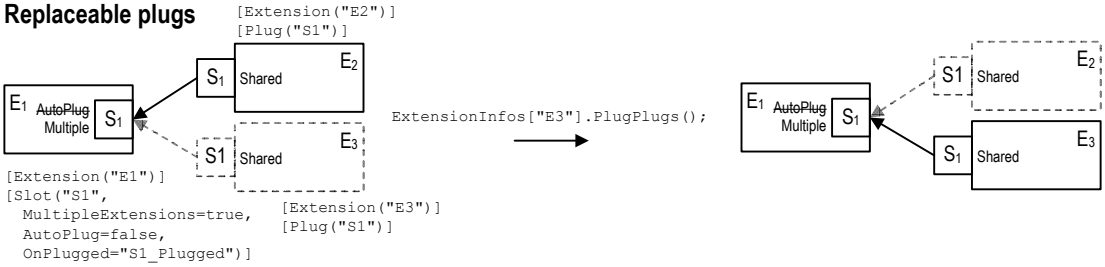


Plug into extension

```
e1.PlugInSlots();
```



Replaceable plugs




```
[Extension("E1")]
[Slot("S1",
  MultipleExtensions=true,
  AutoPlug=false,
  OnPlugged="S1_Plugged")]
```

```
[Extension("E3")]
[Plug("S1")]
```

```
public void S1_Plugged(object s, PlugEventArgs args) {
  foreach(PlugInfo pi in args.SlotInfo.PluggedPlugInfos)
    if(args.PlugInfo != pi) args.SlotInfo.Unplug(pi);
}
```

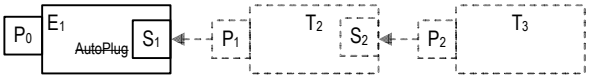
Manually plug extension when its slot is filled

Problem: Extension E_1 uses a contributor in slot S_1. Contributing extension type T_2 depends on contributions to its own slot S_2 to be operational. How can we configure slot S_1 in a way that extension E_2 of T_2 is not plugged until its dependency is resolved, i.e. its slot is filled?
Solution: Configure slot S_1 for manual plugging, create a shared instance E_2 when type T_2 attaches, and let contributing extension E_2 plug itself manually as soon as it is operational.

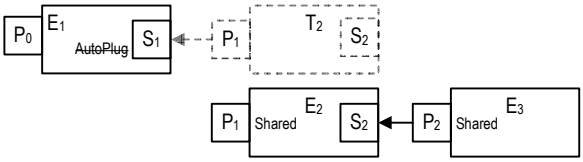
Step 1: Configure slot S_1 for manual plugging.

```
[Extension("E1")]
[Plug("P0")]
[Slot("S1",
 OnAttached="S1_Attached",
 OnPlugged="S1_Plugged",
 AutoPlug=false)]
public class E1 : I0 {
  public void S1_Attached(object s, AttachEventArgs aea) {...}
  public void S1_Plugged(object s, PlugEventArgs pea) {...}
}
```

```
[Extension("E2)]
[Plug("P1")]
[Slot("S2", ...)]
public class E2 : I1 { ... }

[Extension("E3)]
[Plug("P2")]
public class E3 : I2 { ... }
```



Step 2: When T_2 is attached, E_1 creates a shared instance E_2 of T_2, and opens E_2's slots. When slot S_2 is opened, the Plux.NET runtime automatically creates a shared instance of E_3 and plugs it into slot S_2.




```
public void S1_Attached(object s, AttachEventArgs aea) {
  ExtensionTypeInfo eti = aea.PlugTypeInfo.ExtensionTypeInfo;
  ExtensionInfo ei = eti.GetSharedInstance(true);
  ei.OpenSlots();
}
```

Step 3: When extension E_3 is plugged into slot S_2, extension E_2 becomes operational. E_2 plugs itself in all slots where its corresponding type T_2 is attached, i.e. E_2 is plugged into slot S_1.

```
[Extension("E2)]
[Plug("P1")]
[Slot("S2", OnPlugged="S2_Plugged")]
public class E2 : I1 {
  public void S2_Plugged(object s, PlugEventArgs pea) {
    pea.SlotInfo.ExtensionInfo.PlugPlugs();
  }
}
```