

**A10 Container Design**

***Engineering Design Document***

Version 1.0a

2018/01/10

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Author / Editor | Revision | Date | Comments |
| Steven Huang / Rui Zhang | 0.1 | 1/6/2018 | Initial draft |
| Zuoping | 0.2 | 1/6/2018 | Revised draft |
| Rui Zhang | 0.1 | 1/7/2018 | Revised draft |
| Peter Chuang  /Zuoping | 1.0a | 1/10/2018 | First alpha release |

Contents

[Feature Summary 3](#_Toc503365809)

[The Role of A10 GUI Container 3](#_Toc503365810)

[Definitions 4](#_Toc503365811)

[Structure of the THContainer 4](#_Toc503365812)

[Definition of objects 4](#_Toc503365813)

[UML Diagram 5](#_Toc503365814)

[Definition of Entities 5](#_Toc503365815)

[UML of Instance 6](#_Toc503365816)

[Design Patterns 8](#_Toc503365817)

[Command Pattern 8](#_Toc503365818)

[Composite Pattern 9](#_Toc503365819)

[Future Enhancement (Not in the first release) 10](#_Toc503365820)

[References 10](#_Toc503365821)

# Feature Summary

A10Container is the abstract class for all A10 containers to implement. It is based on the A10 GUI Framework to maximize modularity and maintainability. This document captures the definitions and design of A10 Container.

# The Role of A10 GUI Container

A10 GUI consists of 4 layers:

1. A10 GUI framework.
2. A10 GUI Widgets.
3. A10 Infra: Thunder or Harmony.
4. A10 Apps: OnBox UI, Harmony UI, etc.

Figure 1 shows the structure of A10 GUI. A10 Infra is based on A10 GUI framework and A10 GUI Widget is the library of App layer.

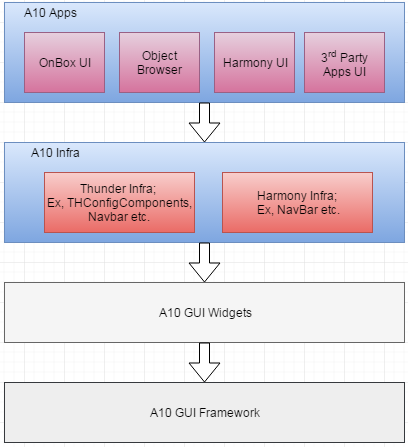


Figure 1The structure of A10 GUI

Highlighted parts in the above image are where A10Container can be. It could be a page level container or a widget level container. The page level container generally will be routed, while the widget level container generally can be reused by other applications.

# Definitions

## Structure of the THContainer

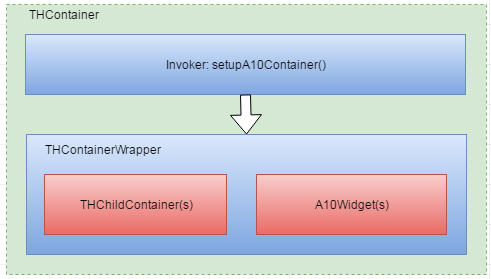


Figure 2 The structure of THContainer

### Definition of objects

1. THContainer: This is a concrete class that extends from the abstract base class A10Container. It is a common container for Thunder apps. A container originally from any App could be extracted and modified from the App layer to be a common container .
2. Invoker: It provides a setupA10Container() method to inject some common jobs to THContainerWrapper.
3. THContainerWrapper: This is a React HOC Component. Framework can use the wrapper to handle some common tasks.
4. THChildContainer(s): The THChildContainer(s) can be a constituent of other THContainer.
5. A10Widget(s): A10Widget is a stateless component mainly concerned with UI, the widget is defined in a separate doc “A10 GUI Widgets”.

# UML Diagram

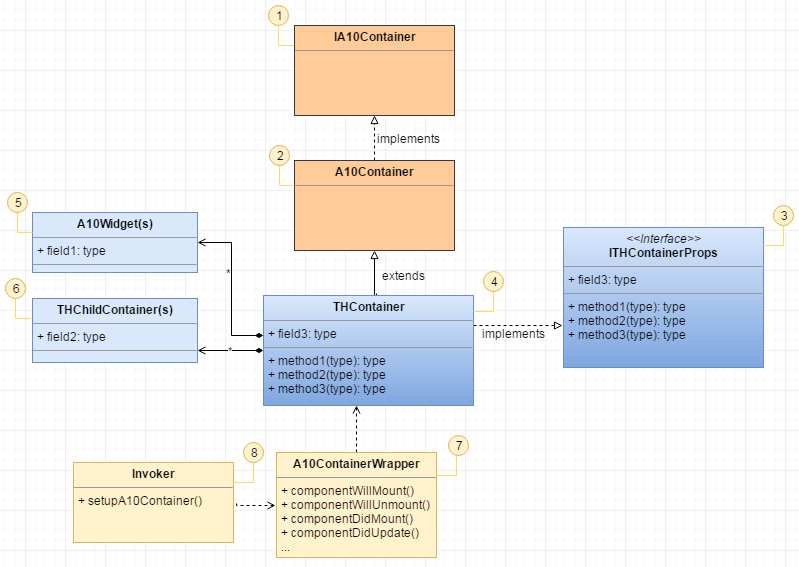


Figure 3 The UML of THContainer

### Definition of Entities

1. IA10Container: The interface for the A10 Container.
2. A10Container: Abstract class for all containers.
3. ITHContainerProps: The interface for THContainer.
4. THContainer: THContainer extends A10Container and implements the interface ITHContainerProps. It is composed of A10Widgets or other THContainers.
5. A10Widget(s): The component(s) from A10 GUI Widgets.
6. THChildContainer(s): The THChildContainer(s) can be a constituent of other THContainer.
7. A10ContainerWrapper: This is the entry point for a Container creation. Framework can use the wrapper to handle some common behavior.
8. Invoker: The invoker provided a setupA10Container() method to inject some common tasks.

### UML of Instance

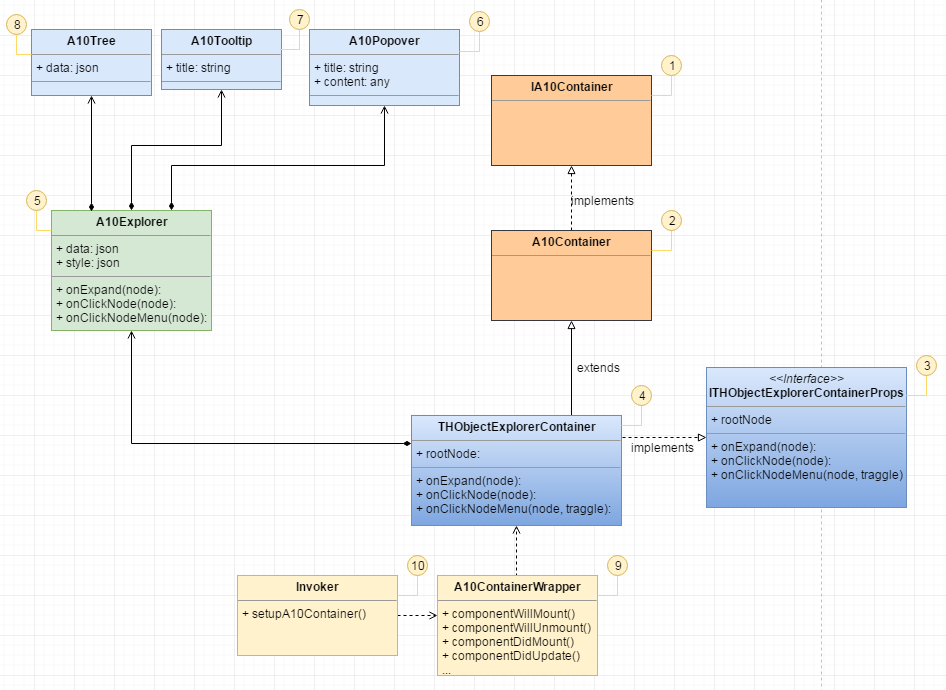


Figure 4 The UML of THObjectExplorerContainer

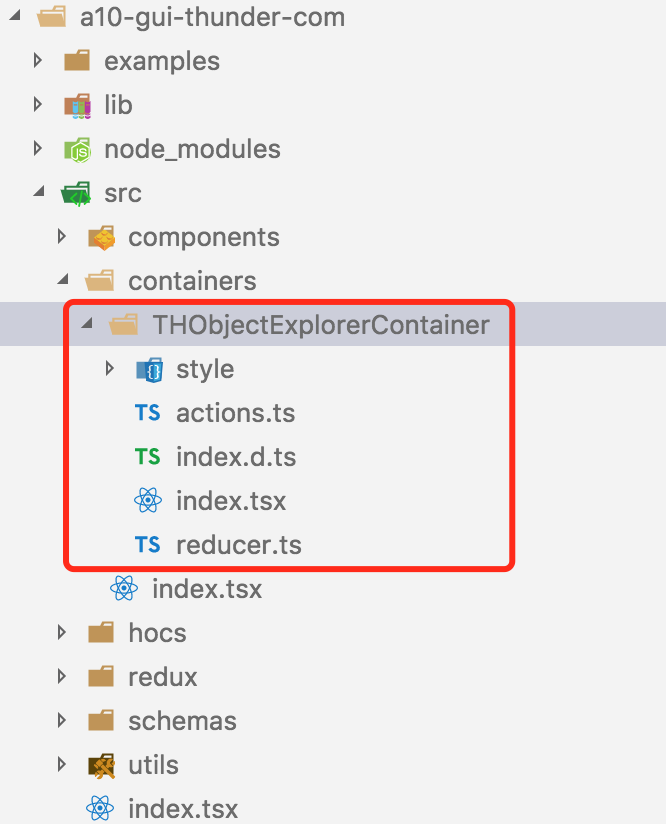


Figure 5: The THObjectExplorerContainer folder

The above image lists all of the related THObjectExplorerContainer files. The index.tsx contains the THObjectExplorerContainer class. index.d.ts defines the interface of THObjectExplorerContainer.

# Design Patterns

### Command Pattern

A10ContainerWrapper creates a Container by using Command design pattern as shown in figure 6. The benefits of this Command approach:

1. Separation of concerns. The one that creates the operation, which is called invoker, does not need to worry about how the operation is being performed by a receiver. The operation binding of invoker and receiver is decoupled.
2. We could have callback before or after the Container creation. Therefore, we could inject some global and shared behavior in each Container.

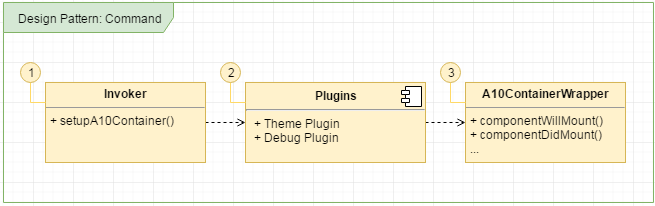


Figure 5 The UML of Design Pattern: Command

1. Command: Defines the interface of the command and declares the method to execute.
2. Invoker: The invoker provides setupA10Container() method.
3. A10ContainerWrapper: This is the entry point for a Container creation. Framework can use the wrapper to handle some common behavior.

### Composite Pattern

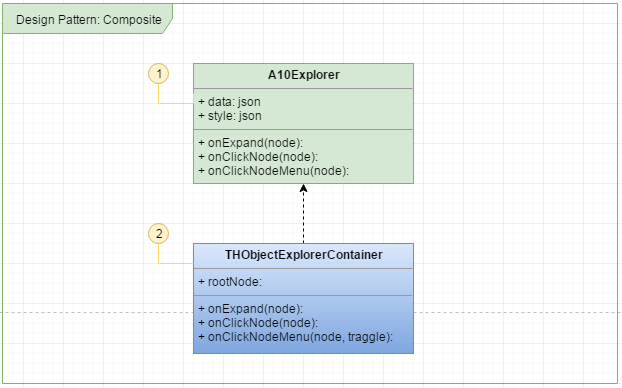


Figure 6 The UML of Design Pattern: Composite

1. A10Explorer: One example class of the A10Widget
2. THObjectExplorerContainer: Comprises A10Explorer and THObject.

|  |
| --- |
| **Code: Interface of A10Container**// Interface of A10Container  export default interface IA10Container {  }  // A10 abstract container  export default Class A10Container extends React.Component<IA10Container> {  }  export default setupA10Container(A10Container) |

Figure 7: Define A10Container and the interface

A10Container is abstract class for all containers. It extends A10Component and implements IA10Container interface.

**Code: Implementation of THObjectExplorerContainer**

|  |
| --- |
| export default interface ITHObjectExplorerContainerProps {  }  export default Class THObjectExplorerContainer extends A10Container<ITHObjectExplorerContainerProps> {  } |

Figure 8: Define THObjectExplorerContainer and the interface

THObjectExplorerContainer extends A10Container and implemented the interface ITHObjectExplorerProps.

## Future Enhancement (Not in the first release)

Please refer to the framework document for theme and locale design.

# References

<https://redux.js.org/docs/basics/Store.html>

<https://redux.js.org/docs/basics/Reducers.html>

<https://redux.js.org/docs/basics/Actions.html>