

Software Design Document

GodHands

GH-100-SDD-001

|  |  |
| --- | --- |
| Document Ref | GH-100-SDD-001 |
| Version | 0.1 |
| Date | 2020-01-09 |

|  |  |  |  |
| --- | --- | --- | --- |
| Rev | Date | Status | Description |
| 0.1 | 2020-01-09 | Draft | Initial pass |
|  |  |  |  |

|  |  |
| --- | --- |
| References |  |
|  |  |
|  |  |

No licence is granted under any patent or any patent rights of Valendian. Information furnished by Valendian is believed to be accurate. No responsibility is assumed by Valendian for its use, nor for any infringements on the rights of other parties that may result from the use of the information herein. All specifications are subject to change without notice

Contents

[1 Overview - 1 -](#_Toc29565036)

[1.1 Purpose of this Document - 1 -](#_Toc29565037)

[1.2 Purpose of this Software - 1 -](#_Toc29565038)

[1.3 Main Goals - 1 -](#_Toc29565039)

[2 Reference Documents - 2 -](#_Toc29565040)

[2.1 Software Design - 2 -](#_Toc29565041)

[2.2 Reference Materials - 2 -](#_Toc29565042)

[3 System Overview - 3 -](#_Toc29565043)

[3.1 System Architecture - 3 -](#_Toc29565044)

[3.2 Major Subsystems - 4 -](#_Toc29565045)

[4 Build Configuration - 5 -](#_Toc29565046)

[4.1 Dependencies and Required Tools - 5 -](#_Toc29565047)

[4.2 Build Environment - 5 -](#_Toc29565048)

[4.3 Test Bench - 5 -](#_Toc29565049)

[4.4 Profiling - 5 -](#_Toc29565050)

[5 Logical Design - 6 -](#_Toc29565051)

[5.1 Logger Subsystem Design - 6 -](#_Toc29565052)

[5.2 Config Subsystem Design - 6 -](#_Toc29565053)

[5.3 Ram Disk Subsystem Design - 8 -](#_Toc29565054)

[5.3.1 Lazy Loading - 8 -](#_Toc29565055)

[5.4 Publish Subscribe Subsystem Design - 9 -](#_Toc29565056)

[5.4.1 Publish - 9 -](#_Toc29565057)

[5.4.2 Register and Unregister - 10 -](#_Toc29565058)

[5.4.3 Subscribe and Unsubscribe - 11 -](#_Toc29565059)

[5.5 Undo Redo Subsystem Design - 12 -](#_Toc29565060)

[5.5.1 Exec - 12 -](#_Toc29565061)

[5.5.2 Undo - 12 -](#_Toc29565062)

[5.5.3 Redo - 12 -](#_Toc29565063)

[5.6 Data Binding Subsystem Design - 13 -](#_Toc29565064)

[5.6.1 Data Binding Performs Ram Disk IO - 13 -](#_Toc29565065)

[5.6.2 Data Binding Performs Undo / Redo - 13 -](#_Toc29565066)

[5.6.3 Data Binding Performs Publishing - 13 -](#_Toc29565067)

# 1 Overview

This is a Software design document.

## 1.1 Purpose of this Document

This is a main section under the above heading.

## 1.2 Purpose of this Software

This is a main section under the above heading.

## 1.3 Main Goals

This is a main section under the above heading.

# 2 Reference Documents

This is a Software design document.

## 2.1 Software Design

This is a main section under the above heading.

|  |  |
| --- | --- |
| Document | Description |
| GH-100-SRS | Software Requirements Specification |
| GH-100-SDD | Software Design Document |
| GH-100-SUD | Software User Document |

Table 2.1: Software Design Documents

## 2.2 Reference Materials

This is a main section under the above heading.

# 3 System Overview

The Software is divided into two major sections: System and Mission Software. The Mission software contains all things which are solely related to this specific application. The System software manages complexities related to the operating system and resource management which would complicate the design if included in the Mission software. In the source code the Model and the View contain Mission Software code, while the rest of the code base is considered System Software.

## 3.1 System Architecture

The top-level processes and the interconnect between them. The View is Mission Software and manages the User Interface. The Model is also Mission Software and manages the User Data. All other modules collectively form the System Software which is the Backend of the system.

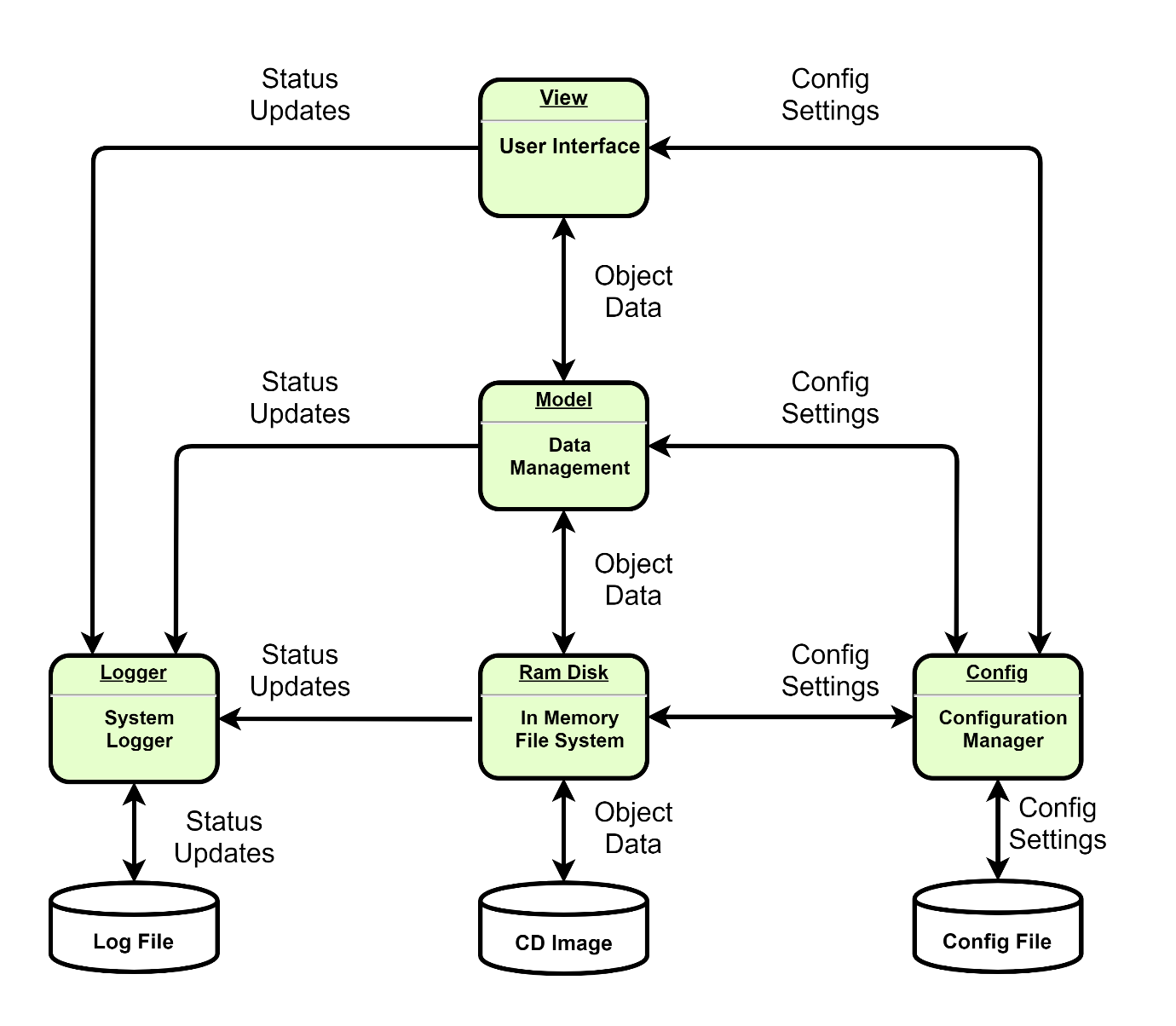


Figure 3.1: System Overview

## 3.2 Major Subsystems

These are the main subsystems and dataflow. The red lines indicate critical data-flow paths.

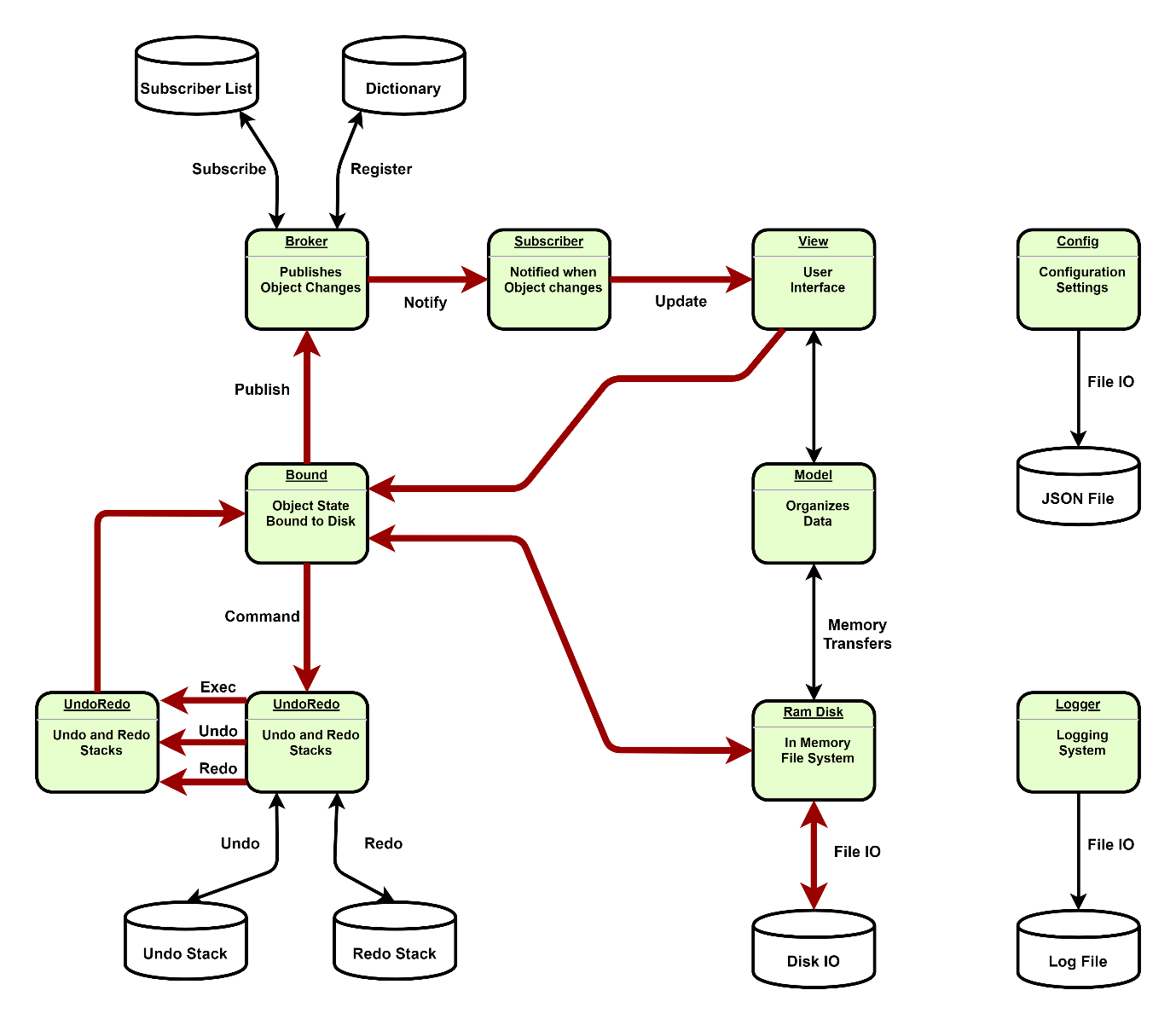


Figure 3.2: Major Subsystems

## 3.3 Data Model

This is a main section under the above heading.

# 4 Build Configuration

This is a Software design document.

## 4.1 Dependencies and Required Tools

This is a main section under the above heading.

## 4.2 Build Environment

This is a main section under the above heading.

## 4.3 Test Bench

This is a main section under the above heading.

## 4.4 Profiling

This is a main section under the above heading.

# 5 System Software Design

The purpose of the System Software is to simplify the design of the Mission Software by isolating accidental complexity from intended complexity. The System Software provides the necessary execution environment in which the Mission Software may operate. The following chapter describes the main subsystems, defines their key responsibilities and outlines how they interact with each other.

## 5.1 Logger Subsystem Design

The Logger Subsystem maintains the system log. Modules print status updates to the Logger and the Logger commits those changes to file. Messages can be displayed in the Status Bar and Error / Warning messages are displayed in a Message Box.

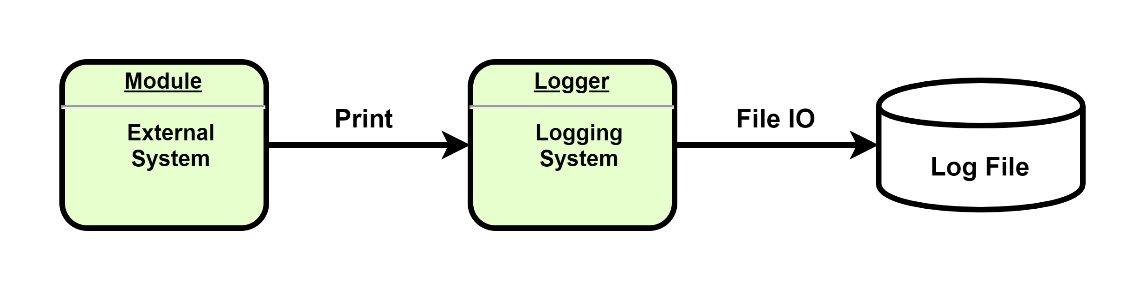


Figure 5.1: Logger Subsystem

## 5.2 Config Subsystem Design

The Config Subsystem serializes system settings to/from a JSON file. The file is loaded into a Class which is accessible by external modules. A third Party Json Serializer is used.

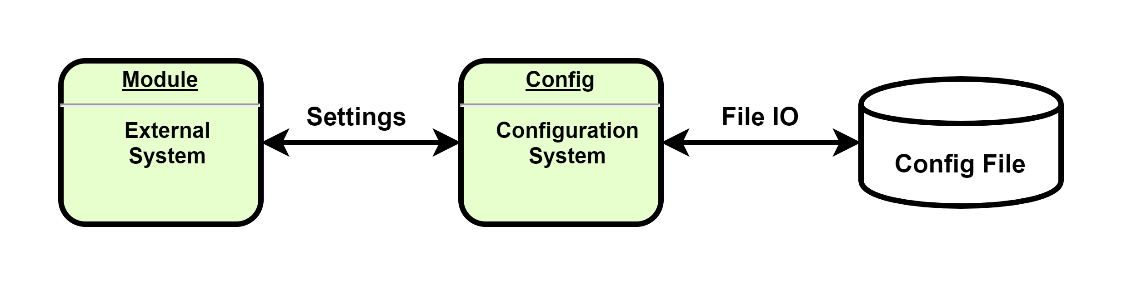


Figure 5.2: Config Subsystem

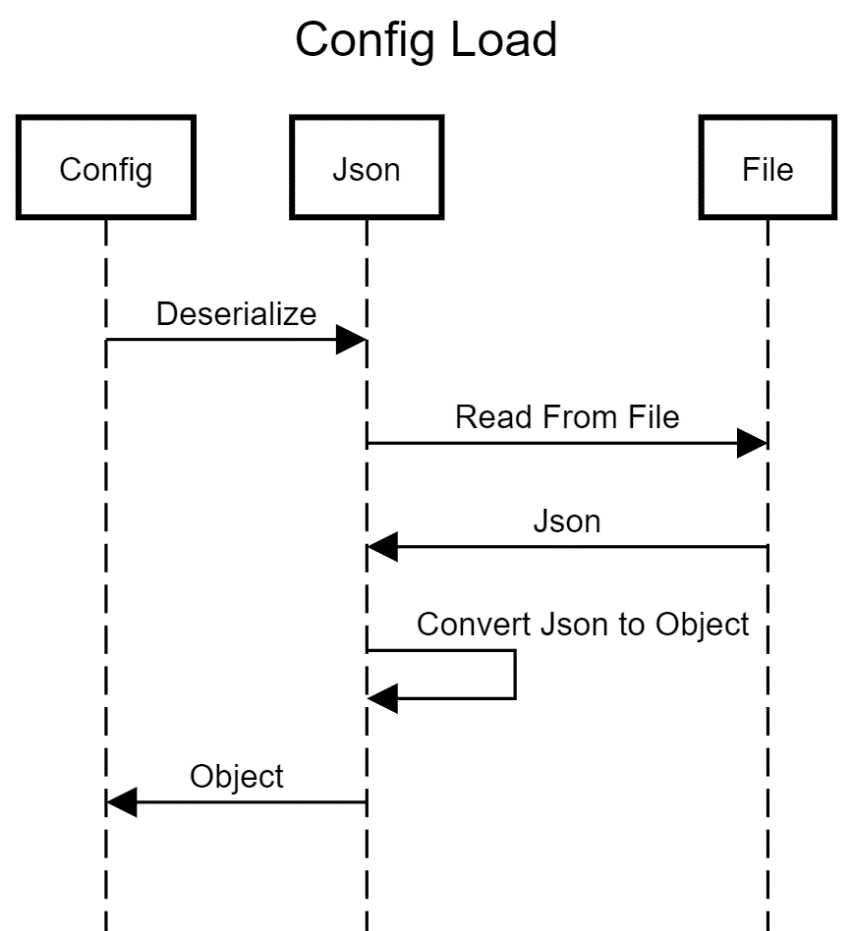


Figure 5.2.1: Config Load Sequence Diagram

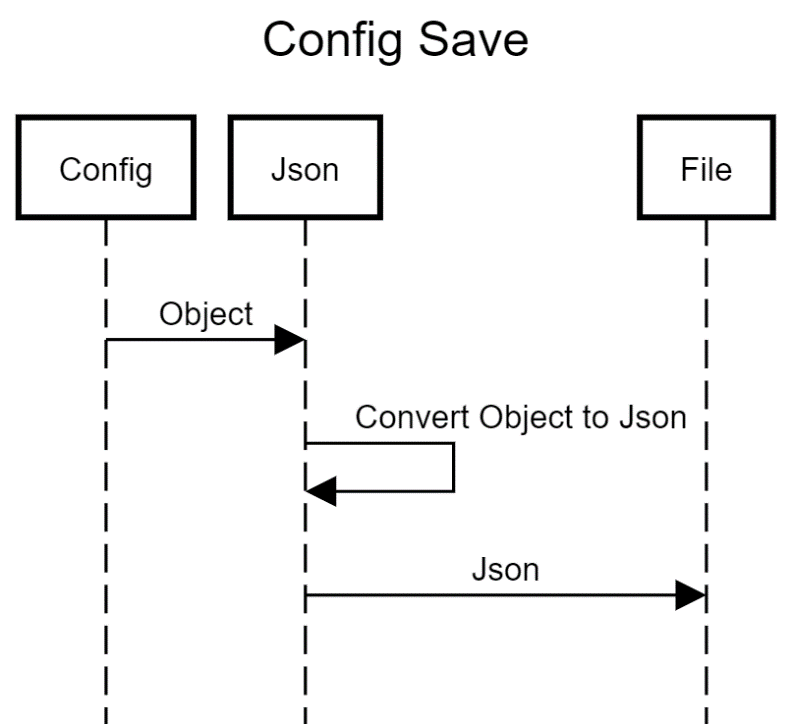


Figure 5.2.2: Config Save Sequence Diagram

## 5.3 Ram Disk Subsystem Design

The Ram Disk is an in-memory File System which performs Block IO to read and write date to/from disk file. Performance of this module is critical to system responsiveness.

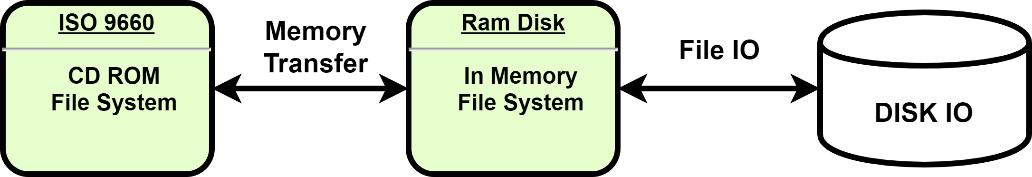


Figure 5.3: RamDisk Subsystem

### 5.3.1 Lazy Loading

When data is not in-memory the RamDisk read from file. All writes are immediately committed to Disk. The user does not have to save the file explicitly.

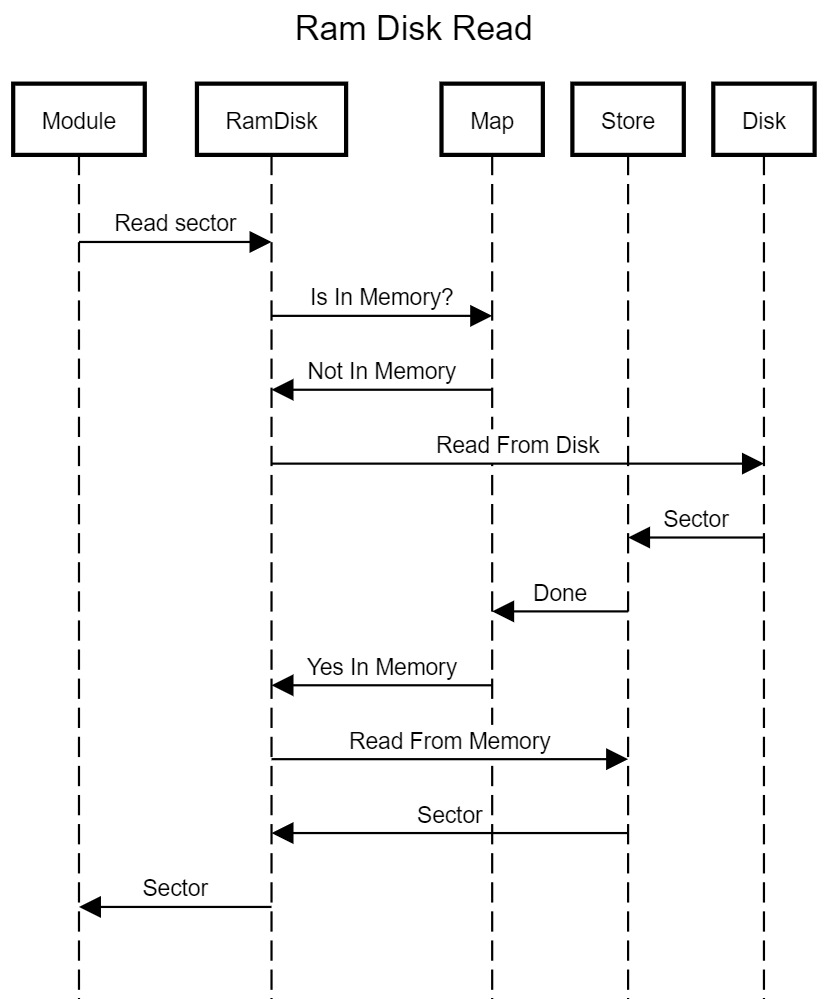


Figure 5.3.1: RamDisk Read Sequence Diagram

## 5.4 Publish Subscribe Subsystem Design

The publish subscribe subsystem implements the observer pattern. Subscribers can apply to the Broker to receive notifications when an object’s state changes. Changes are Published to the object and the Broker notifies all Subscribers.

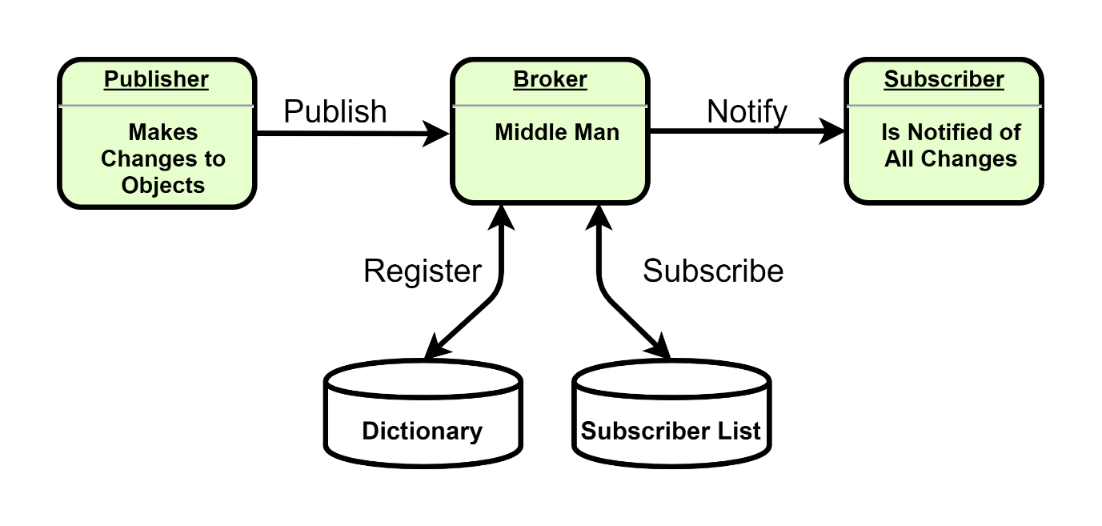


Figure 5.4: Publish Subscribe Subsystem

### 5.4.1 Publish

Changes must be Published to an object. When Publishing changes the Broker retrieves a list of subscribers which are then notified.

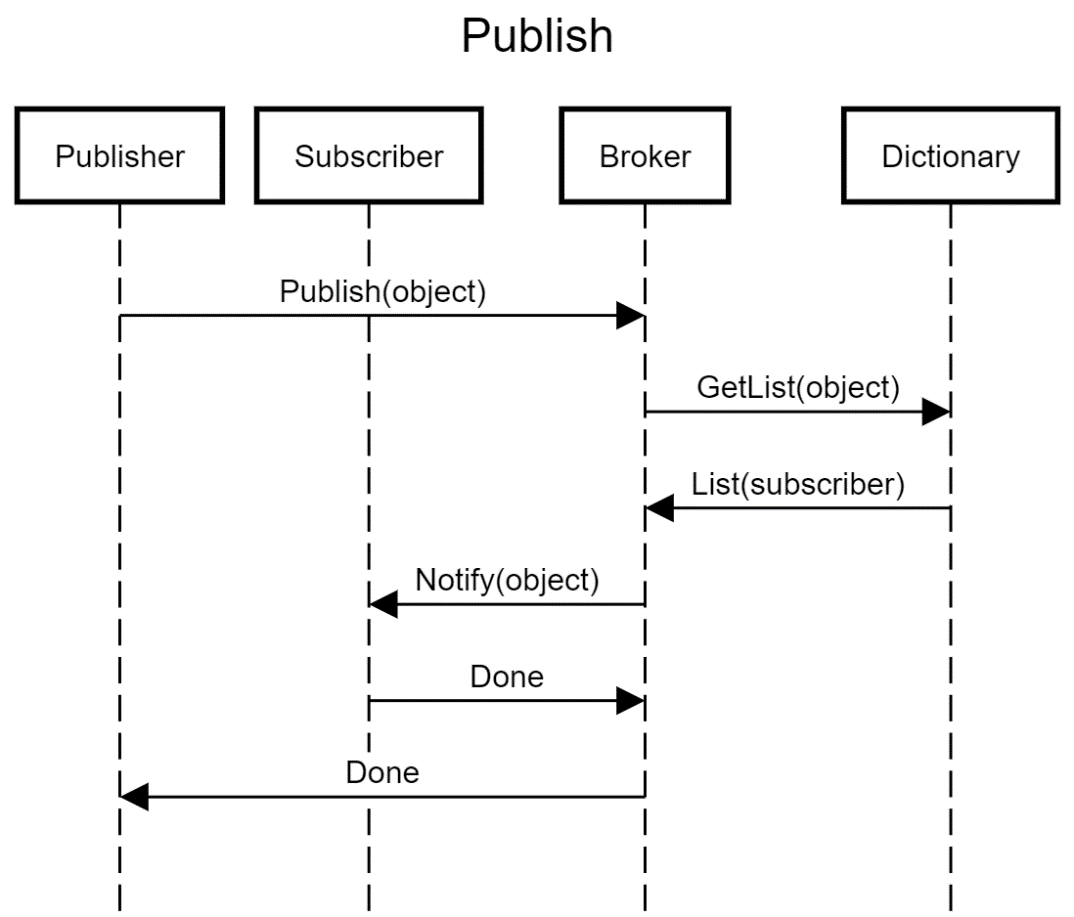


Figure 5.4.1: RamDisk Read Sequence Diagram

### 5.4.2 Register and Unregister

Objects are registered with the Broker. Objects are assigned a URL when being Registered. The broker inserts the object into a dictionary using the URL as the associated key. An empty list of Subscribers is created and associated with the same URL during Registration.

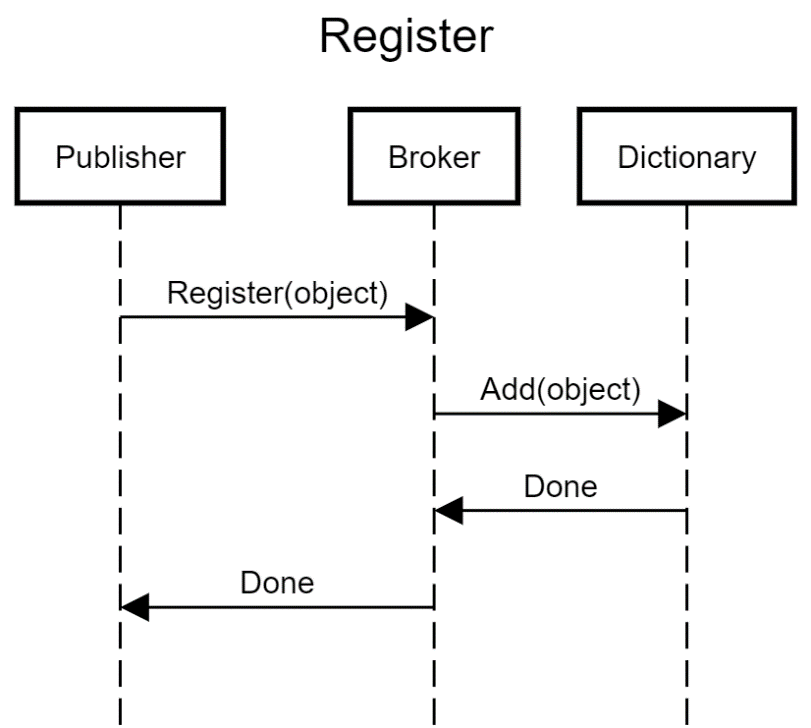


Figure 5.4.2: Broker Register Sequence Diagram

When an object is destroyed it must be Unregistered. The Broker will remove both the object and its list of Subscribers from the Dictionary.

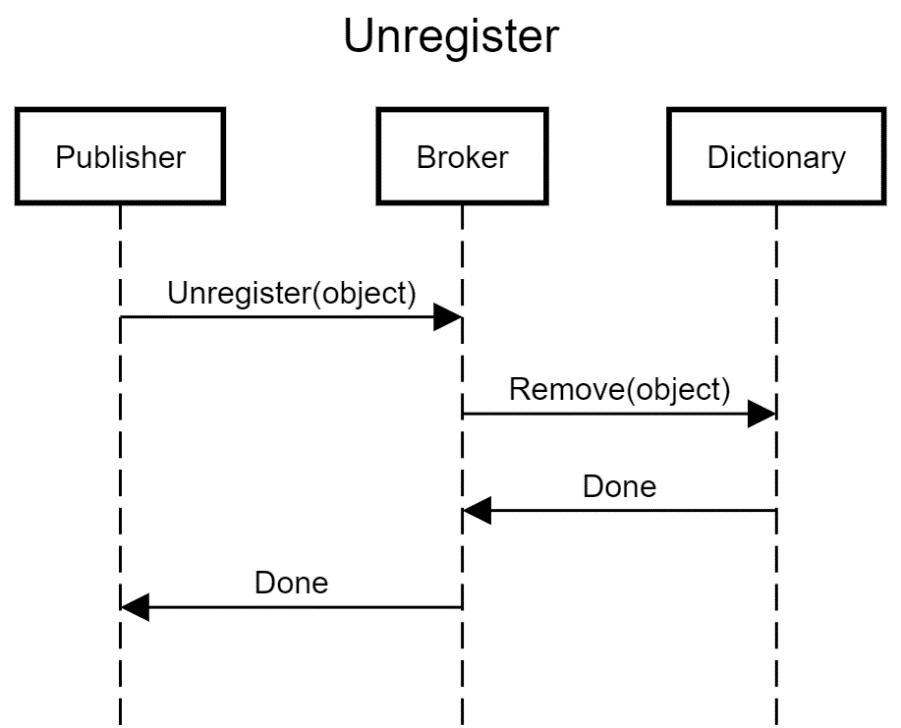


Figure 5.4.2: Broker Unregister Sequence Diagram

### 5.4.3 Subscribe and Unsubscribe

Any observer may Subscribe to get Notifications from the Broken when changes are Published to the object. Each object has a list of subscribers associated with it. When the observer Subscribes to the object, the Broker simply records the Subscriber in the relevant list.

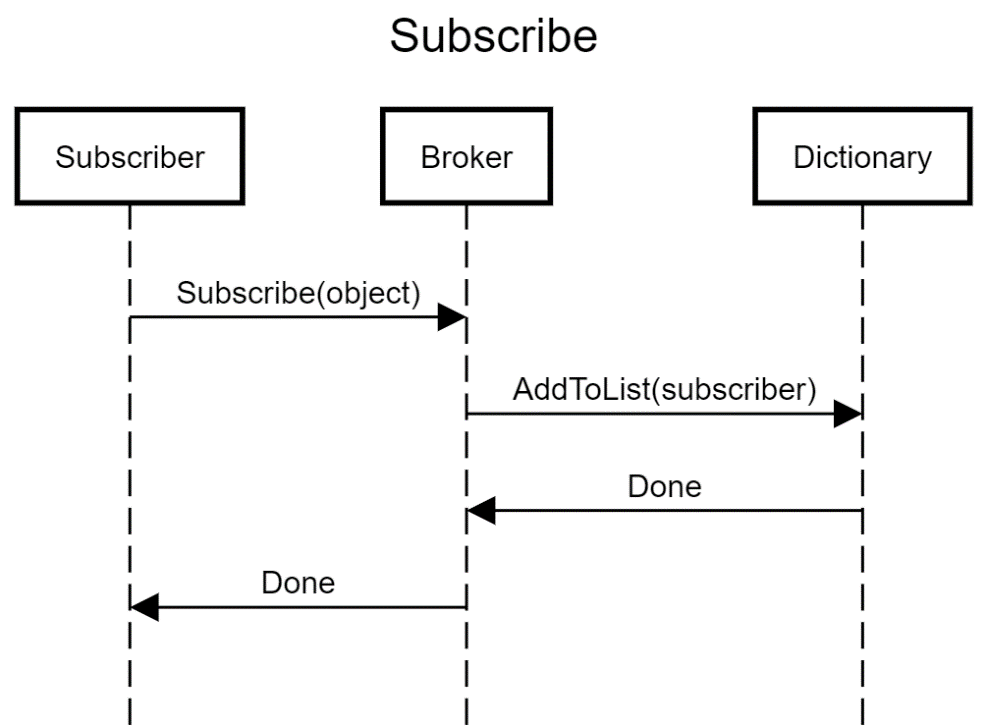


Figure 5.4.3: Broker Subscribe Sequence Diagram

When an observer is destroyed or is assigned to a different object, then the observer must Unsubscribe from the object. When this occurs, the Broker will remove the Subscriber from the relevant list.

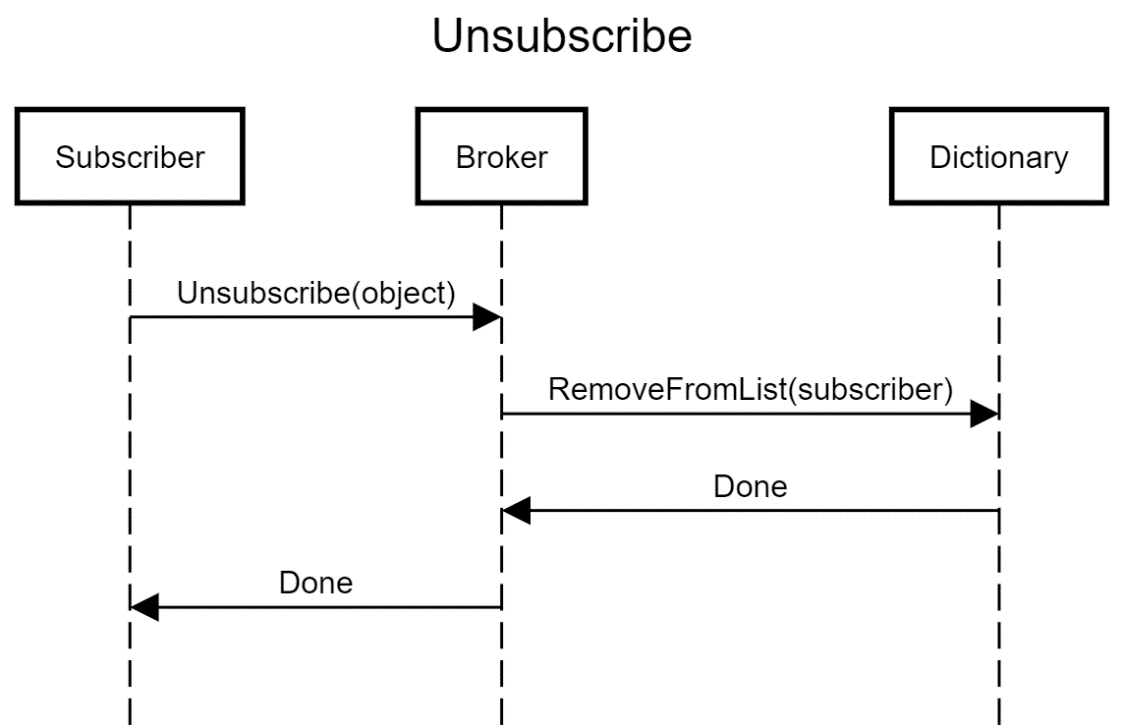


Figure 5.4.3: Broker Subscribe Sequence Diagram

## 5.5 Undo Redo Subsystem Design

The Undo Redo subsystem tracks actions in the form of Command objects and allows changes to be undone at a later stage. The Command object performs the actual work and also stores the data needed to undo / redo the actions later.

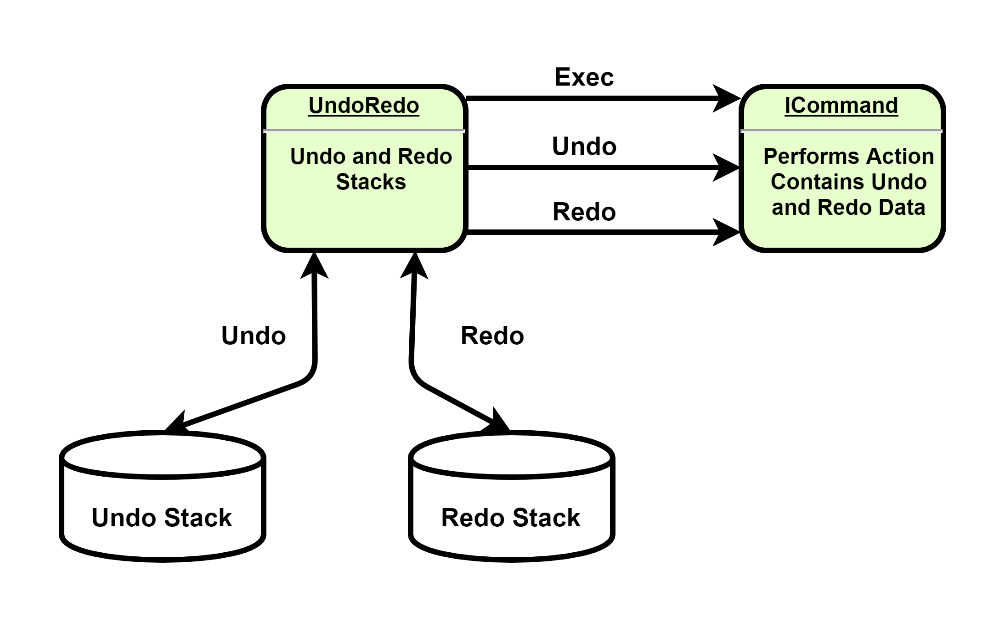


Figure 5.5: Undo Redo Subsystem

### 5.5.1 Exec

The Exec action is performed when a action is first initiated. It is most often the same as Redo except in a few situations where initial preparations must be made (such as allocating resources).

### 5.5.2 Undo

When the Undo action is performed the most recent action is popped from the undo stack and the associated commands Undo method is invoked. The action is then added to the redo stack.

### 5.5.3 Redo

When the Redo action is performed the most recent undone action is popped from the redo stack and the associated commands Redo method is invoked. The action is then added to back into the undo stack.

## 5.6 Data Binding Subsystem Design

Externally the Model system presents a collection of objects. The backend of the system streams object data to/from disk storage. This Dataflow goes through the RamDisk. The RamDisk presents a Binding interface to simplify the process from the objects end point by providing a means of binding fields in the object to byte arrays on disk.

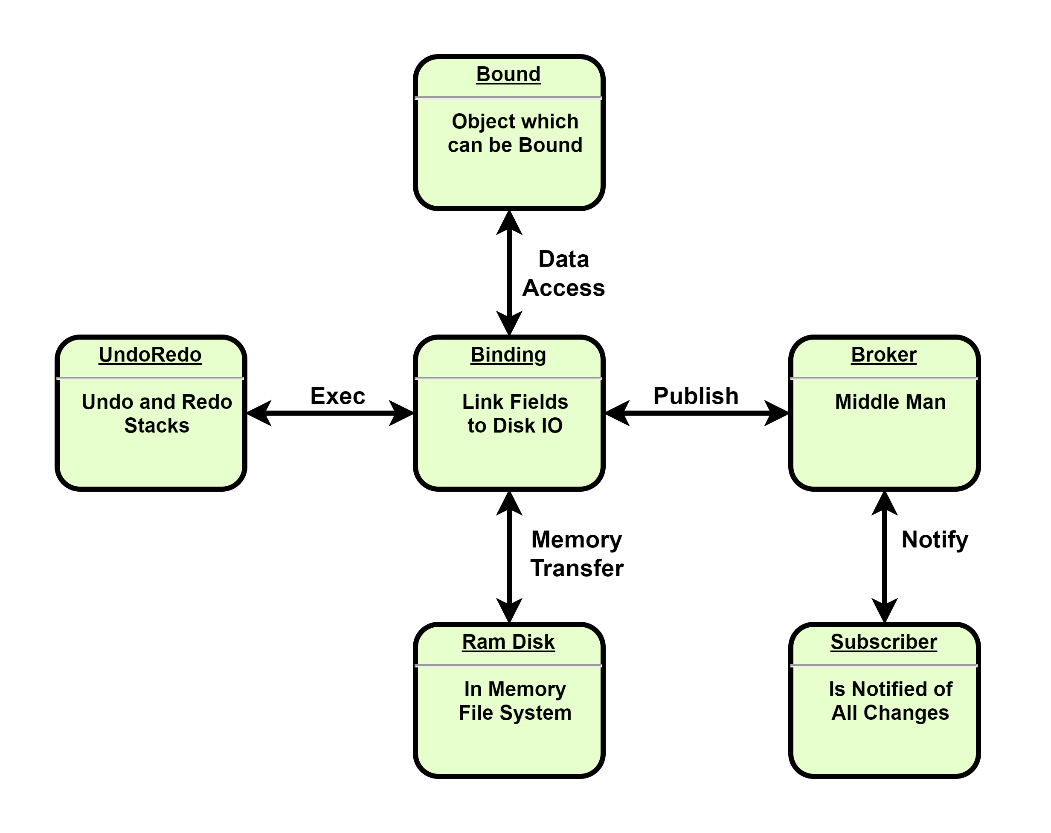


Figure 5.6: Data Binding Subsystem

### 5.6.1 Data Binding Performs Ram Disk IO

Data Binding routes access to its own members through the RamDisk system. All reads come from Disk and all writes are immediately committed to Disk.

### 5.6.2 Data Binding Performs Undo / Redo

The Data Binding module tracks changes made to an object so that they can be undone at a later stage. The Data Binding is responsible for interacting with the Undo Redo subsystem.

### 5.6.3 Data Binding Performs Publishing

The Data Binding module announces changes made to an object so that observers may respond accordingly. The Data Binding is responsible announcing changes to the Broker.

# 6 Mission Software Design

The Mission Software is purely application specific software which requires an execution environment provided by the System Software.

## 6.1 View Subsystem Design

The View represents the User Interface. The View is responsible for gather user input and presenting data to the User. The actual processing of data is handled within the Model.

## 6.2 Model Subsystem Design

The Model manages the User Data, performs data processing and organizes the data in a structured manner. The model requires a Ram Disk to operate.



**Email:** [**info@valendian.com**](mailto:info@valendian.com) **Web: www.valendian.com**

©2020 Valendian Ltd. All rights reserved. Trademarks and registered trademarks are the property of their respective owners