

# Table of Contents

[Introduction](#Top_of_index_html)

[Baisc](#Top_of_10_basic_html)

[Live-update Server](#Top_of_11_live_update_server_htm)

[Project List](#Top_of_20_project_list_html)

[InQuire App](#Top_of_21_inquire_app_html)

[InQuire Web](#Top_of_22_inquire_web_html)

[Gerrit](#Top_of_23_gerrit_html)

Table of Contents

[Introduction](#Top_of_index_html) 1.1

[Baisc](#Top_of_10_basic_html) 1.2

[Live-update Server](#Top_of_11_live_update_server_htm) 1.2.1

[Project List](#Top_of_20_project_list_html) 1.3

[InQuire App](#Top_of_21_inquire_app_html) 1.3.1

[InQuire Web](#Top_of_22_inquire_web_html) 1.3.2

[Gerrit](#Top_of_23_gerrit_html) 1.3.3

# Introduction

Version 0.1.1

本交接文檔使用 [Gitbook](https://toolchain.gitbook.com/) 撰寫，並提供以下格式：

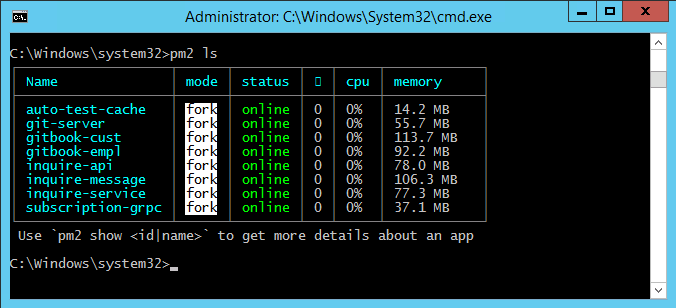
* GitBook: <http://inquire.insyde.com:24880/>
* PDF: [Click to Download](http://inquire.insyde.com:24880/resources/book.pdf)
* EPUB: [Click to Download](http://inquire.insyde.com:24880/resources/book.epub)
* DOCX: [Click to Download](http://inquire.insyde.com:24880/resources/book.docx)

# Basic

目前僅描述 Live-update Server 上的設定參數，除了本文件外還另有一份手寫的 Live-update Server 上各個 Service 的 Administrator 帳號密碼

## Live-update Server

Live-update 各個 Service 所使用的 Port 整理在 C:\README.md，目前有註冊啓動時會執行的 Script C:\startup.ps1，其內容是使用 [PM2](https://github.com/Unitech/pm2) 去啓動管理各個 Service



### Directory Structure

各個 Service 的 Binary 並不是放在 C:\Program Files 之下，而是 C:\01.web\_service 目錄

* Service Binary
  + [Apache HTTP Server](https://httpd.apache.org/)
    - Path: C:\01.web\_service\Apache2.2
    - Target: 原本使用 80 Port 的 Web Server，目前棄用
  + [Gitea](https://gitea.io/)
    - Path: C:\01.web\_service\gitea
    - Target: 方便部署用的 Git Server
  + [MariaDB](https://mariadb.org/)
    - Path: C:\01.web\_service\MariaDB 10.1
    - Target: 與 MySQL 相容，目前未使用
  + [Nginx](http://nginx.org/)
    - Path: C:\01.web\_service\nginx-1.11.9
    - Target: Web Server 兼 Reverse Proxy Server，使用 80 與 8080 Port
  + [PostgreSQL](https://www.postgresql.org/)
    - Path: C:\01.web\_service\PostgreSQL9.6
    - Target: 功能比較強大的 RDBMS，目前 Subscription 會拿來存儲已讀記錄與我的最愛
  + [RabbitMQ](https://www.rabbitmq.com/)
    - Path: C:\01.web\_service\RabbitMQ Server
    - Target: Message Queue Server，原本是要給 InQuire 的 Notification Web API 使用的

我已經寫了一些基本的 Batch Script 已方便重啓 Service，亦放在 C:\01.web\_service 下

* Nginx Restart Steps
  1. C:\01.web\_service\taskkill\_nginx.bat
  2. C:\01.web\_service\stop\_nginx.bat
  3. C:\01.web\_service\start\_nginx.bat
* Directory Tree

C:\  
+---01.web\_service  
| +---Apache2.2  
| +---gitea  
| +---MariaDB 10.1  
| +---nginx-1.11.9  
| +---PostgreSQL9.6  
| +---RabbitMQ Server  
| \---\_projects  
| +---api.inquire  
| +---auto-test-cache  
| +---doc.inquire  
| | +---cust  
| | \---empl  
| +---message.inquire  
| +---server.inquire  
| \---subscription-grpc  
+---  
\_file\_  
storage  
| +---FileCenter  
| +---gitea-repositories  
| +---LiveUpdate  
| \---\_reverse  
\---  
\_sql\_  
data  
 +---LiveUpdate  
 \---\_reverse

# Project List

目前手頭上的 Project 都可以在 [InQuire Git Server](http://inquire.insyde.com:3000/) 上找到，而 Repository 的實際存放位置在 Live-update Server 上的 C:\\_file\_storage\gitea-repositories\。每一個 Project 都會有 README.md 概述解說架構，如何編譯或執行，因此本文檔並不會詳述，只會略微帶過

## InQuire App

### Desktop Application

* Projects
  + [InQuire\_WL / InQuire\_WL](http://inquire.insyde.com:3000/InQuire_WL/InQuire_WL)
* To-do
  + [ ] [升級 CEFPython 到 v66.0，感謝 Lampix 願意資助](https://github.com/cztomczak/cefpython#funding-for-v660-release)
  + [ ] Linux 版本安裝腳本 (for Live-update)
  + [ ] 使用 Python3 與 wxPython4.0 重構

InQurie 是使用了 Python2 寫成的 Desktop Application，其目的是在不改變公司現有系統的前提下，整合所有服務成爲唯一入口。大部分的 Dependencies 已經透過 [Pipenv](https://docs.pipenv.org/) 管理，需要手動安裝的部分請參考 README.md

構成 InQuire 主體有三大 Framework：[wxPython Classic](https://wxpython.org/news/wxpython-3.0.2.0-release/index.html)、[CEF Python](https://github.com/cztomczak/cefpython) 以及 [Twisted](https://twistedmatrix.com/trac/)，分別負責 GUI、WebView 以及 Event Loop

本應用使用 [PyInstaller](http://www.pyinstaller.org/) 編譯，在 Windows 下使用 [InnoSetup](http://www.jrsoftware.org/isinfo.php) 打包爲 Installer。目前 Linux 版本僅於 [Ubuntu 16.04 LTS](http://releases.ubuntu.com/16.04/) 與 [Ubuntu 14.04 LTS](http://releases.ubuntu.com/14.04/) 測試。目前尚未撰寫安裝用的 Bash Script，不過在 Ubuntu 上其實也可以使用 Python Script 來處理

### System Tray

* Projects
  + [kevin.chen / InQuireTrayTest](http://inquire.insyde.com:3000/kevin.chen/InQuireTrayTest)
  + [kevin.chen / LiveUpdateTray](http://inquire.insyde.com:3000/kevin.chen/LiveUpdateTray)

開發語言爲 Kotlin 與 Java，目前公司最多人使用的 Linux Desktop 環境是 Ubuntu Desktop，Ubuntu 在 18.04 之前的 LTE 版本使用的是 Unity，而 Unity 的 System Tray 與 Windows 相比少了部分功能，而高層希望在 Linux 的行爲要與在 Windows 一樣，因此改使用 [AWT](https://www.wikiwand.com/en/Abstract_Window_Toolkit) 來畫出 System Tray。

編譯流程使用 [Gradle](https://gradle.org/) 管理，目錄架構就是一般標準的 Java 目錄架構。要注意的是 Ubuntu 14.04 LTS (Trusty Tahr) 的 default-jdk 是 Java 7，因此不能直接使用 Lambda

### Start Menu

* Projects
  + [InQuire\_WL / inquire-react-start-menu](http://inquire.insyde.com:3000/InQuire_WL/inquire-react-start-menu)

使用 [Create React App](https://github.com/facebook/create-react-app) 創建的 Web App，會透過 InQuire 的 Twisted Web Server 得到當前使用者可使用的 Service 列表並渲染在網頁上

## InQuire Web

### Web Application and Web API

* Projects
  + [InQuire\_WL / inquire-subscription](http://inquire.insyde.com:3000/InQuire_WL/inquire-subscription)
  + [InQuire\_WL / inquire-server](http://inquire.insyde.com:3000/InQuire_WL/inquire-server)
  + [InQuire\_WL / inquire-api](http://inquire.insyde.com:3000/InQuire_WL/inquire-api)
* To-do
  + [ ] InQuire\_WL / inquire-subscription 可以列出當前使用者所有可見的 Publications 列表，無需切換 Channel

InQuire 所提供的服務絕大部分都與 Web 相關，當應用需求不是那麼適合使用 wxWidgets 來呈現的情況下，使用傳統的 Web Site 或 [Single-page Application](https://medium.com/@hulitw/introduction-mvc-spa-and-ssr-545c941669e9) 在開發及維護上是比較方便的

這三個 Project 皆使用 [Node.js](https://nodejs.org/en/) 開發，在 Live-update Server 上使用 [nvm-windows](https://github.com/coreybutler/nvm-windows) 來管理 Node.js 版本，使用 [PM2](https://github.com/Unitech/pm2) 來管理各個 Process。這些 Service 在 Live-update Server 上使用的 Port 請參考 Live-update Server 的 C:\README.md，啓動腳本請參考 C:\startuo.ps1

InQuire\_WL / inquire-subscription 爲 Single-page Application，使用 [React Starter Kit](https://github.com/kriasoft/react-starter-kit) 作爲 Boilerplate，其目的是將 [原本於 e-Insyde 的 Publication](https://sys.insyde.com/Publication/PubList.asp) 加上 Group 與 Category 屬性

InQuire\_WL / inquire-server 使用 [Express' application generator](https://github.com/expressjs/generator)，，使用 [Pug (Jade)](https://github.com/pugjs/pug) 作爲 Template Engine，主要提供 File Center 與 Training(Youtube) 的頁面

InQuire\_WL / inquire-api 與 InQUire\_WL / inquir-server 同樣都使用 Express.js，主要提供 Email Verification 與 Skype for Bisiness 的 Web API

### Web API

* Projects
  + [InQuire\_WL / auto-test-cache](http://inquire.insyde.com:3000/InQuire_WL/auto-test-cache)
  + [InQuire\_WL / auto-test-grpc](http://inquire.insyde.com:3000/InQuire_WL/auto-test-grpc)
  + [InQuire\_WL / subscription-grpc](http://inquire.insyde.com:3000/InQuire_WL/subscription-grpc)
* To-do
  + [ ] 使用 InQuire\_WL / auto-test-grpc 取代 InQuire\_WL / auto-test-cache

這三個服務皆使用 Golang 撰寫，其目的在於 Serivce Owner 爲提供 Web API 而是提供 Database 權限直接讀取，而頻繁讀取可能造成 Database 效能低落，因此自行建造具 Cache 機制的 Web API Service 來避免 Client 直接讀取 Database

Dependencies 以透過 [govendor](https://github.com/kardianos/govendor) 來進行管理，目錄架構參考了 [Standard Go Project Layout](https://github.com/golang-standards/project-layout)。其中 InQuire\_WL / auto-test-cache 是較早期的應急 Project，用於改善執行 MySQL Query 需要數十秒的問題，而之後的新 Project InQuire\_WL / auto-test-grpc 其目的是希望 Client 不要直接讀取 Database，而是全部透過 Web API 來進行 Auto-test 的相關操作，但目前尚未完工

InQuire\_WL / subscription-grpc 是針對 Notification 的資料 Cache，Subscription Message 以及 User 對應的 Channel 資料皆是存放於公司的 ERP Database，但因效能不佳問題，需在本地備份避免頻繁存取

### User Guide

* Projects
  + [InQuire\_WL / inquire-user-guide](http://inquire.insyde.com:3000/InQuire_WL/inquire-user-guide)
  + [InQuire\_WL / inquire-gitbook-theme](http://inquire.insyde.com:3000/InQuire_WL)

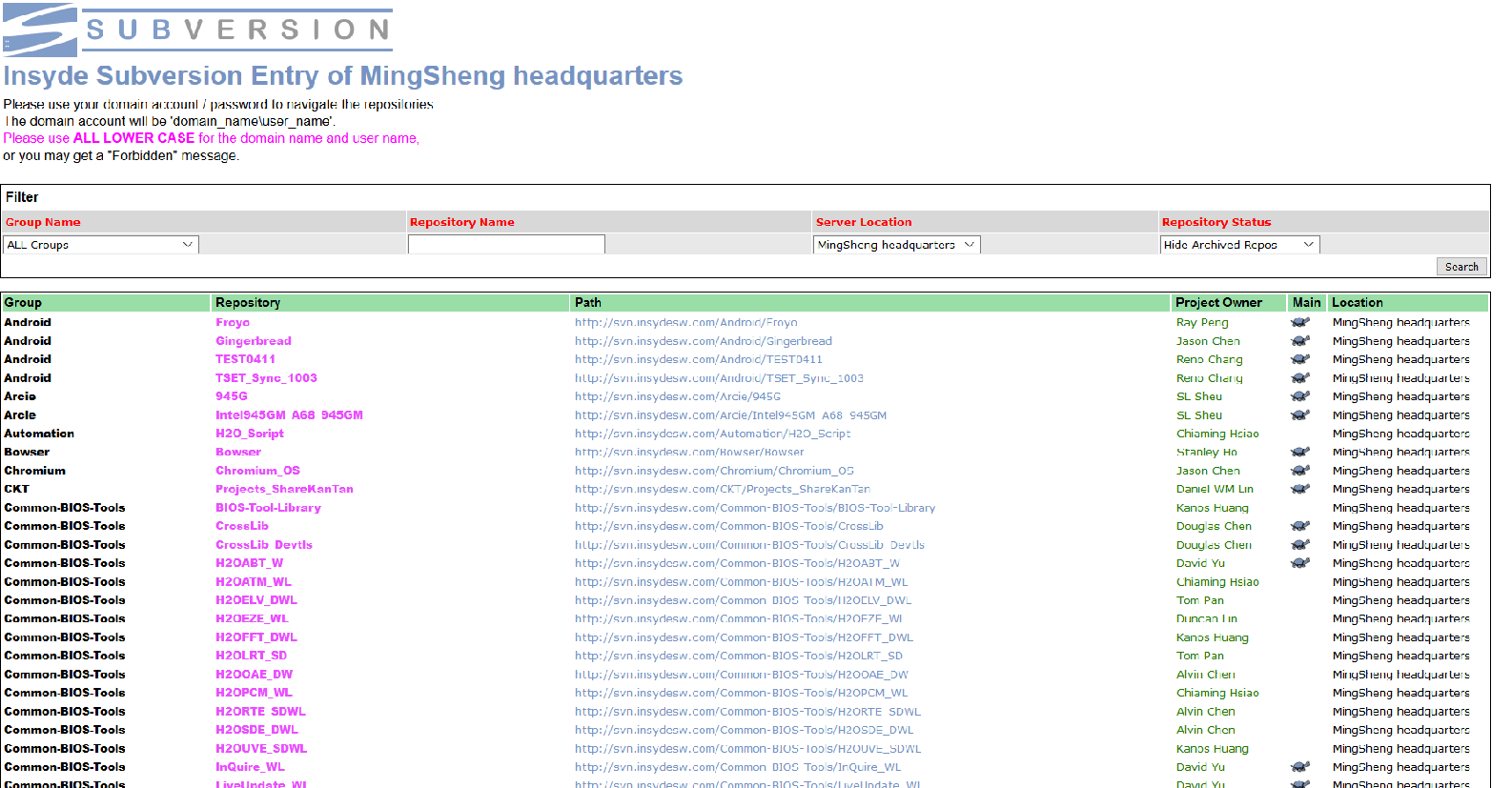
InQuire 的 User Guide 使用 Gitbook 撰寫，詳細文檔請參考 [GitBook Toolchain Documentation](https://toolchain.gitbook.com/)，爲了處理 URI 路徑問題因此對 Default Theme 做了一些修改

## Gerrit

### Web Portal

* Projects
  + [InQuire\_WL / flask-git-repos](http://inquire.insyde.com:3000/InQuire_WL/flask-git-repos)

Git Web Portal 是模仿公司內部 SVN Server 的既有網站行為，使用 [Flask](http://flask.pocoo.org/) 所打造的入口網站。公司所導入的 Git Server 使用的是 [Gerrit Code Review](https://www.gerritcodereview.com/)，由於 Gerrit 舊版本的 UI 界面不是太親民，而新版本的 UI 還在開發階段尚未完善且不一定會導入，因此透過一外部網站使用 [Gerrit REST API](https://gerrit-review.googlesource.com/Documentation/rest-api.html) 來取得 Project 列表與相關資訊，並加上額外的 Group 與 Owner 欄位來方便分類管理

* 既有的 SVN 入口網站 

### Plugin

* Projects
  + [kevin.chen / gitiles](http://inquire.insyde.com:3000/kevin.chen/gitiles)

Gerrit 本身並沒有提供 Web UI 來顯示 Source Code，通常是透過 [GitWeb](https://review.openstack.org/Documentation/config-gitweb.html) 或是 [Gitiles](https://gerrit.googlesource.com/gitiles/) 來呈現。GitWeb 使用 Perl CGI 來實作，Gitlies 則是基於 [JGit](https://www.eclipse.org/jgit/)、[Eclipse Jetty](https://www.eclipse.org/jetty/) 與 [Closure Templates](https://developers.google.com/closure/templates/) 所撰寫的 Git Browser。目前的功能需求是將 [Google Repo Manifest](https://gerrit.googlesource.com/git-repo/+/master/docs/manifest-format.txt) 檔案轉譯為目錄結構呈現，理論上是使用 Gitiles 來實現此功能會比較容易

### Digression

最近在 GitHub 上看到一個有趣的 Project: [wmhilton/nde](https://github.com/wmhilton/nde)，透過 JavaScript 將 Code git clone 至 Client 端顯示。但對於解析 Google Reoo Manifest 這個需求還是在 Server 端處理渲染會比較好