**System Specification for OnTask Tool**

Prepared by the OnTask Development Team

July 14th, 2017

System Specification Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Name | Description |
| 0.1 – draft | 15/03/2017 | OnTask | Initial Creation |
| 1.0 | 16/05/2017 | OnTask | First draft finish |
| 1.0.1 | 05/07/2017 | OnTask | Up to date of technical requirements to deploy OnTask in an institutional setting |
| 1.0.2 | 10/07/2017 | OnTask | Documentation and functionality in the platform to integrate with ldap and active directory authentication  Other authentication integration development suggestion  New release change log |
| 1.0.3 | 14/07/2017 | OnTask | Security Enhancement and UI Changelog |

Table of Contents

[Introduction 4](#_Toc487807424)

[Readership 4](#_Toc487807425)

[Document Overview 4](#_Toc487807426)

[Project Repository 5](#_Toc487807427)

[Repository Organization 5](#_Toc487807428)

[Downloading from Bitbucket 5](#_Toc487807429)

[Prerequisite and dependencies 6](#_Toc487807430)

[Prerequisite Softwares 6](#_Toc487807431)

[Dependencies 6](#_Toc487807432)

[Deployment 7](#_Toc487807433)

[Database 7](#_Toc487807434)

[SimpleQueue settings 7](#_Toc487807435)

[Token-server settings 7](#_Toc487807436)

[OnTask Web Server settings 7](#_Toc487807437)

[Ldap Settings 8](#_Toc487807438)

[Active directory Settings 8](#_Toc487807439)

[Authentication Integration Development 9](#_Toc487807440)

[Installing and Running 10](#_Toc487807441)

[Change Log 11](#_Toc487807442)

[Server side queue style 11](#_Toc487807443)

[Fragmented upload 11](#_Toc487807444)

[User authentication 11](#_Toc487807445)

[Security 11](#_Toc487807446)

[Interface change 11](#_Toc487807447)

[Security Enhancement 12](#_Toc487807448)

[Browser 12](#_Toc487807449)

[Web Server 12](#_Toc487807450)

[Database 12](#_Toc487807451)

[Communication 12](#_Toc487807452)

[API (Similar to Web Server) 12](#_Toc487807453)

[User Interface (UI) Changelog 13](#_Toc487807454)

[Home landing screen & overview screen 13](#_Toc487807455)

[Dada screen 13](#_Toc487807456)

[Matrix screen 13](#_Toc487807457)

[Rules screen 13](#_Toc487807458)

[Summary screen 13](#_Toc487807459)

[Legal 14](#_Toc487807460)

# Introduction

This document describes how to config and deploy the OnTask Tool. All the information that is essential to the config and deployment process is unfolded: how to config and verify the various deliverables, library dependencies, how to install and run the software, and the system requirements. In addition to that, we collected some other useful information related to the deployment process such as a description of the project repository.

## Readership

The primary audience of this document is system administrators and power-users that wish to build the software (or any other available project artifact, like documentation) from scratch. We assume the reader has some familiarity with the Node.js and a minimum knowledge of database.

## Document Overview

The rest of this document is organized in the following sections:

* **Project Repository**: Contains useful information on how the repository is organized and how to download it.
* **Prerequisite and dependencies**: Details on the various prerequisite software and dependencies the tool needs.
* **Deployment**: Details on all servers’ settings.
* **Installing and running:** How to install all package and start running the servers.
* **Change log:** The changes that made in the latest release.
* **Security Enhancement:** Security enhancement procedures that integrated in the system and need to be done by the server side security measurement.
* **Legal**: Licenses and copyright information.

# Project Repository

The OnTask source code and related artifacts can be found on bitbucket.org. It is only accessible by invite bitbucket user into the developers and administrators group.

## Repository Organization

The repository contains several directories as follows:

* SimpleQueue: This directory contains a simple job queue used for sending email.
* Token-server: This directory contains token service to assign and verify token that used for secure client – server interaction.
* Web: This directory contains main structure of OnTask tool. It used sails.js as framework.

## Downloading from Bitbucket

In order to download the repository, user need to provide their bitbucket account and get invited into groups and assign read access. Then they can use https or ssh to clone the repository.

# Prerequisite and dependencies

The build process is designed to be cross-platform, it is based on node.js which is very easy to handle library dependencies. In fact, all the libraries required by the application can simply installed using npm as the package manager. The remainder of this section details the various build targets and provides useful information related to the build process.

## Prerequisite Softwares

* Redis v2.8 or later (recommend using latest version)
* Node.js v6.10 or later (recommend using latest version)
* Mysql or MariaDB (recommend using latest version)
* Python v2.7
* Ruby v2.2.6 or later (recommend using latest version)

## Dependencies

All server side dependencies are managed by npm and all client side dependencies are managed by bower. There are inside both the /simplequeue and /web directory named `package.json`. Insider this file, all server side dependencies are listed with version number. Inside /web directory there is another file named `bower.json`, which contains all client side dependencies. All these dependencies can be installed through npm and bower, so no individually install required.

# Deployment

Install and running Ontask tool has a number of steps: Clone the repository from bitbucket, and put somewhere on your filesystem, set all the configs and install all dependencies then run. This section details all the configurations, installing and running procedures.

After install all prerequisite softwares (python need to be added into system path when using windows). There are several pre-settings need to be done:

## Database

Pre-create two empty databases in mysql. The sails will automatically create table inside the OnTask database first time started. (Remember to change /web/config/models.js, `migrate` attribute value to ‘safe’ after tables been created. The ‘alter’ value is used for auto-creation of the tables, but will have potential risk in losing data).

For mysql version > 5.5, please add following settings into my.cnf file:

[mysqld]

Innodb\_strict\_mode = 0

Innodb\_file\_per\_table

Innodb\_file\_format = Barracuda

Innodb\_file\_format\_max = Barracuda

## SimpleQueue settings

* inside directory /simpleQueue/config, set the settings.json with the proper redis and email smtp server info.

## Token-server settings

* Go to /token-server/key, open “key generation command.txt” file, generate ssh key based on the command line provided or the document suggested.
* Put generated .key file and .pub file in /token-server/key folder
* Open /token-server/index.js file, change `privateKey` and `publicKey` used ssh file name based on generated ssh file, also change the config settings like port/expires\_time by needs.

## OnTask Web Server settings

* The two pre-created database, one is for all the external data (e.g uploaded csv file). The other one is the ontask system database. Go to /web/config/constant.js file, change `serverInfo` object’s `adminDB` attribute to external database’s name. Go to /web/config/connections.js, change `MysqlServer` object’s `database` attribute to ontask system database’s name, then set all the other attributes required to connect to mysql (e.g. port, password, host, user).
* Go to /web/config/connections.js file, change object ‘MysqlServer’ with host, user, password and port property based on your mysql settings.
* Go to /web/config/constant.js file, change `email` object’s attributes to your own email smtp server settings.
* Default port for ontask server is 1337, you can set port by change /web/config/local.js
* If the server need to be running on https, go to /web/config/local.js file, add bundle, key and cert file path in ssl object
* Go to /web/config/constant.js, change ‘serverInfo’ object, serverinfo.config has same property value as /web/config/connections.js `MysqlServer` object.
* Change /web/config/session.js host, port using your redis server host and port.
* Go to /web/config/constant.js file, change ‘authType’ to ‘ldap’ or ‘activeDirectory’ based on authentication needs.

## Ldap Settings

Go to /web/config/constant.js file. In it find `ldapSettings` part, there are several attributes:

* **url**: The ldap server url
* **binDN**: server account dn
* **password**: server account password
* **filter**: filter for the search of target groups
* **scope**: scope of group search
* **attributes**: returned result fields of the group search
* **partition**: partition for the group search
* **firstName**: attribute name which contains user’s firstName (can be empty)
* **lastName**: attribute name which contains user’s lastName (can be empty)
* **fullName**: attribute name which contains user’s fullName (can be empty)
* **userDN**: can add any userDN structure in the function, each type of userDN can be constructed by username and dn. High priority userDN will push into the array first.
* **memberOfAttr**: used for search which group this user belongs to, need to fill in with the users’ memberOf field name (usually default as ‘memberOf’).
* **memberAttr**: used for search all the members in this group, need to fill in with the groups’ member field name (usually default as ‘member’)
* **admin**: admin group’s dn (can be multiple groups, add as an array)
* **staff**: staff group’s dn (can be multiple groups, add as an array)
* **student**: student group’s dn (can be multiple groups, add as an array)

## Active directory Settings

Go to /web/config/constant.js file. In it find `activeDirectory` part, there are several attributes:

* **url**: The active directory server url
* **bindDN**: server account dn
* **baseDN**: base dn to start search on
* **password**: server account password
* **firstName**: attribute name which contains user’s firstName (can be empty)
* **lastName**: attribute name which contains user’s lastName (can be empty)
* **fullName**: attribute name which contains user’s fullName (can be empty)
* **roleAttribute**: if this attribute has value, then the admin, staff and student attributes below will be ignored. This attribute contains the attribute name of the group from user’s active directory search result. E.g. search user from active directory returns result with attribute ‘description’, the this attribute is set to roleAttribute: ‘description’.
* **roleAttributeValue**: the value of roleAttribute, need to map admin/staff/student with the roleAttribute value. E.g. ‘admin: Admin’, ‘staff: Staff’, ‘student: Student’
* **userDN**: can add any userDN structure in the function, each type of userDN can be constructed by username and dn. High priority userDN will push into the array first.
* **admin**: function used to create admin group list array
* **staff**: function used to create staff group list array
* **student**: function used to create student group list array

When all settings are correct, users can use their username and password in the login page, the ldap/active directory service will return which group (admin/staff/student) the user belongs to, then authorize different level of access based on the group info.

## Authentication Integration Development

There are other authentication services that has not been integrated into the system. In order to work with another authentication service other than ldap and active directory, following steps can be used to develop new service:

1. Find a node module that has API to authenticate. E.g. oauth package from npm.
2. Go to /web/api/services/ADService.js file, under ‘module.exports’ object, add another service function, e.g. ‘oauth: function () {}’.
3. Go to /web/config/constant.js file, change ‘authType’ to the service function name. In this example, ‘oauth’. Add another object in the constant.js file called ‘oauth’ with all the required settings in it. Then ADService.js file can use the object as ‘sails.config.constant.oauth’.
4. Write the function to connect to the authentication API, can return authentication results.
5. This function is supposed to get authentication info and return an object with structure like following:

{admin: false,

staff: false,

student: false,

firstName: ‘’,

lastName: ‘’,

fullName: ‘’}

The admin, staff and student attributes are important, if true, it means the user has that role.

1. Restart server, the login authentication will go through oauth.

Please note that an official API will be provided to interact with external authentication system in the next release.

# Installing and Running

After all the config and settings ready. In order to install and run the server, following steps need to be done:

1. Start redis server
2. open a command terminal, go to simplequeue directory, run `npm install`, after all installation finished. Run `Node app.js`
3. open a command terminal, go to token-server directory, run `npm install`, after all installation finished. Run `Node index.js`
4. open another command terminal, go to web directory, run `npm install`, after all installation finished.
5. in command terminal at web directory, run`bower install`, after all installation finished. Run `Node app.js`
6. Access the OnTask tool on your browser by http://<domain>:<ontask-port>

# Change Log

There are several important changes in the latest release:

## Server side queue style

Most resource consumed or time consumed works and requests from the system are now been processed by queue service. E.g. upload file, process file content and send email etc. A backend queue is running with redis to provide the ability to deal with peak demands, finish work with small, self-contained programs.

## Fragmented upload

new csv upload function provides the ability to deal with large dataset. Meanwhile, error handling and exception handling during csv upload is provided to make sure better upload experience.

## User authentication

Integration with ldap and active directory are provided. Now user can enter their email and password to authenticate, and get admin/staff/student role based on the information returned from ldap/active directory service.

## Security

Security changes as refer to Security Enhancement changelog section

## Interface change

Interface changes as refer to UI changelog section

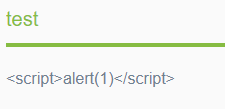
# Security Enhancement

## Browser

* **Cookie (Identity fraud)**: Secure Cookie storage and transmission by adding secured and HttpOnly flag. Cooked signed everytime user successfully login. Automatically expire obsolete token Cookie (7 days) if not re-login or logout.
* **Cross-site scripting (XSS)**: Enforce same domain policy. Inherited NodeJS server default setting, cross origin resources sharing (CORS) disabled

## Web Server

* **Denial-of-service attack (DoS)**: requires server infrastructure security measure. Set up proper configured firewall controlling incoming traffic: requires server infrastructure security measure. Set up load balancing server, if possible, and set up rules for requests from the same source.
* **Arbitrary code execution (Code injection)**: Separated form content from page DOM loading, treat form input as plain text to make sure no script or code running. Never run ad-hoc code from the browser side.



## Database

* **SQL injection**: Use ORM framework to do necessary checking. If in no ORM scenario, use Pattern Checking, Escaping, and Parameterised Statements to avoid SQL injection
* **Data breach**: Use separate accounts to perform read-only and read/write operations. Hash or encrypt sensitive data before storing into database. Ldap/active directory only store token, password not stored.

## Communication

* **Man-in-the-middle attack**: Use SSL encryption via HTTPS protocol
* **Spoofing attack**: require server side settings

## API (Similar to Web Server)

* Define role-based permission control (ACL - access control list) on different API endpoints.
* Follow W3C REST standards (<https://www.w3.org/2001/sw/wiki/REST>).
* When using a third-party service (like user authentication), use the above measures to evaluate its security threats.

# User Interface (UI) Changelog

## Home landing screen & overview screen

* OnTask version number upgrade to 1.0.0
* Start using onTask official logo.
* Re-create landing page welcome picture & make it to be uncluttered beautifully bare
* Re-create login section, fix detail bugs & add new hint functions
* Add new software license clarification on landing page
* Re-create heading navigation bar, add functional buttons for user to easy navigating from different pages.
* Fix create new workflow bugs.
* Fix workflow function search bugs.
* Align UI style with icon + description.
* Fix wording error.
* Fix loading animation bugs.

## Dada screen

* The workflow thumbnail window is disabled temporarily.
* The operation history window is dis-functional, so decided to hide it temporarily.
* Re-create user assistant system
* Fix updating long-list csv file bugs
* Change “My data” section to “Matrix” in order to align next section
* Hide dis-functional blocks.

## Matrix screen

* The workflow thumbnail window is disabled temporarily.
* The operation history window is dis-functional, so decided to hide it temporarily.
* Re-create Conversion block
* Fix data conversion bugs
* New UI style for custom attributes block
* Fix viewing table visualisation bugs

## Rules screen

* Change wording issues
* Modify UI style
* Add automated verification function, disabled “verify” button

## Summary screen

* Modify UI styles bugs
* Change wording

# Legal

To be included