d

TOne Security Document

Contents

[Document Purpose 2](#_Toc421264385)

[Security Design 3](#_Toc421264386)

[Business Entities Structure 3](#_Toc421264387)

[Permissions 4](#_Toc421264388)

[Groups 4](#_Toc421264389)

[Group Members 4](#_Toc421264390)

[Group Permissions 4](#_Toc421264391)

[Authentication 6](#_Toc421264392)

[Authorization 7](#_Toc421264393)

# Document Purpose

The purpose of this document is to discuss the security implementation of TOne. This is composed of:

* Authentication
* Authorization

# Security Design

The scope of this section is to discuss the design of the authorization part of the security. The authorization consists of allowing an authenticated user to ONLY do what he is allowed to do while using TOne UI (or API). It is summarized in the following statements:

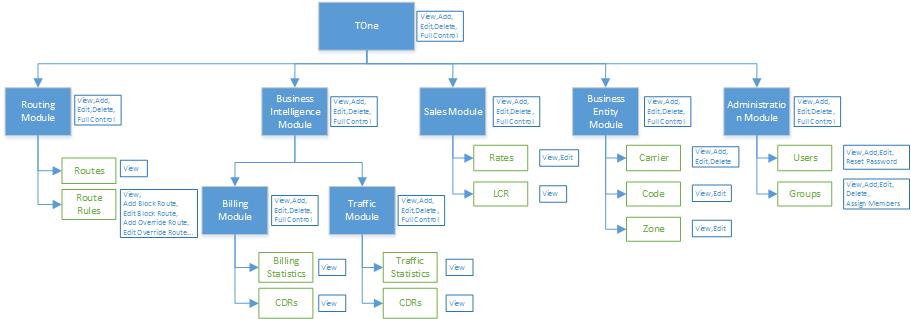
* An authenticated user can access ONLY TOne pages that he/she has permissions to view
* An authenticated user can see ONLY data that he/she has permissions to view
* An authenticated user can perform ONLY actions that he/she has permissions to perform

In order to implement the above requirements in a flexible manner, we would:

* Define a structure for the system business entities in a way that allows the system administrator(s) to configure permissions easily
* To implement user groups membership which also gives more flexibility in configuring permissions

## Business Entities Structure

TOne Business Entities would be structured in a hierarchical manner:



The above diagram shows an example business entities hierarchy. In this case, TOne would be composed of multiple modules (e.g. Routing, Business Intelligence…). Each module would also have multiple modules and/or entities (e.g. Routes, Route Rules, Rates…).

Each entity in the hierarchy would defines the types of actions it has. For example:

* The Route Rules entity defines the following actions:
  + View
  + Add Block Route
  + Edit Block Route
  + Add Override Route
  + Edit Override Route
  + …
* The Users entity defines the following actions:
  + View
  + Add
  + Edit
  + Reset Password

And each module in the hierarchy defines the following actions:

* View
* Add
* Edit
* Delete
* Full Control

## Permissions

A permission is composed of four parts:

* Permission Holder: User or Group
* Secure Resource: Business Entity or Module (as discussed in previous section)
* Action: e.g. View, Add, Edit…
* Permission Flag: Allow or Deny

The following rules would be also applied:

* The business entities inherit the permissions assigned on their parent modules in the Business Entities hierarchy
* It would be possible to stop Permission Inheritance on any Module/Entity in the hierarchy
* The Deny Permission Flag overrides any Allow Permission Flag

## Groups

As discussed earlier, the system will give ability to create user Groups to make security configuration easier. The Group is composed of two parts:

* Members: this part would give a way or more to tell which users are members of this Group
* Permission: this part would give a way or more to tell what permissions this Group has and to which business entities

### Group Members

Group Members would be assigned in two ways:

* Explicitly: the administrator sets the users that are the members of the group
* Implicitly: the members of the group would be defined implicitly this way:
  + Each user would have a category attribute (e.g. Financial, Technical, Administration…)
  + The administrator would then configure the group to include as members the users that have specific category(ies)

### Group Permissions

Group Permissions would be also assigned in two ways:

* Explicitly: the administrator sets the permissions of the group directly on the Business Entity hierarchy
* Implicitly: the permissions of the group would be defined implicitly this way:
  + Each Module/Entity would have a category and sub-category attributes (e.g. Financial, Technical, Billing, Traffic…)
  + The administrator would then configure the permissions of the group by assigning permissions on the modules/entities that have specific category(ies) and sub-category(ies)

# Authentication

# Authorization