Cello How-To Guide

Package Management

techcello

Contents

1 Package Management 4
1.1 How to implement packages in cello 4
1.2 Cello Package Structure 4
1.3 How to Configure a Module 5
1.4 How to Configure a Feature 6
1.5 Usages 6
1.6 How to assign usages? 6
2 Price Plans 7
3 Introduction to Self Registration 8
3.1 Tenant Activation Process 9
3.2 How to Enable/Disable Account activation via Email 10
4 Configuring selective subscriptions plans for Self Registration 10
4.1 How to configure Package for Self Registration 11
5 Package Administration 13
5.1 Introduction to Subscriptions 14
5.2 Types of Subscriptions 15
5.3 How to Create Subscriptions 15
5.4 How to create Evaluation Subscription? 18
5.5 How to Create Trial Subscription? 18
5.5.1 Login after Trial Expires 19
5.6 Types of Trial Package 19
5.7 Set Usage 20
5.8 Edit Package 20

5.9 Subscription Self Upgrades/Downgrades 23

5.9.1 Upgrade/Downgrade Subscription plan of Tenant by Prod Admin 25

Subscription Changes (Prorated) 27

Self Upgrade 27

6 Tenant's Subscription History 29

7 How to register application specific Modules 30

8 Contact Information 34

1 Package Management

Package management systems provide Product owners with the necessary tools to associate software service with pricing plans and collect revenue. CelloSaaS subscription management seamlessly integrate with provisioning, metering and payment systems. The benefit of tight systems integration is the ability to deliver subscription pricing to tenants based on a customized editions of software. CelloSaaS also provides automation of trial-to-paid conversion, the ability to define service bundles based on modules, features and the ability to define license models and usage policies for SaaS offerings.

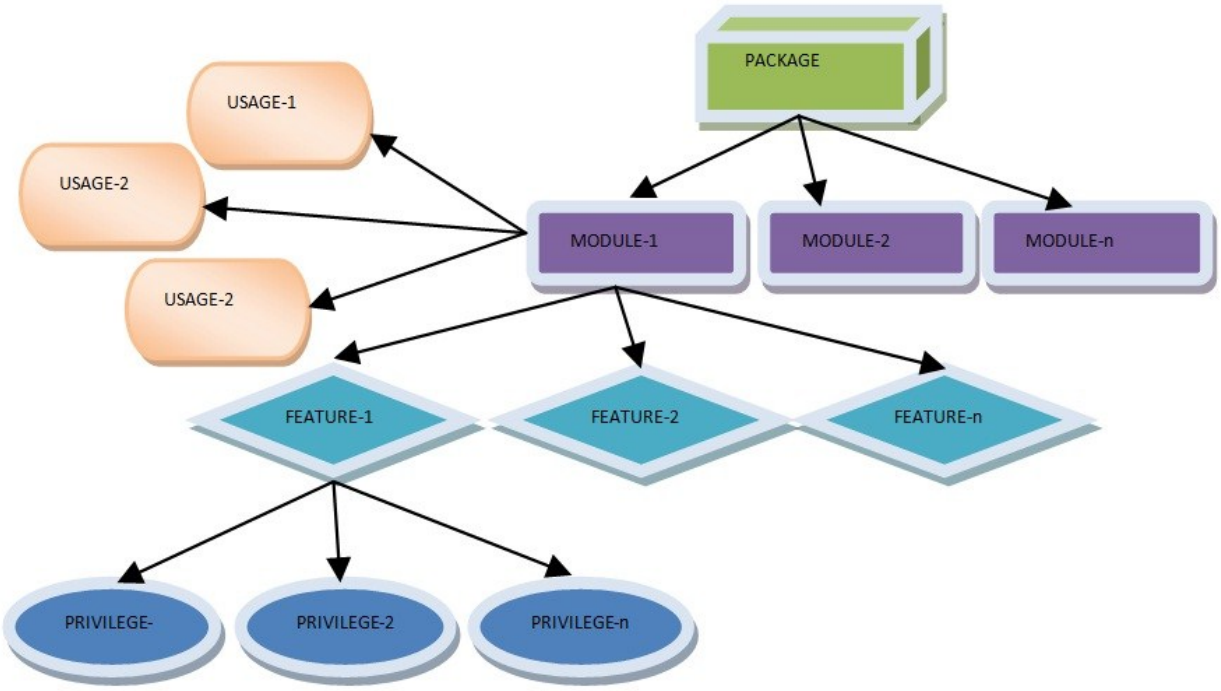
1.1 How to implement packages in cello

- In general SaaS application would bundle the modules, features and usages in various packages and have the tenants subscribe for the package. For example, Standard edition, Premium edition etc.
- Cello provides subscription management capabilities that let ISVs to configure multiple editions of the product using the dashboard with fine granularity. Developers can extend these features as per the product requirements.

1.2 Cello Package Structure

- Package is a collection of Modules, Features and Usages, decided by the ISV (SaaS application owner).
- Module is a logical grouping of set of features. For example, “Employee Management” is a module which consists of features like the ones given below.
- Feature is a logical representation of any functionality. For example, “Employee Profile Management”, “Organization Hierarchy Management” is features.
- Usages represent any metric based restrictions that we want in the system. For example, “Number of customers created” in a CRM application can be a usage based restriction.
- Each tenant will subscribe to a package, based on their need.
- Product admin/Tenant Admin will select the package for tenant based on what they intend to purchase.

Note: A Tenant can subscribe only to a single package in a given point of time.



1.3 How to Configure a Module

A module is a logical Grouping of one or more features in the application. Click here to know how to build a module. Once a module is built, use Fluent API to register Module in the system. This integration process is important for the framework to know the custom modules built by the developers. When the modules are registered, it will be automatically displayed in the Add New subscription screen. To read more about how to register Custom Modules, click here.

Example:

In a typical CRM Application, we can see some of the modules like Help Desk, Knowledge Management, Customer Management System etc, and these modules will have features inside.

1.4 How to Configure a Feature

A feature within a module is a logical representation of a particular functionality. Click here to know how to build a feature. Once a feature is built, use Fluent API to register feature in the system. This registration or configuration is important for the framework to know the custom features built by the developers. To read more about how to register Custom features, click here.

1.5 Usages

Usages or transactions are the basis of billing. When the customer buys a subscription of a product and starts using it, the usage auditing service starts tracking the actual usage. This usage audits serves as the input for calculating the invoice for the customers.

Cello billing service checks the billing frequency of each subscriber and raise the billing according to the billing plan and selected frequency

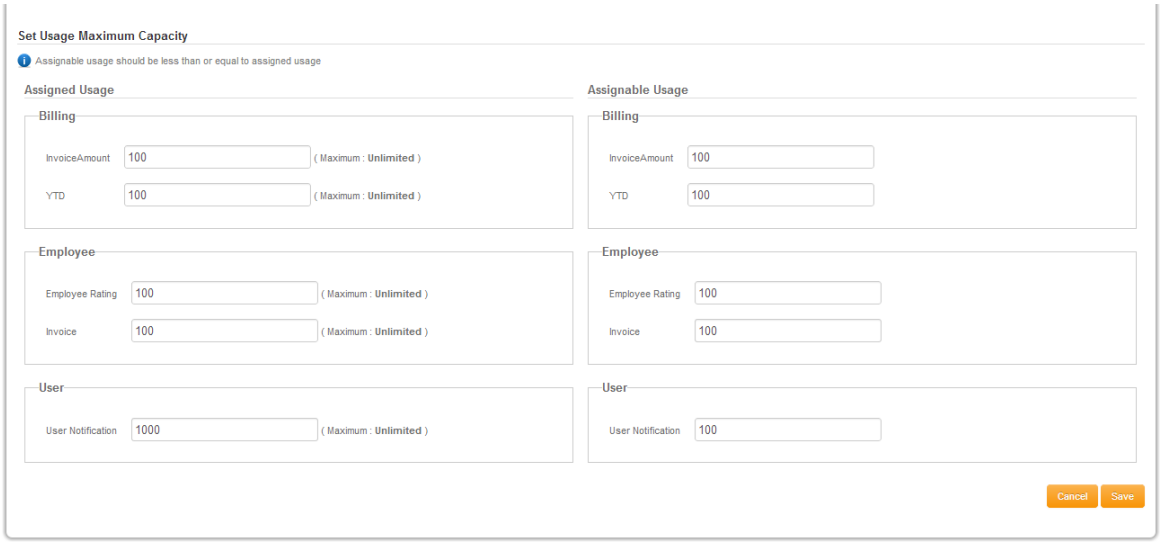
1.6 How to assign usages?

While creating a subscription plan, the product administrator could either set threshold [Number of Allowed Transactions] against each of the metering variables else set to "0" that is unlimited usage. Customers who are subscribed to the packages which has threshold will be metered against each and every transactions and when the customer reach the set threshold, the cello framework will not allow the customer to do any more transaction within the application.

In case if the customer wants to use the application any further, either the customer has to wait till the threshold resets at the end or beginning for the billing cycle ie month, or by upgrading to an higher which might allow more transactions than the current subscription.

1. Navigate to Admin→ Manage Subscriptions→ Click on Add

2. Scroll down to the bottom of the page and look for “Set usage Maximum Capacity”.



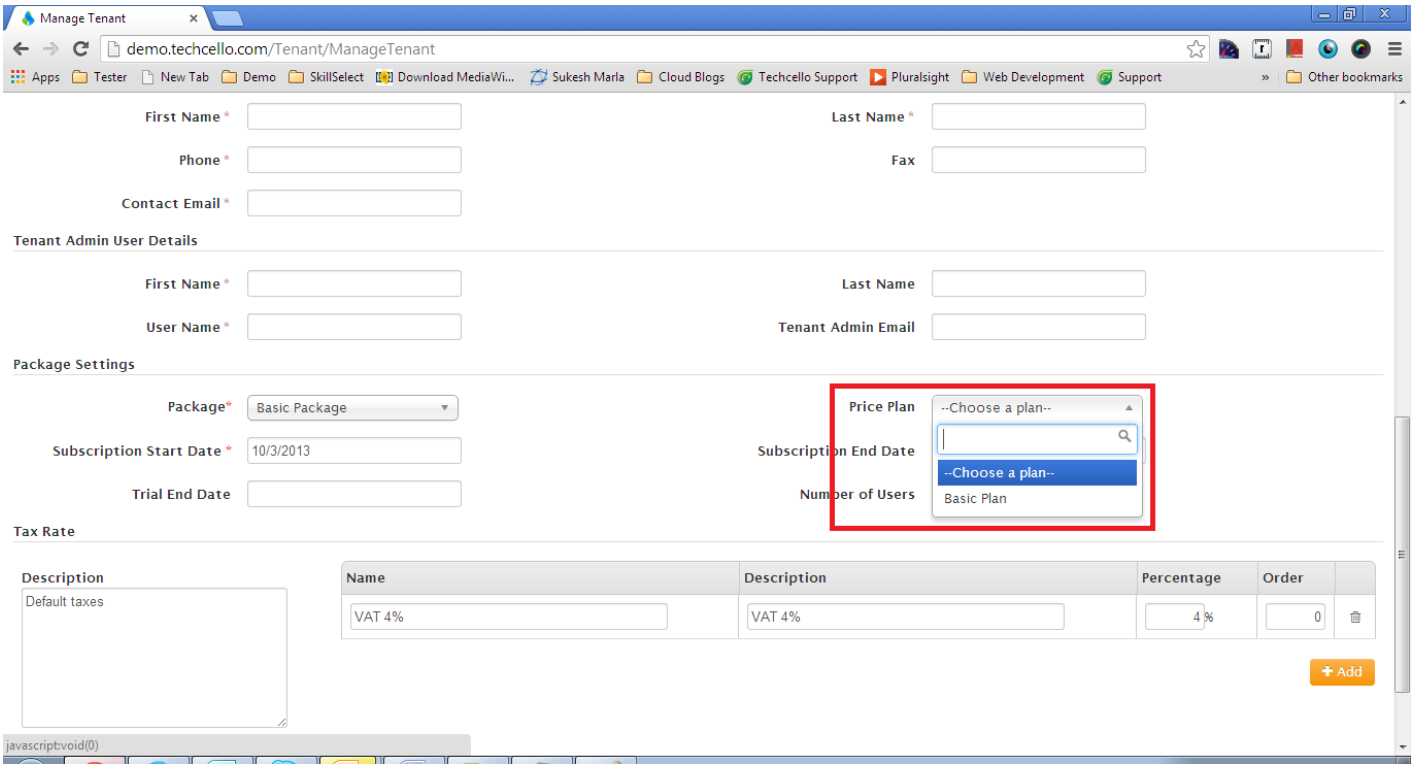
Create Subscription Page

Refer “Metering the Usage” topic in Billing Document.

2 Price Plans

Whatever may be your Pricing Strategy, CelloSaaS offers a configurable and customizable price plan module that suits 99.99 of the products which are built on SaaS model. It allows ISVs to configure variety of Price Plans in the application using the dashboard.

Once the price plans are configured, it will be listed on the “Add/Edit Subscription Page” as shown in the screenshot below.



To know more about "How to Configure Price Plans", Click here.

3 Introduction to Self Registration

Self-registration allows users to setup the application by themselves without any assistance from the product owners and create their own login and password in the application by filling basic information's and with or without Credit Card Information. A Customer who self registers is immediately and automatically granted access to the application and a new user account is created in the application's identity store.

The Application administrator has options to either Turn On or Off Self Registration Feature in the application. When it is Turned On, anyone is allowed to self-register, that is any public user, a Register link or button displays on the home page. To enable this feature refer "Enabling Self Registration".

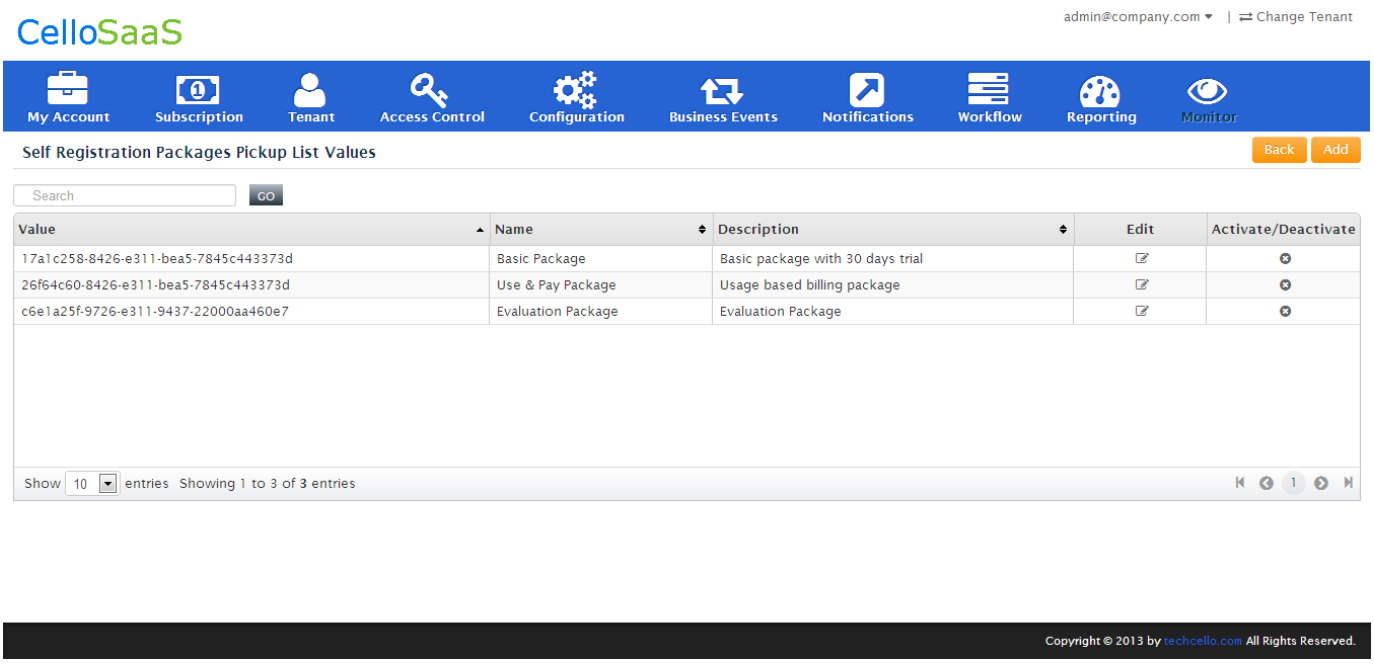
Note: Although Subscriptions can contain multiple Price Plans, the chosen subscription plans for self registration should only contain one price plan.

Scenario1:	**Scenario2:**
Plan Name: Enterprise Plan – SR(Self	**Plan Name:** Enterprise Plan

Registration) Is marked for self registration: Yes Number of Price Plans: 1	Is marked for self registration: No Number of Price Plans: 5

In Scenario 1, the plan named “Enterprise Plan - SR” has been chosen for Self Registration, so it has been mapped only with one plan, whereas in Scenario 2, the plan named “Enterprise Plan” is not marked for Self Registration and it has been mapped with 5 different Price Plans. Refer – http://www.techcello.com/downloads/kb/4.3.1/how_to_price_plans.pdf

Select the “Self Registration Packages” and click on Manage Values. Now add selected packages information as the pickup list values.



Caution: The packages configured in the pickup list will be only available for self registration.

3.1 Tenant Activation Process

After the self registration process, the Tenant administrator will receive an activation link with activation code on the email id registered with the product. The user has to activate and proceed further. After successful activation, the user can log in to the product and use all the modules and features which come with the subscription.

The activation code is valid for a fixed number of days, beyond the particular period the activation code would become invalid, so the user is expected to activate the account within the defined number of days. The administrator can decide and configure the number of days within which the account has to be activated.

In case, if the product administrator decides not to have account activation via Email and instead allow instant activation, the product owner can choose to disable activation via email Refer: How to disable Email mode account Activation below.

In case of Instant Activation, as soon as the user registers for the product, the account will be activated for them and the user can start using the product immediately.

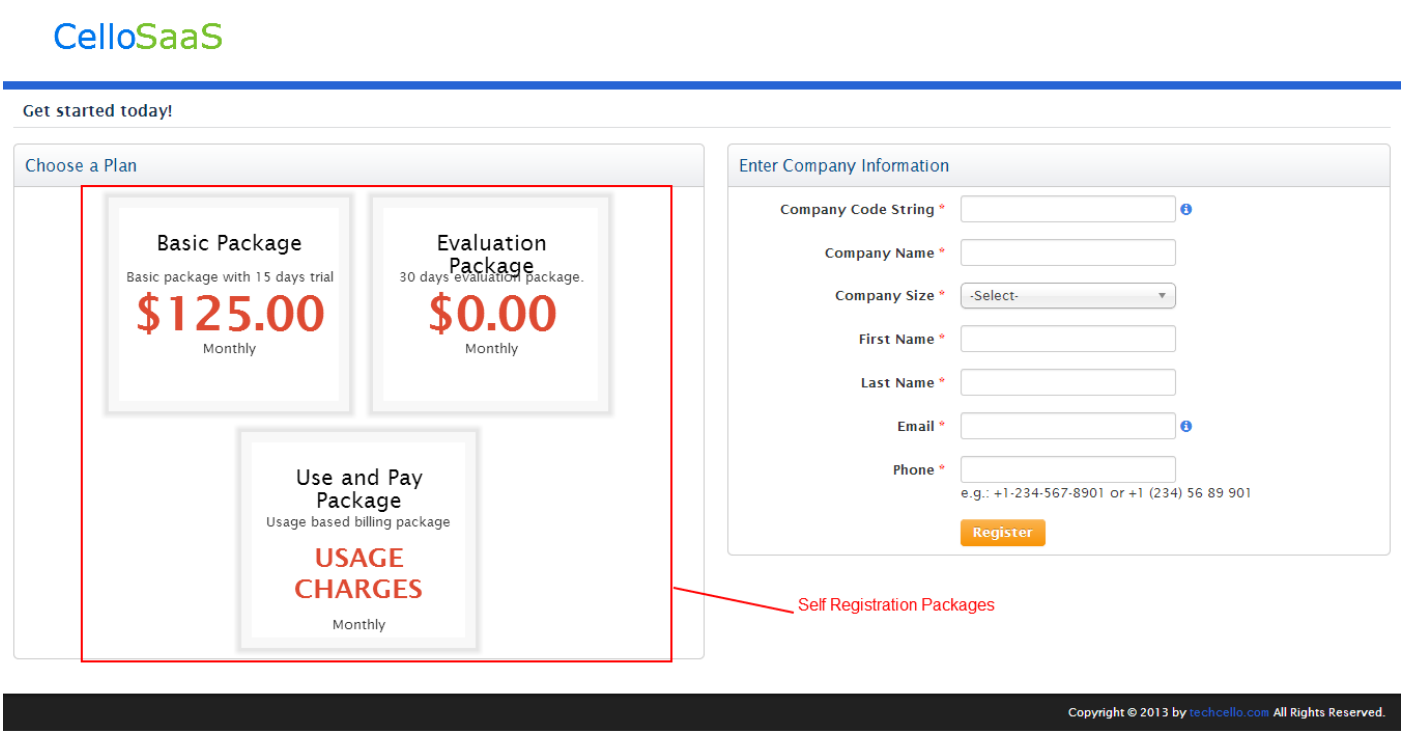
3.2 How to Enable/Disable Account activation via Email

To enable/disable account activation using email notification, change the settings in ***web.config*** as shown in the image below.

```
<!--To send the activation mail to newly created tenant -->
<add key="SendTenantActivationLink" value="true"/>
<!-- Set expiration day -->
<add key="TenantActivationExpirationDays" value="10"/>
```

4 Configuring selective subscriptions plans for Self Registration

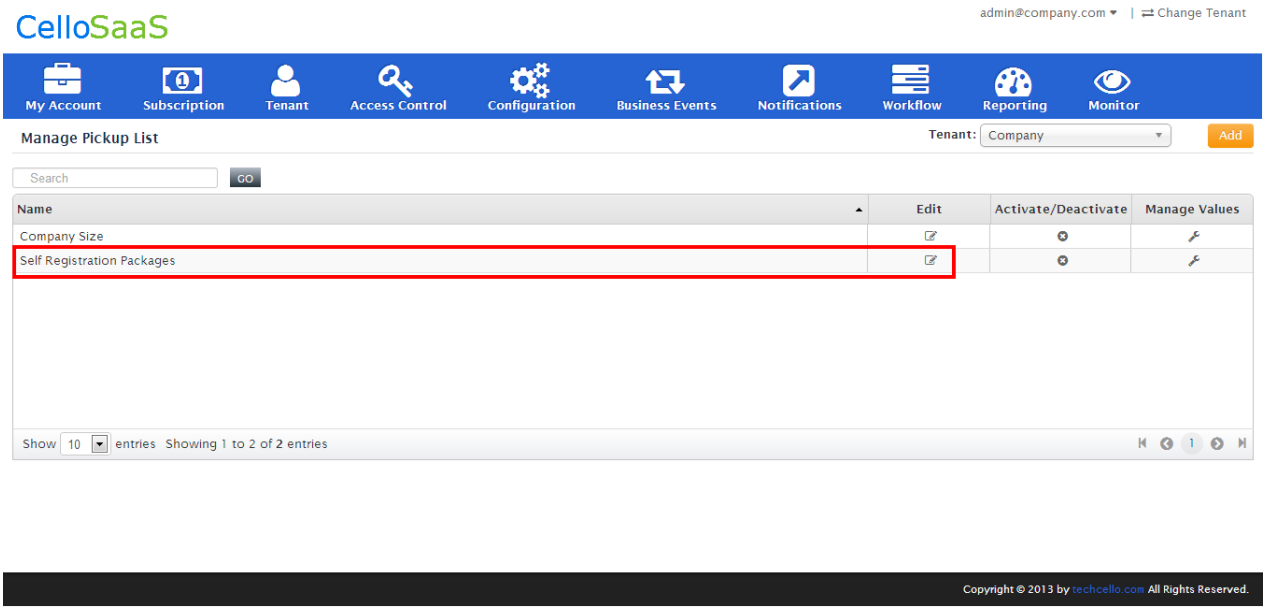
An application can contain any number of Subscription plans, but the application administrator has the facility to mark selected packages for self registration purposes. When marked for Self registration, the selective packages alone are displayed in the self registration process. Users will be able to subscribe to these subscriptions plans and use the modules and features which come with these plans.



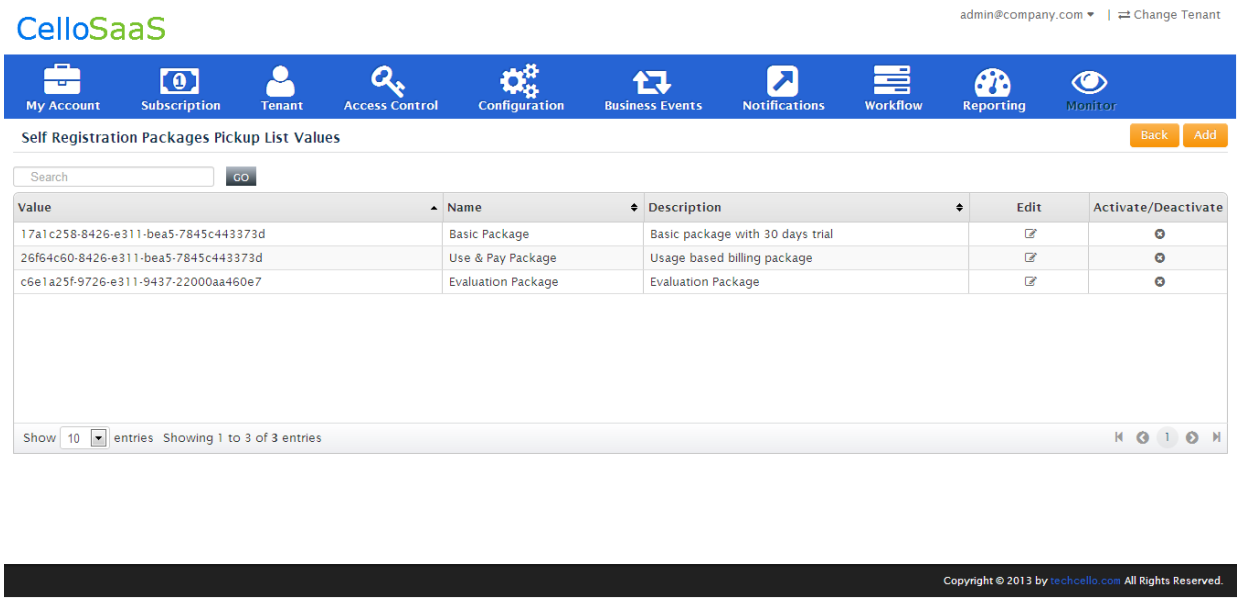
4.1 How to configure Package for Self Registration

Navigate to Admin → Configuration Management → Manage Pickup Lists

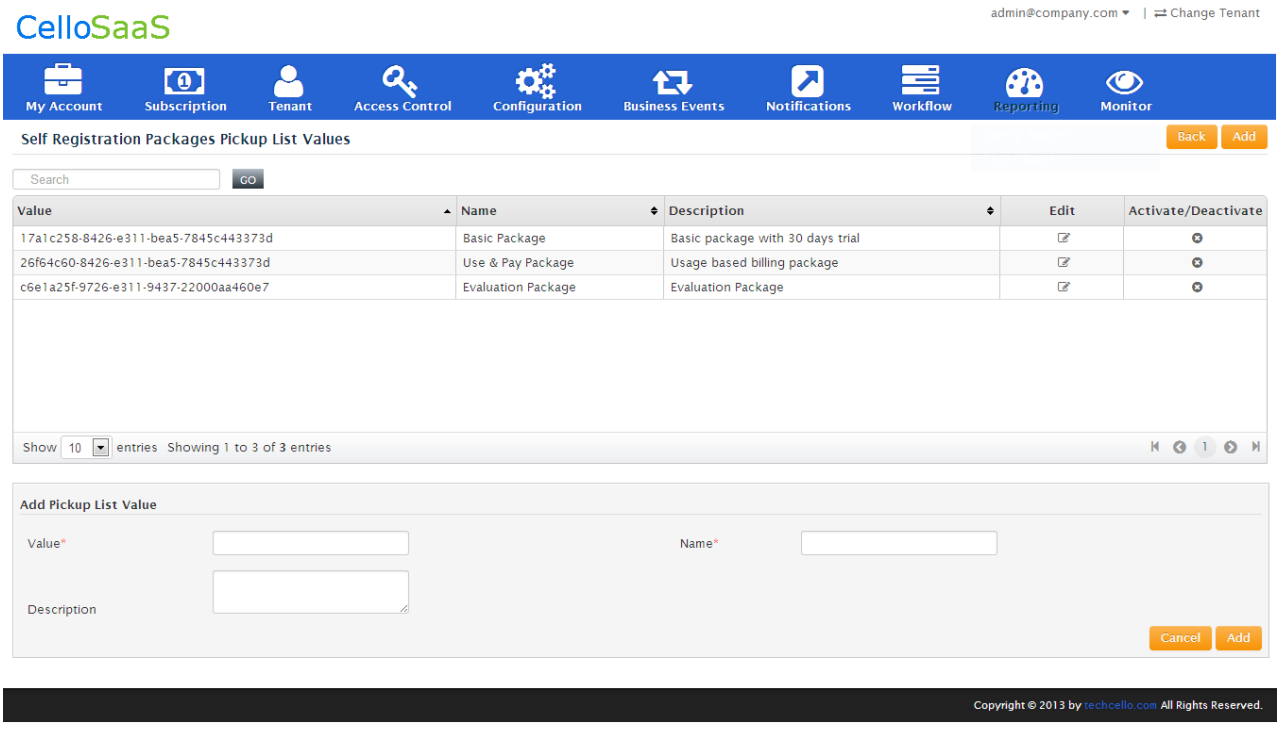
The product owner can mark selective or all packages to be subscribed in the Self Registration model.



1. Click on Manage Values



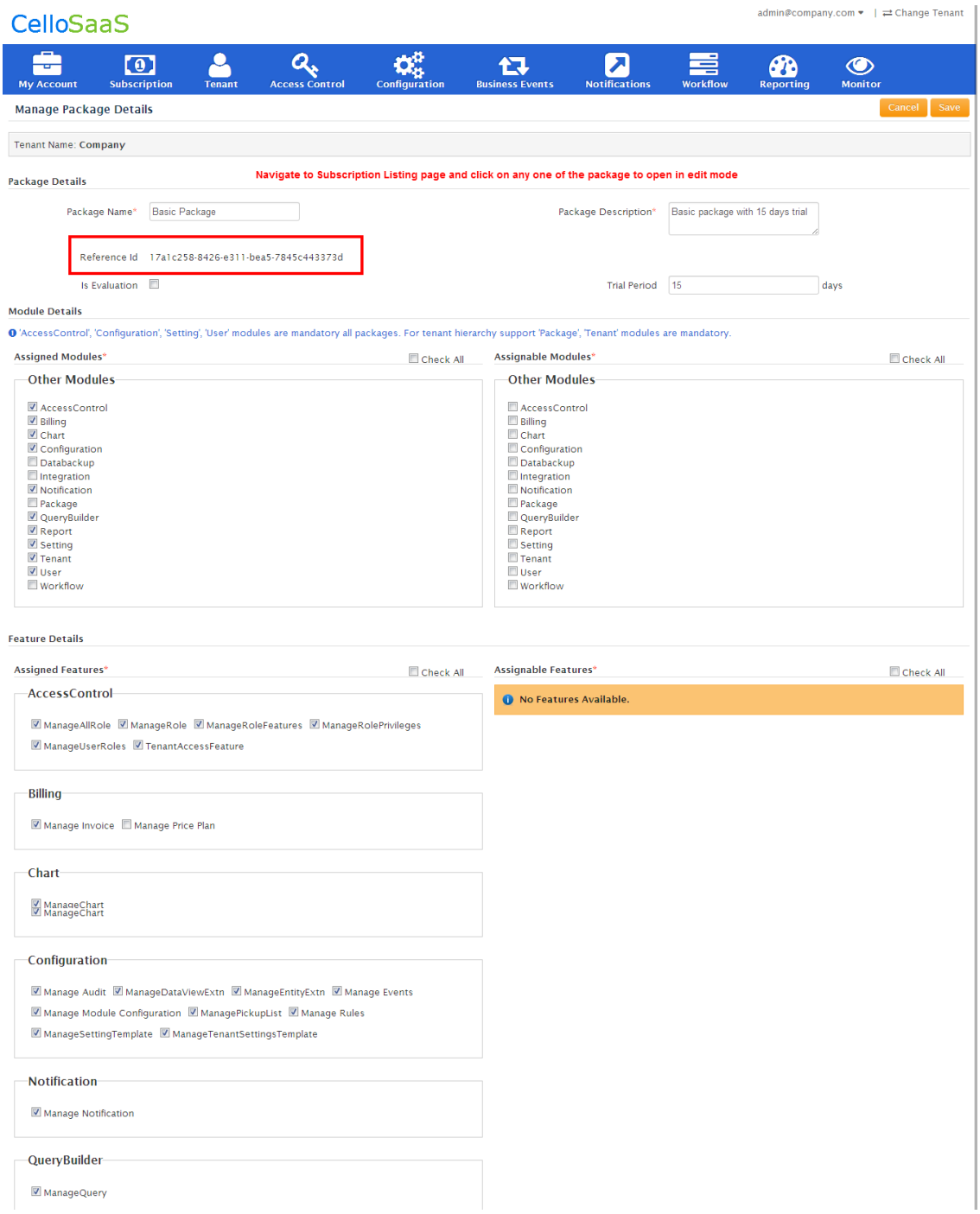
2. Click on Add



3. Add the following details

Package ID in ***Value*** Field (You can get the package ID as shown in the below screenshot)

Name of the ***Package*** and its ***Description*** and click Save

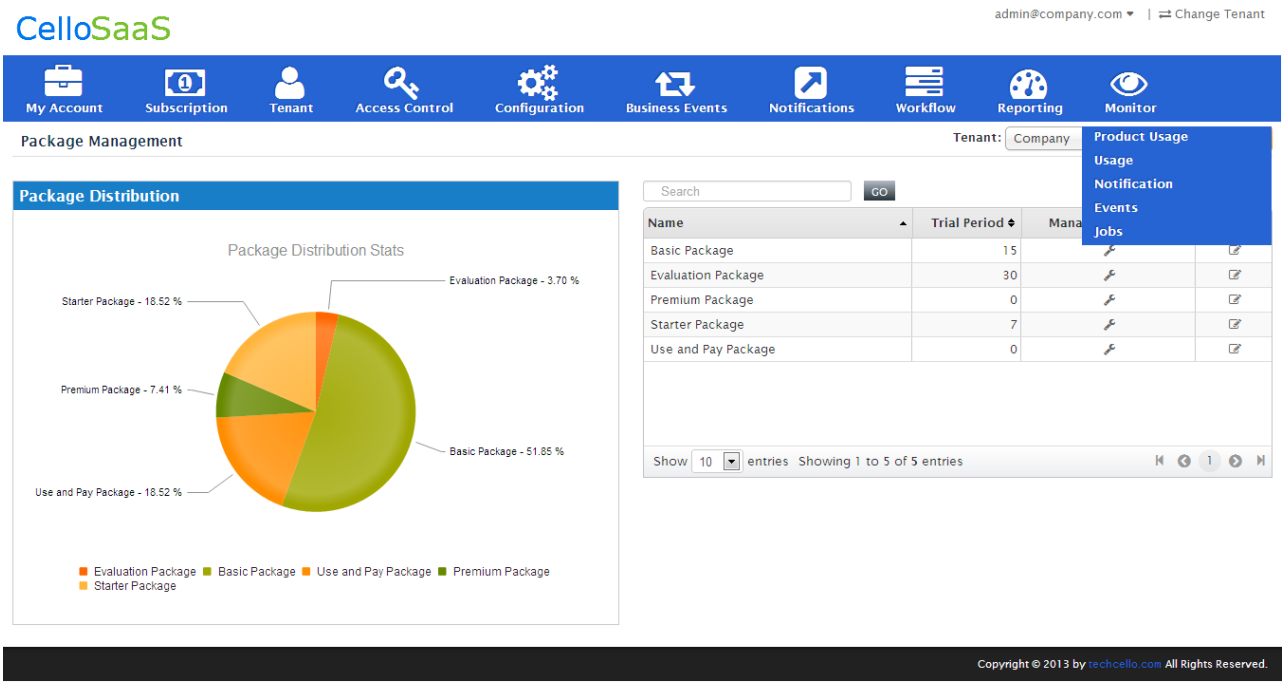


5 Package Administration

Cello comes with the inherent capability to manage packages

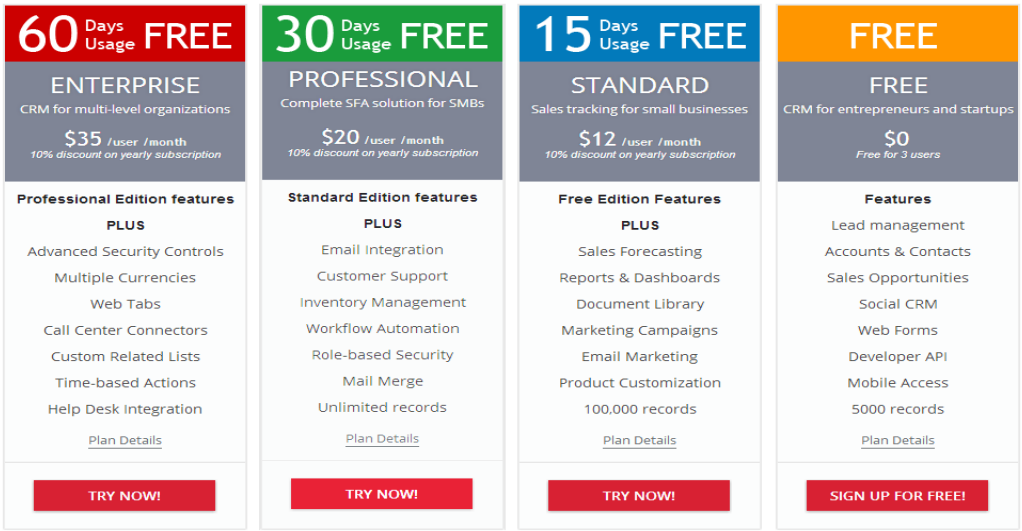
Manage Package

- By default product admin has permission to add a new package.
- Navigate to **Subscription-> Manage Subscription**



5.1 Introduction to Subscriptions

A Subscription in a SaaS Product is a basket of selective Features or all the features canned for customers to subscribe. These subscriptions can be either pre-canned based on the nature of the product or dynamically created as and when required.



5.2 Types of Subscriptions

Product Owners can create different types of Subscriptions. They are

Title	Description
Evaluation/Trial Subscription	Customers can enroll for the Product without any obligation of paying the Product. Usually Trial Subscription would either have usable Day, Feature or usage limitation.
Regular Subscription	Customers who want to use the product without any limitation can subscribe and pay for the usage.

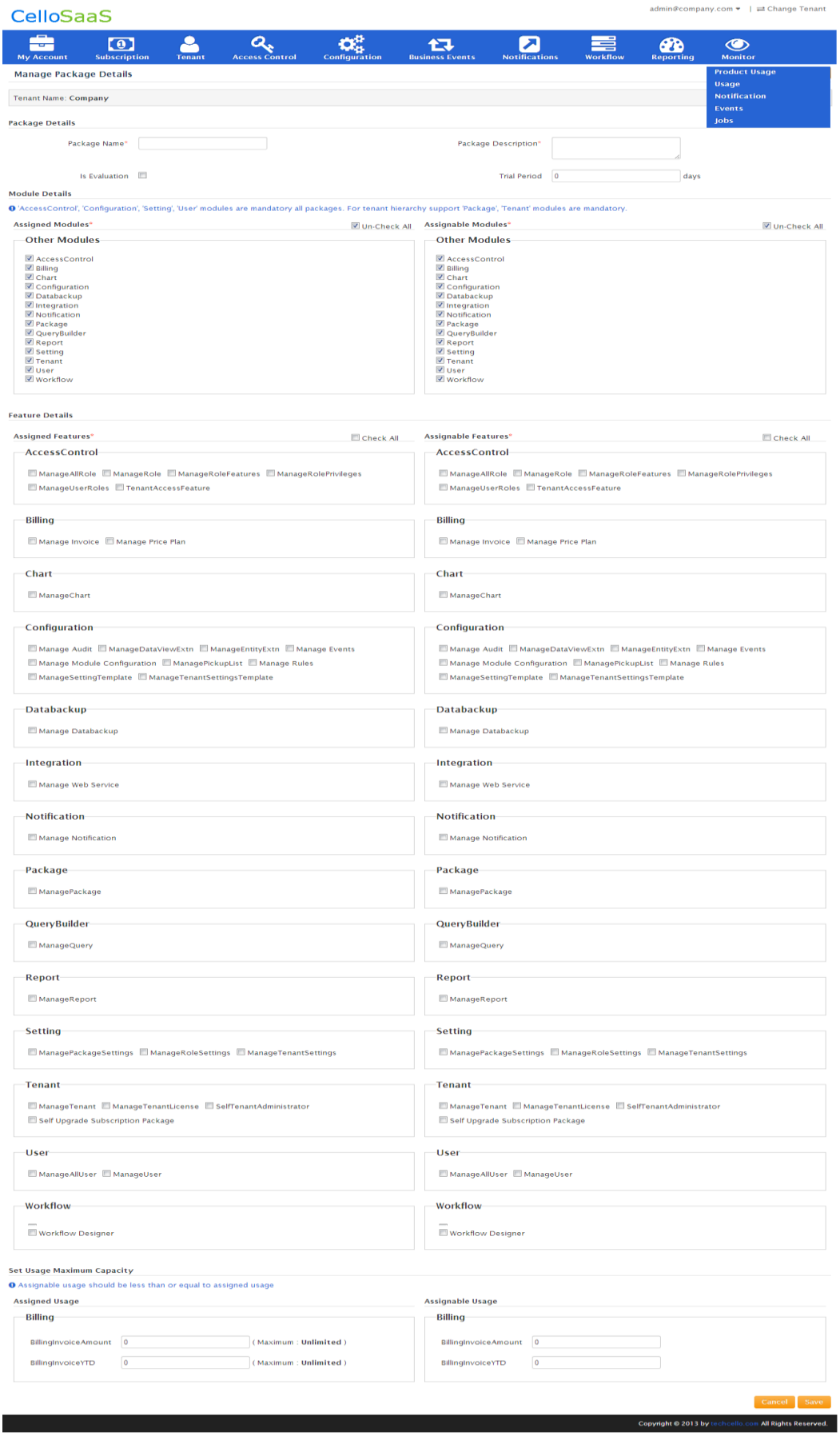
5.3 How to Create Subscriptions

- To create subscriptions, Navigate to Subscription → Manage Subscription
- Click add button in package list page to create new package.
- Give unique package name and its description.
- This page lists all the available modules, features and Usages for the corresponding modules registered with the framework. This page has two columns named as Assigned and Assignable.

  Note: By default this page lists all the modules and features of cello. In order to add application specific modules and features refer to section "**How to add application specific Modules**"

Note: Check the "Integrate for Billing" checkbox to use Cello's Integrated Billing Provider, uncheck the same to avoid Cello' Integrated Billing Provider.

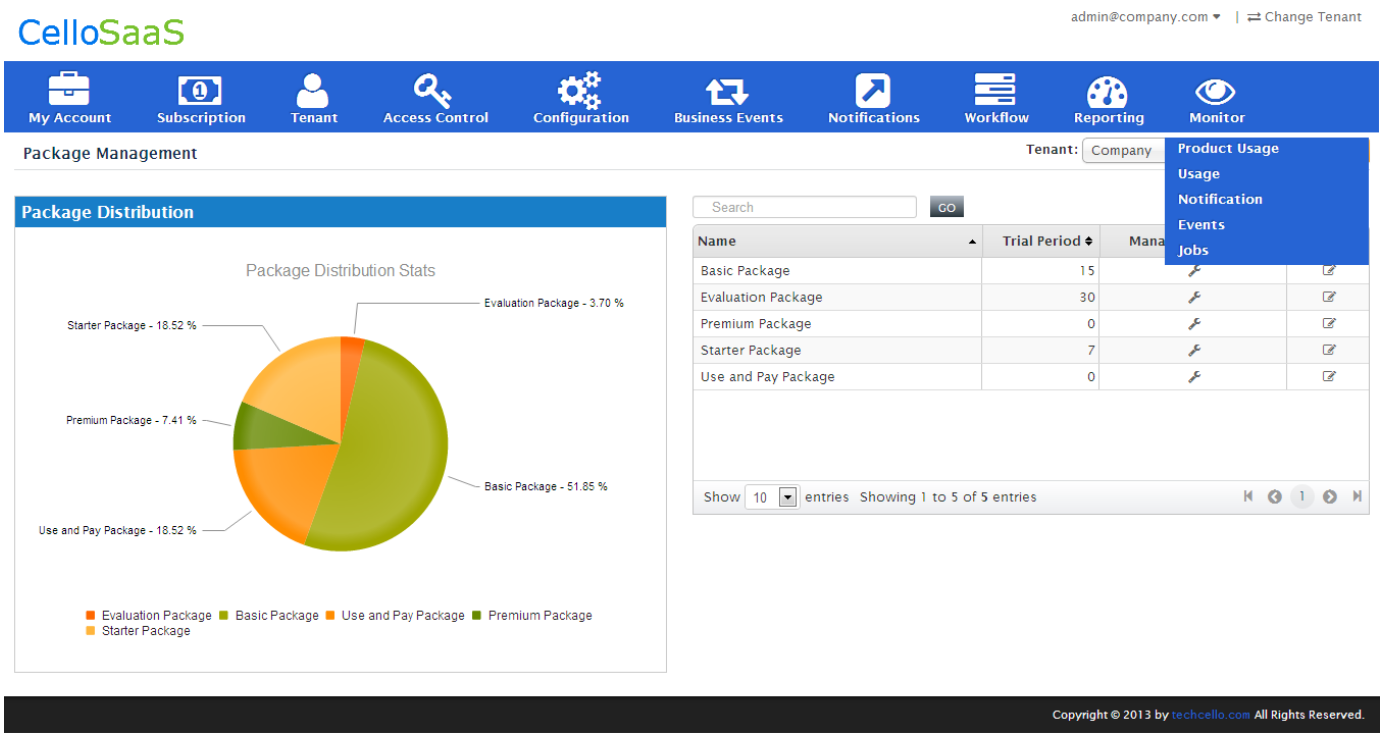
How-To – Package Management



- On selecting modules from the module list, the corresponding features and usages will get populated below.
- **Assigned:** The modules, features and their corresponding usages contained in this section are applied to the tenant that posses this package.
- **Assignable (This is relevant only while dealing with hierarchy):** The modules, features and their corresponding usages contained in this section are the ones that the tenant possessing this package can provision to the tenant's that he creates or to his child tenants. If the feature "manage packages is checked" at least one of the features in assignable has to be checked

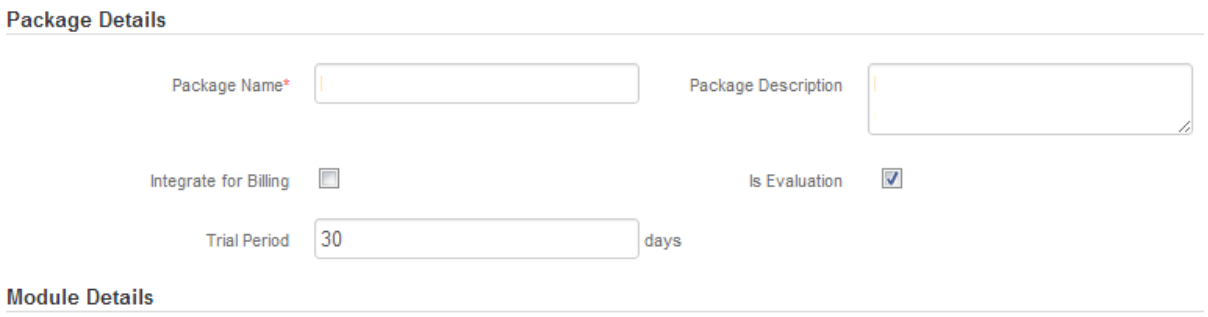
  Note:

  1. Access Control, Configuration, Setting, Users modules and its features are mandatory for primary Tenant
  2. Package and Tenant modules and its features are mandatory for Secondary Tenants
- Click the save button to create new package.



5.4 How to create Evaluation Subscription?

Evaluation copy is a trial version of the actual software service. This usually means it has limited functionality or it has all the functions as the software, including a time limit of how long you can use it before it deactivates. They are used as a kind of "try before you buy" stuff to make the customer to buy the software. To create an evaluation package, check the “Is Evaluation” checkbox present in the create new subscription page. Also while creating the evaluation copy, it is mandatory to set up the number of Trial Period Days [Refer the image below].



When a package is marked as Evaluation copy, and number of days set in Trail period text box, it is considered as Evaluation/Trial Package. Customers who choose evaluation copy will be able to use the application only up to the defined days. Customers will be notified before and after the account gets de-activated.

Note: Extension of evaluation period is restricted in the evaluation package and the only way to further use the application is to either upgrade the subscription or create a new trial account with different set of credentials, but the data that you saved with the previous account will be lost.

When a package is not marked as Evaluation copy, and set with Trail period duration, it will be considered as a regular package with Free/No Free Usage. For Example, When 30 days is set as Trial Period on a package, for the first 30 days the customer will not be billed, post 30 days the billing will be metered as per the chosen subscription pricing plan.

5.5 How to Create Trial Subscription?

Trial package and Evaluation package are one and same, and both have definite trial duration, but the only difference between Trial and Evaluation package is, with evaluation package, a customer can access the full or limited features and modules of the application for a given period of time, once the evaluation period is over, the customer has to buy a priced subscription and start using the product. Whereas in Trial Package, the customer can request the product owner to extend the Trial period for some more days, so a trial duration can be extended in non evaluation package.

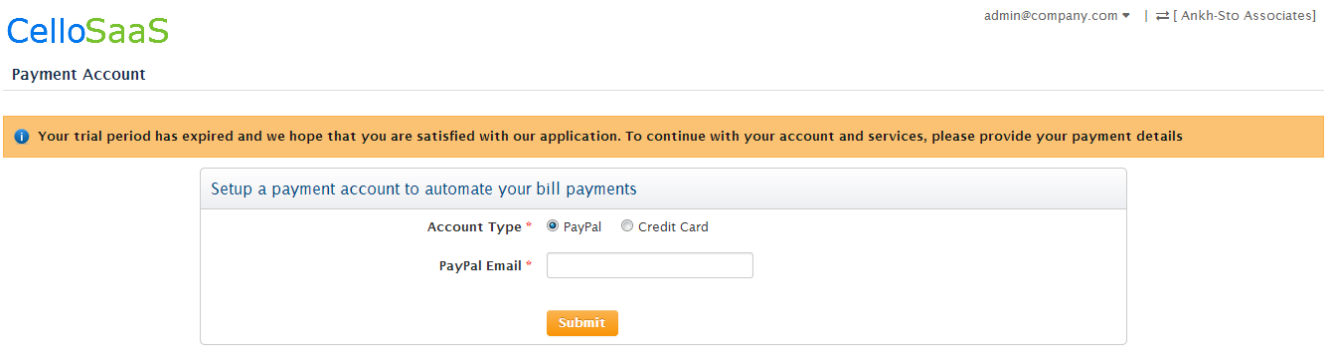
5.5.1 Login after Trial Expires

Move it to the Billing and just provide the basic info about trial expiry

Cello doesn't mandate capturing Credit card while setting up new customer in the application, but it will mandate only when the tenant nearing the Trial Expiration.

If the subscription already expired, the application will force the user to upgrade to the regular packages by filling the Credit card information, once the customer upgrades the application, the customer status will becomes "Paid Customer".

When the tenant login after Trial Expiration date, he will receive the following page soon after the Authentication.



Payment Account

In case if the Customer doesn't upgrade the account after the trial period, the user status will become "Suspended" and the customer will not be allowed to use the application until the tenant upgrades their account.

Note: If the customer doesn't want to upgrade the subscription but want to take the backup of his personal/enterprise data from the application saved during the Trial Period can be backed up and taken out form the system.

5.6 Types of Trial Package

There are two kinds of Trial Package can be created.

1. On Going Trial (Unless the Admin Blocks the account, it will remain active, so the Product Admin has to manually de-activate the account)

2. Blocking Trial – The Trial subscription will automatically be de-activated once the Trial Period expires.

5.7 Set Usage

While creating a subscription, it is optional to setup usage for the package.

Example:

If you are intent to build a Document management system and you are billing model is based on the number of documents uploaded by the customer. Then you may set

Basic Package:

Document Limit: 500 (in numbers)

Advanced Package:

Document Limit: 5000 (in numbers)

Enterprise Package:

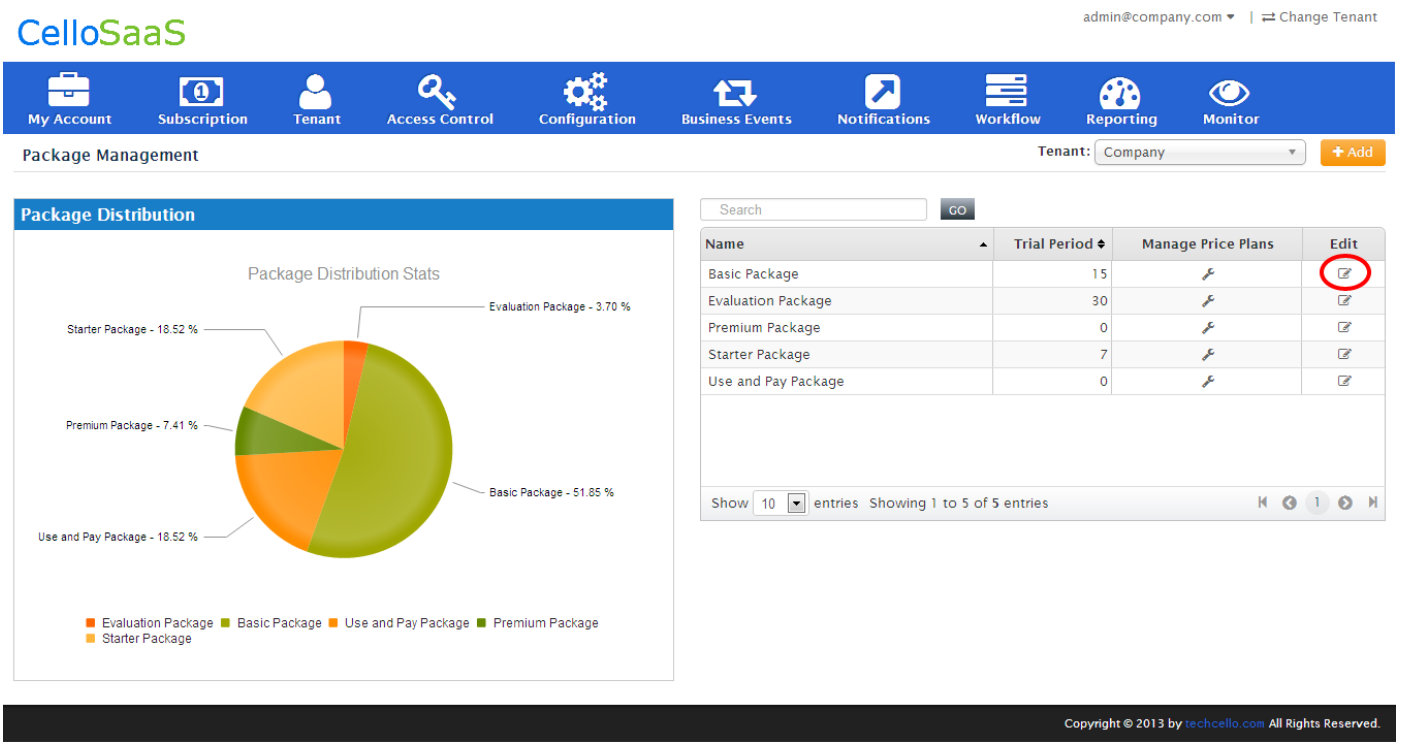
Document Limit: Unlimited

Note:

1. Modules like 'Access Control', 'Configuration', 'Setting', 'User' modules are mandatory for all packages. For tenant hierarchy support 'Package', 'Tenant' modules are mandatory.
2. Leave blank or set ‘0’ to make the threshold unlimited.

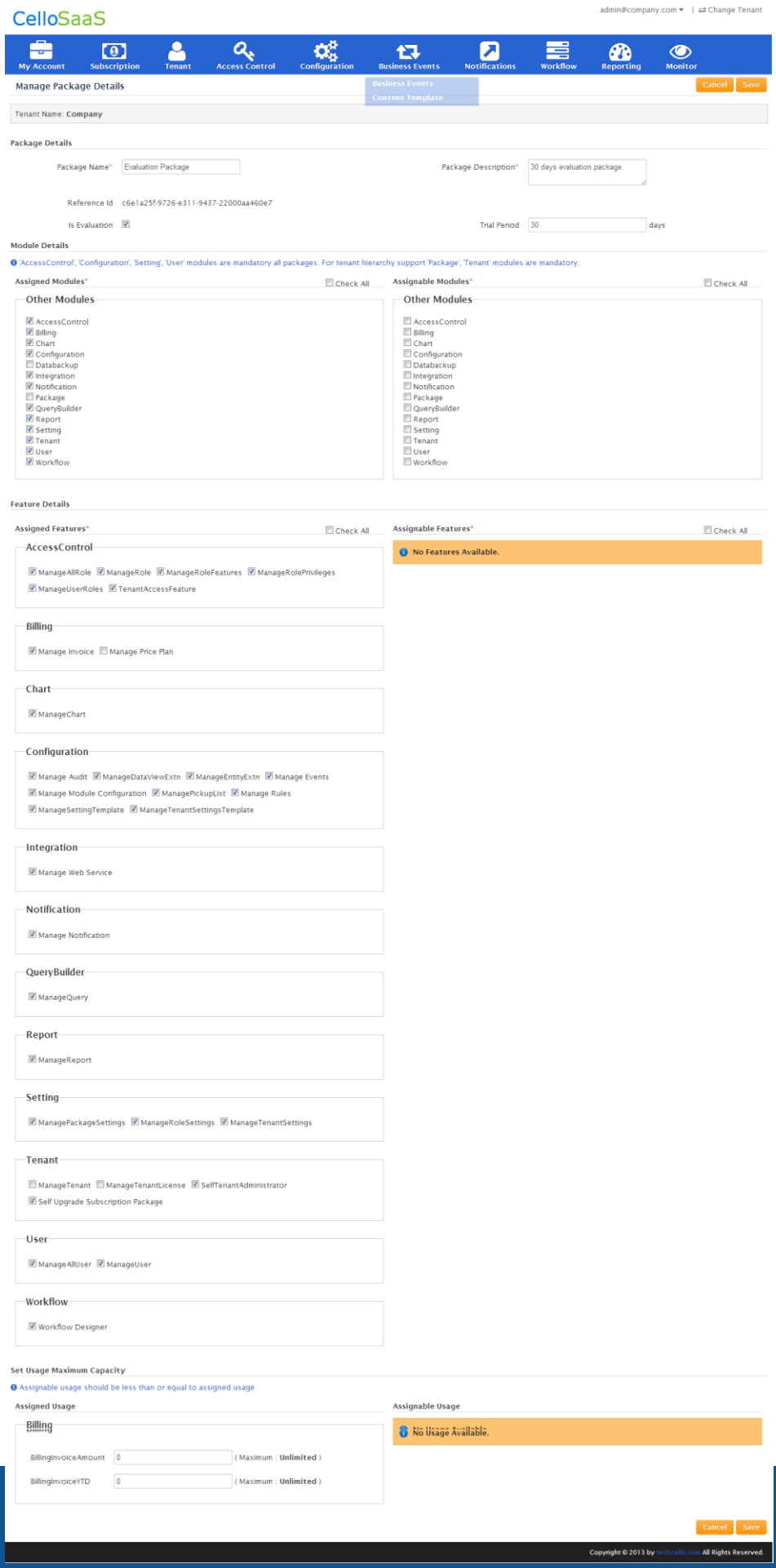
5.8 Edit Package

- Click edit icon to edit package details of specific package from package list.



- Edit package details will contain the assigned and assignable modules, features and usage for the specific package.
- Click save button to save the changes in the package details.

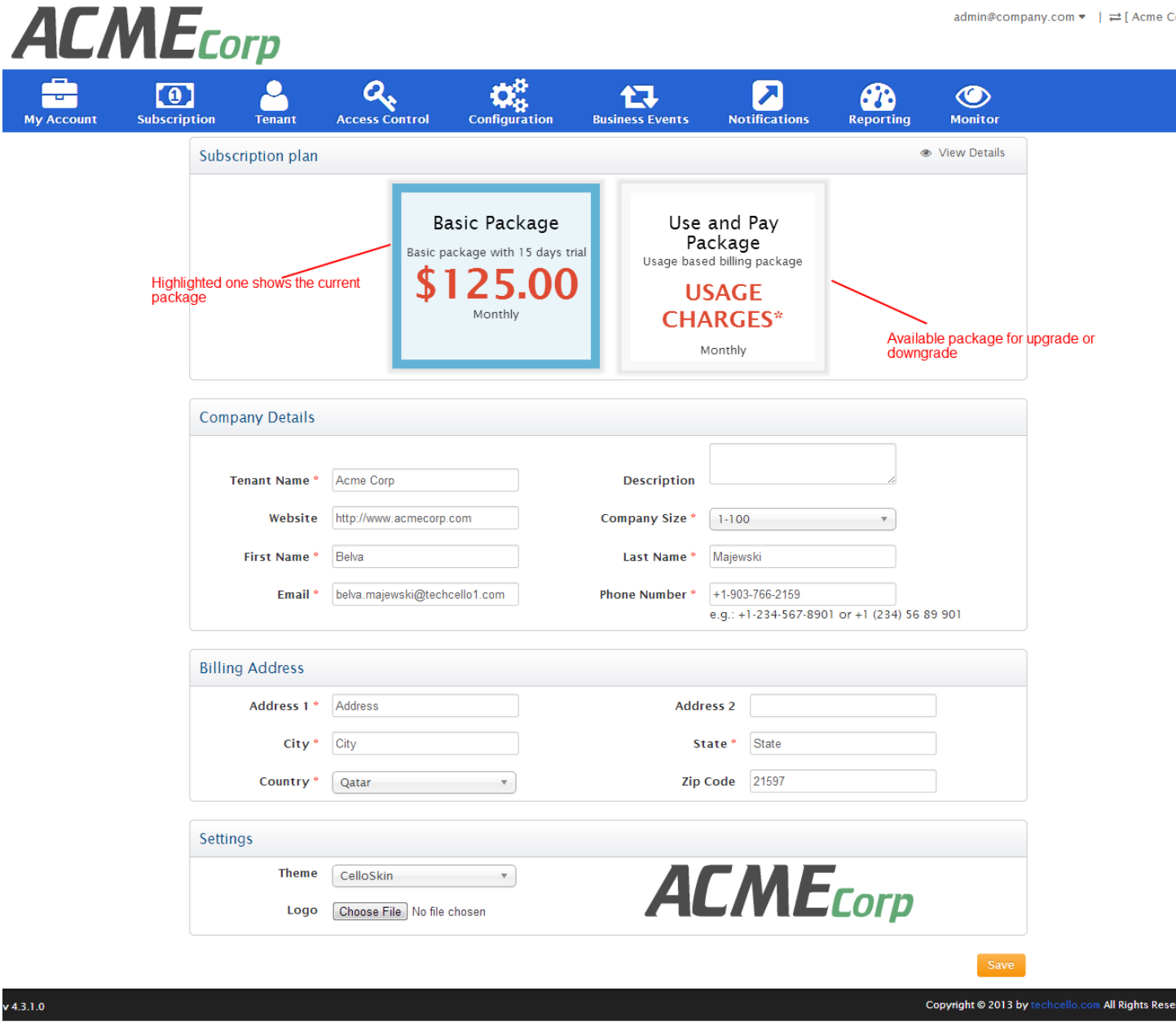
How-To – Package Management



5.9 Subscription Self Upgrades/Downgrades

To initiate an upgrade or downgrade through the Admin Interface, Navigate to My Account → Account Settings.

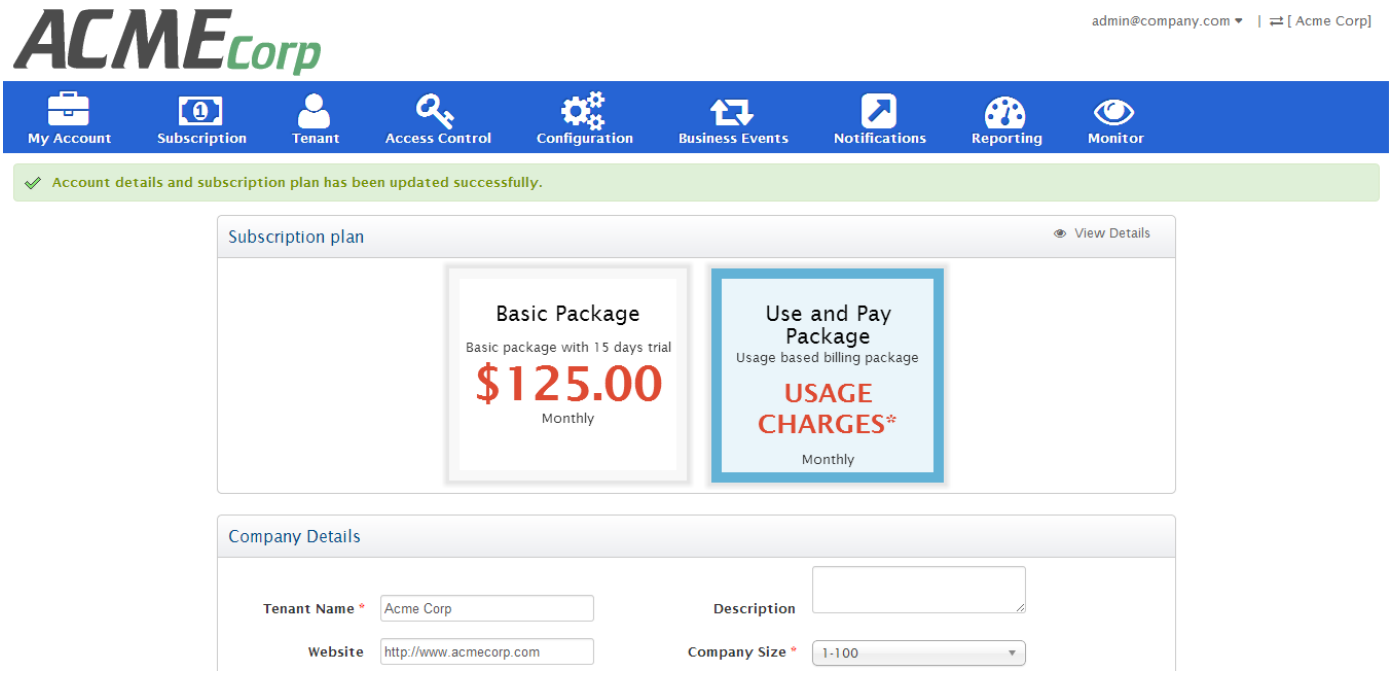
The account settings page displays the current holding Subscription plan (Highlighted) and other plans for upgrade or downgrade.



Account Settings

Select the Plan you want to upgrade/downgrade to and click on save. Once the subscription plan is changed, an automated emailer will be sent from Notification Engine to the respective Tenant administrator. Product Administrator can update the content of the "Plan Change Emailer". To update Navigate to Notifications→ Content Template and look for "Tenant Subscription Changed Mail" and click on Manage Template icon.

You will see a success message upon completion of the upgrade or downgrade (Refer the below Screenshot).



Upgrades/Downgrades can also be initiated through API; Refer API Guide

Note: While there is no limitation imposed while upgrading from small plan to higher end plan, where as when downgrading from higher end plan to lower plan, there will be limitations based on the current usage of the customer.

Consider a Scenario:

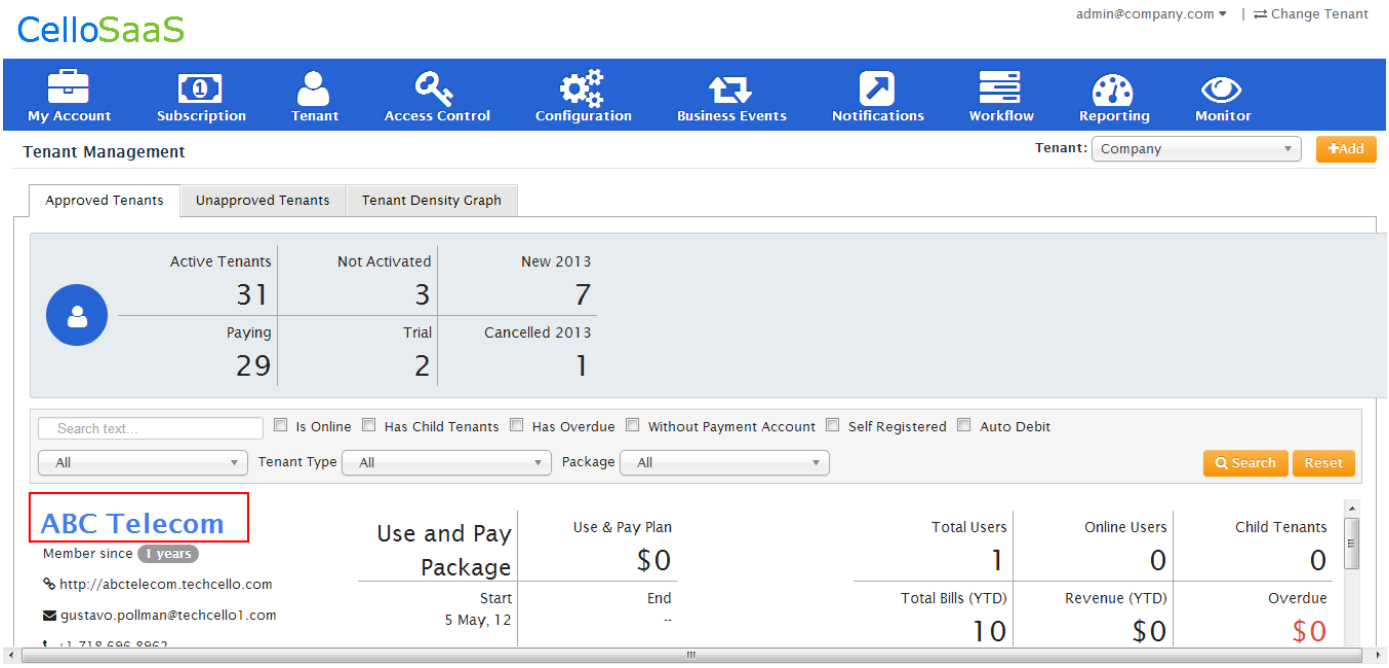
Customer A currently holds an **Enterprise Plan** which allows them to send up to **1000 email** for about **1000 $** in a billing cycle.

During the 15th of a particular month, he has used up to **800 emails,** now if he decides to downgrade the plan to another; he must receive the plans which have the email counts less than **200 emails.** (This is totally under the discretion of the Product owner, but cello doesn't impose such Policies by default).

5.9.1 Upgrade/Downgrade Subscription plan of Tenant by Prod Admin

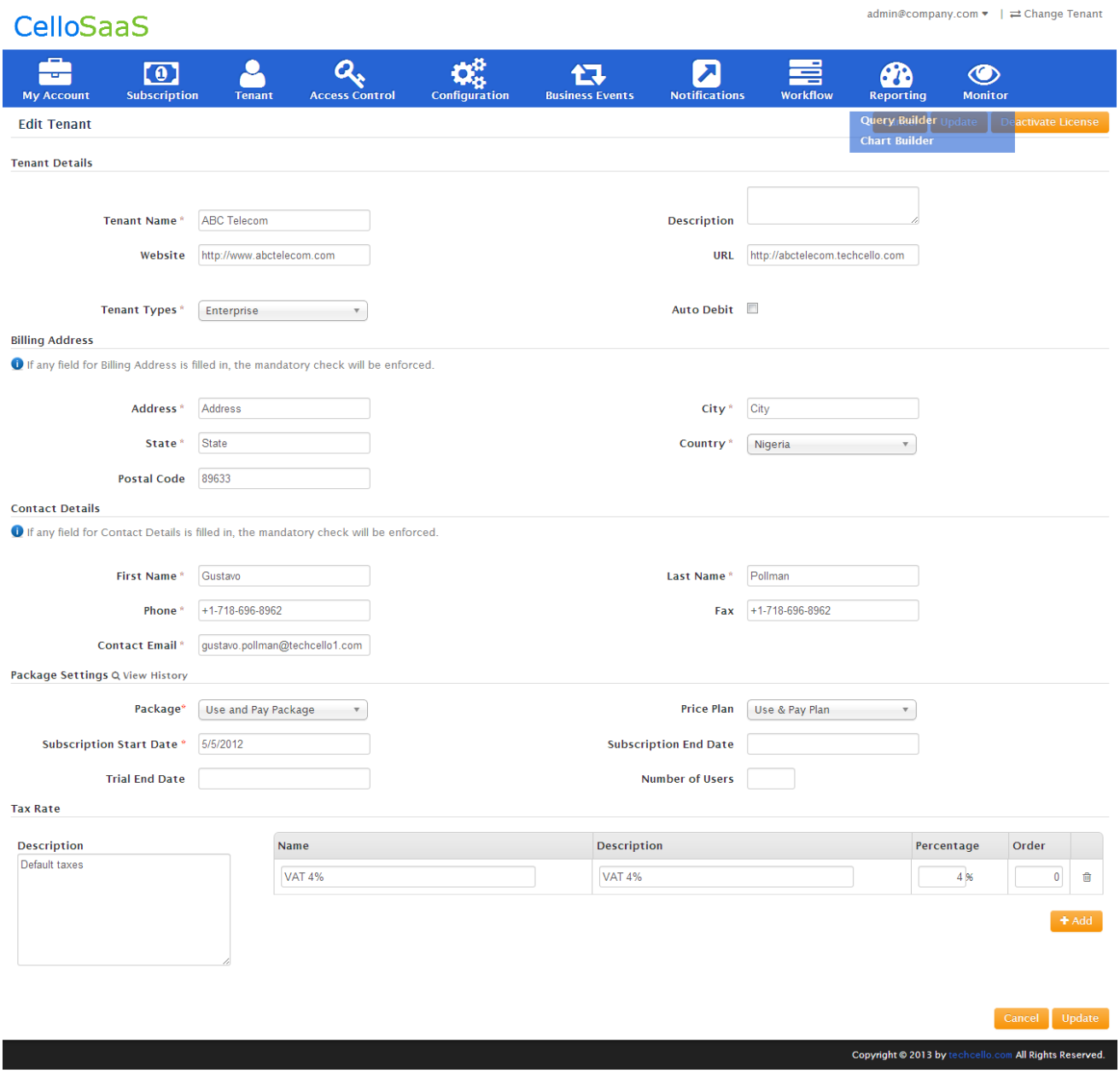
Product administrator can change the subscription plans of the Tenant using Tenant Management Module.

- To change the Subscription Login as Product Administrator → Tenant→ Manage Tenants [Refer below Screenshot]



Tenant Management

Click on the Tenant Name, this opens the Tenant details in the edit mode [Refer below Screen]



Edit Tenant Details

- Select the desired package and a plan
- Click on Save

Subscription Changes (Prorated)

Prorated credit: A prorated credit will be created to reimburse the Customer for their current product. Note that this is not a refund; no money will be returned to the user. The credit only affects the Customer's balance.

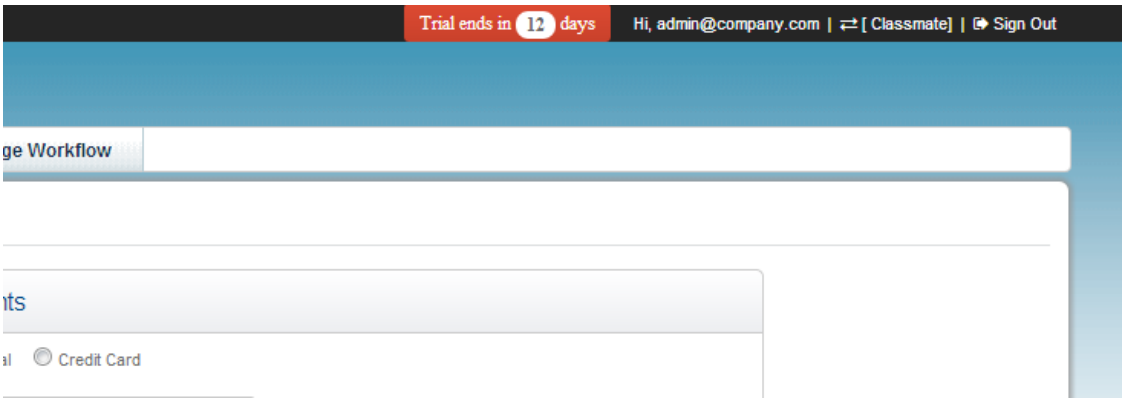
Charge: A charge for the full amount of the new product will be applied

Customers can be allowed to upgrade/downgrade the subscribed plans at any point of time,

a. Usage Limit Alerts
b. Upgrade Alerts

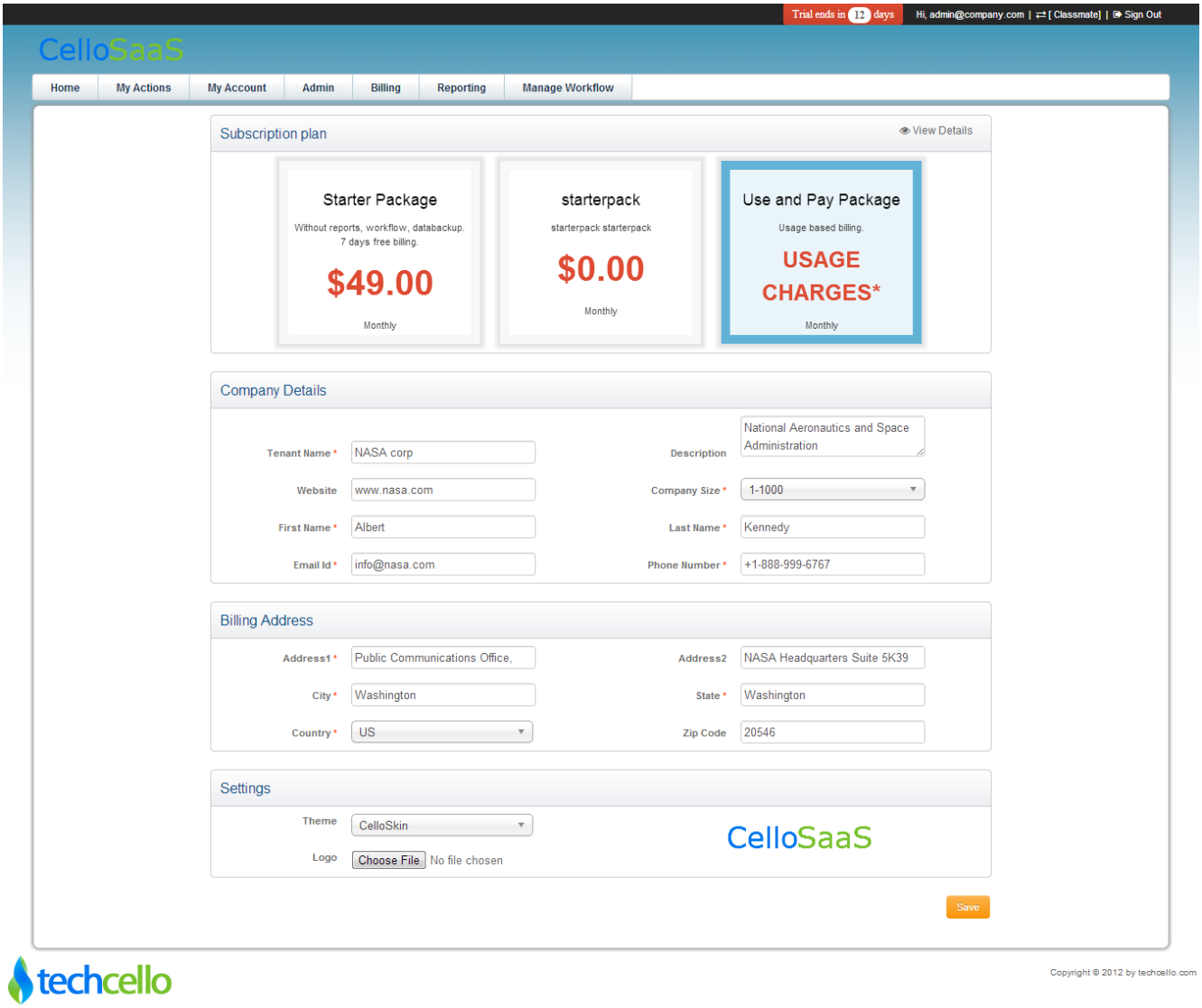
Self Upgrade

Customers who are currently using Evaluation Packages will receive a Call-To-Action message in the top bar with the number of Evaluation Days left. Towards the end of the Trial period, the customer will receive an email notification as the evaluation period is over and links to upgrade with enterprise plan.



During or after the evaluation period, Customers can click on the Call-To-Action alert message and self upgrade by themselves without any assistance from the product administrator.

Customers can also upgrade/downgrade their current plan by navigating to My Account → Upgrade/Downgrade Subscription.

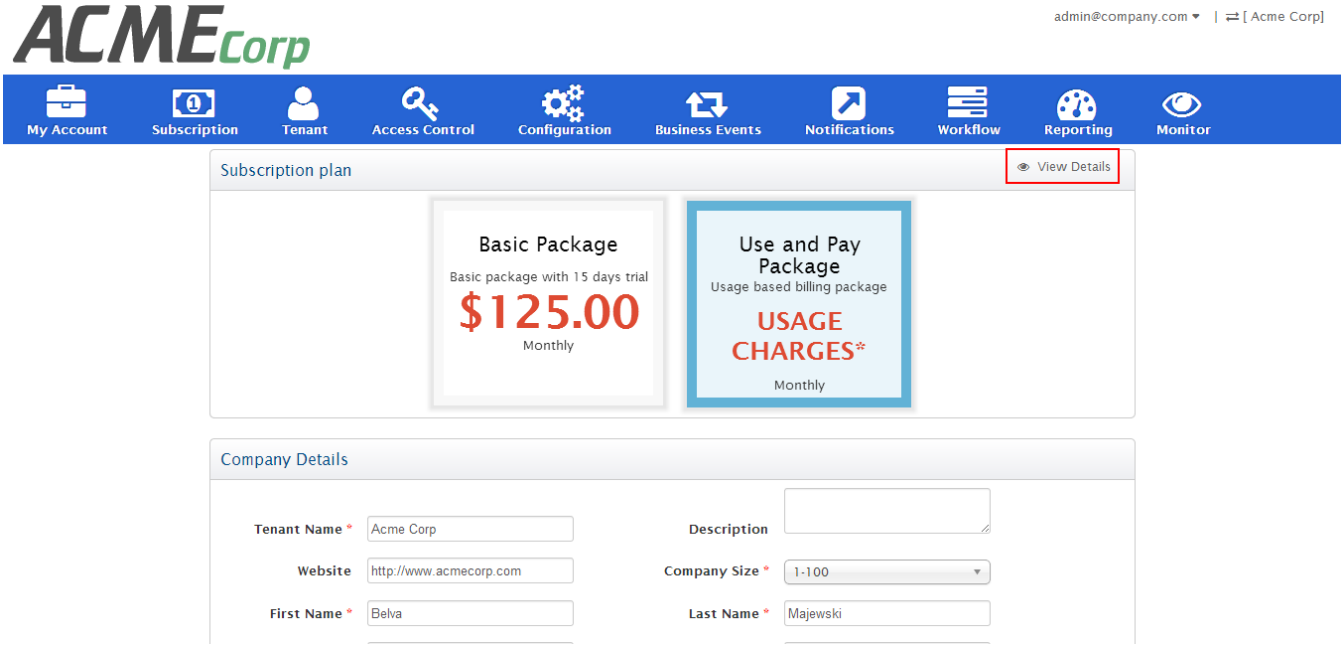


i. Validations

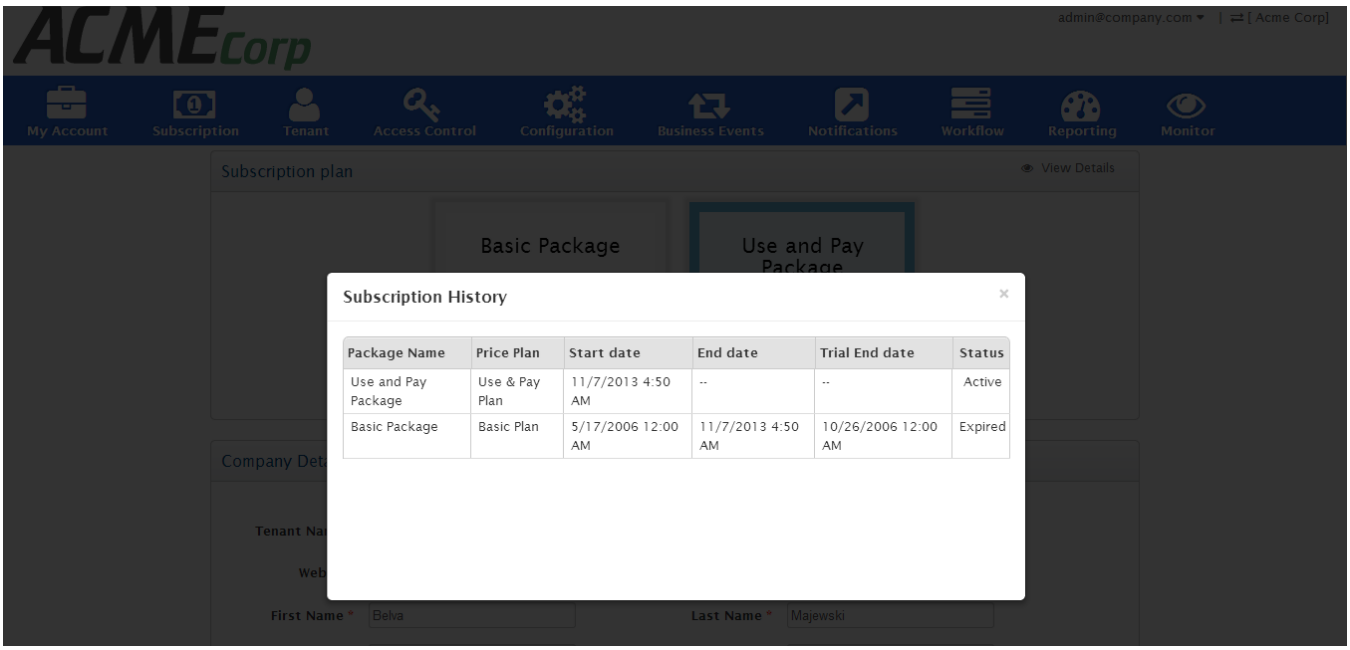
c. Admin Upgrade
   i. Validations
d. Upgrade - Invoice calculations
e. Upgrade Trial Period Impact

6 Tenant's Subscription History

Over the course of Tenant's lifetime in the application, customers might change any subscription plan, customers can upgrade/degrade the subscription plans. Tenant administrators can view the entire subscription history by navigating to My Account → Account Settings→ Click on View details



Account Settings



Subscription History Modal Pop up

7 How to register application specific Modules

Module Configuration through Fluent API

- Each module has to be configured with the Cello framework.
- The Module Configuration class has to implement **IModuleConfigurator** interface and provide implementation and pass reference object of type **ModuleConfig**.

- The **ModuleConfig** has the following methods and members.

Method and Parameters	Description
Add(string moduleCode, string moduleName = null)	Specify the Code and Name for the particular Module
WithFeatures(Action<FeatureConfig> featureFunction)	Specify the Feature name
WithName(string moduleName)	Pass the privilege name for the module
WithService(Action<Service> funService)	Pass the service details for the module

Add()

It takes Module Code as the first parameter and Module name as the second parameter [optional].

Eg:

```
moduleConfiguration.Add("ProjectModule", "Project Module")
```

WithFeatures()

It takes action methods as delegates and pass, ***FeatureConfig.Add*** method and passes the *Feature Name* as the parameter then calls ***FeatureConfig .WithName*** method and pass Friendly name as the parameter for the feature.

Eg:

```
.WithFeatures(f => f.Add("ProjectFeature").WithName("Project Feature")
```

Note: Every feature should be associated with its corresponding privileges. This will be used for restricting the access to the features. Please refer the Access Control management feature for additional details in Cello framework developer guide.

WithPrivileges()

It takes action methods as delegates, pass ***FeaturePrivilegeConfig.Add*** method and provides the Feature Name as the parameter then call ***FeaturePrivilegeConfig.WithName*** Method and pass Friendly name as the parameter then *call* ***FeaturePrivilegeConfig.WithDescription*** and provide Privilege description as the parameter.

Eg:

```
.WithPrivileges(p => p.Add("Search_Project").WithName("Search Project").WithDescription("user
can search their Project  list")
                                        .Add("View_Project").WithName("View
Project").WithDescription("user can View their Project  list")
                                        .Add("Add_Project").WithName("Add
Project").WithDescription("user can add their Project  list")
                                        .Add("Edit_Project").WithName("Edit
Project").WithDescription("user can edit their Project  list")
                                        .Add("Delete_Project").WithName("Delete
Project").WithDescription("user can delete their Project  list")))
```

WithUsage()

WithUsage method allows you to define usages against each module configured in the system. The Parameters are 1. Usage Variable Code 2. Usage Variable Name (Friendly Name) 3. Type of usage along with basic threshold if any i.e 500 Documents or contacts allowed etc.

Examle:

```
.WithUsages(u => u.Add("ProjectUsage").WithName("Project
Usage").WithTypeName(UsageTypeConstants.BlockUsage).WithTypeCode(UsageTypeConstants.BlockUsage)
)
```

WithEntities()

WithEntities method allows you to all the business entities of the application.

Examle:

```
.WithEntities(e => e.Add<Model.Project>())
```

Module Configuration Example:

```
using CelloSaaS.Configuration;
using CelloSaaS.Rules.Execution;
using CelloSaaS.ServiceContracts.LicenseManagement;
using ProjectManagement.Model;

namespace ProjectManagement
{
    /// <summary>
    /// Project module configuration
    /// </summary>
    public class ProjectModuleConfigurator : IModuleConfigurator
    {
        /// <summary>
        /// Register Project details with CelloSaaS.
        /// </summary>
        /// <param name="moduleConfiguration">The module configuration.</param>
  public void Configure(ModuleConfig moduleConfiguration)
  {
  //cellohelp:moduleconfiguration
  moduleConfiguration.Add("ProjectModule", "Project Module")

  //cellohelp:usageconfiguration
  .WithUsages(u => u.Add("ProjectUsage").WithName("Project
Usage").WithTypeName(UsageTypeConstants.BlockUsage).WithTypeCode(UsageTypeConstants.BlockUsage))

  //cellohelp:featureconfiguration
  .WithFeatures(f => f.Add("ProjectFeature").WithName("Project Feature")

  //cellohelp:entityconfiguration
  .WithEntities(e => e.Add<Model.Project>())

  //cellohelp:privilegefiguration
  .WithPrivileges(p => p.Add("Search_Project").WithName("Search Project").WithDescription("user
can search their Project  list")
  .Add("View_Project").WithName("View Project").WithDescription("user can View their Project
list")
  .Add("Add_Project").WithName("Add Project").WithDescription("user can add their Project  list")
  .Add("Edit_Project").WithName("Edit Project").WithDescription("user can edit their Project
list")
  .Add("Delete_Project").WithName("Delete Project").WithDescription("user can delete their
Project  list")))

}

}
```

Global.asax Configuration Example:

```
private void Configure()
        {

CelloSaaS.Configuration.CelloConfigurator.RegisterModule<CelloSaaS.Notification.NotificationModule
Configurator>();

CelloSaaS.Configuration.CelloConfigurator.RegisterModule<CelloSaaS.WorkFlow.WorkflowModuleConfigur
ator>();

CelloSaaS.Configuration.CelloConfigurator.RegisterModule<CelloSaaS.DataBackup.DataBackupModuleConf
igurator>();

CelloSaaS.Configuration.CelloConfigurator.RegisterModule<CelloSaaS.Integration.IntegrationModuleCo
nfigurator>();

CelloSaaS.Configuration.CelloConfigurator.RegisterEntity<CelloSaaS.WorkFlow.WorkflowEntityConfigur
ator>();

CelloSaaS.Configuration.CelloConfigurator.RegisterModule<CelloSaaS.Configuration.DBCelloModuleConf
igurator>();

CelloSaaS.Configuration.CelloConfigurator.RegisterEntity<CelloSaaS.Configuration.DBCelloEntityConf
igurator>();

CelloSaaS.Configuration.CelloConfigurator.RegisterDataView<CelloSaaS.Configuration.DBCelloDataView
Configurator>();


CelloSaaS.Configuration.CelloConfigurator.RegisterEntity<ProjectManagement.ProjectEntityConfigurat
or>();

CelloSaaS.Configuration.CelloConfigurator.RegisterModule<ProjectManagement.ProjectModuleConfigurat
or>();

CelloSaaS.Configuration.CelloConfigurator.RegisterDataView<ProjectManagement.ProjectDataViewConfig
urator>();

CelloSaaS.Configuration.CelloConfigurator.RegisterDataView<ProjectManagementViewModel.ProjectViewM
odelDataViewConfigurator>();


CelloSaaS.Rules.Execution.RuleConfiguration.Register<ProjectManagement.ProjectRuleConfiguration>()
;

CelloSaaS.Configuration.CelloConfigurator.Configure();


CelloSaaS.Configuration.DalExtensionsConfiguration.Register<CelloSaaS.Services.DataEventHandlers.C
URDEventRasieEventHandler>();
}
```

8 Contact Information

Any problem using this guide (or) using Cello Framework. Please feel free to contact us, we will be happy to assist you in getting started with Cello.

Email: support@techcello.com

Phone: +1(609)503-7163

Skype: techcello