Cello How-To Guide

Configuring and Consuming CelloSaaS WCF Services

techcello

Contents

1 Introduction

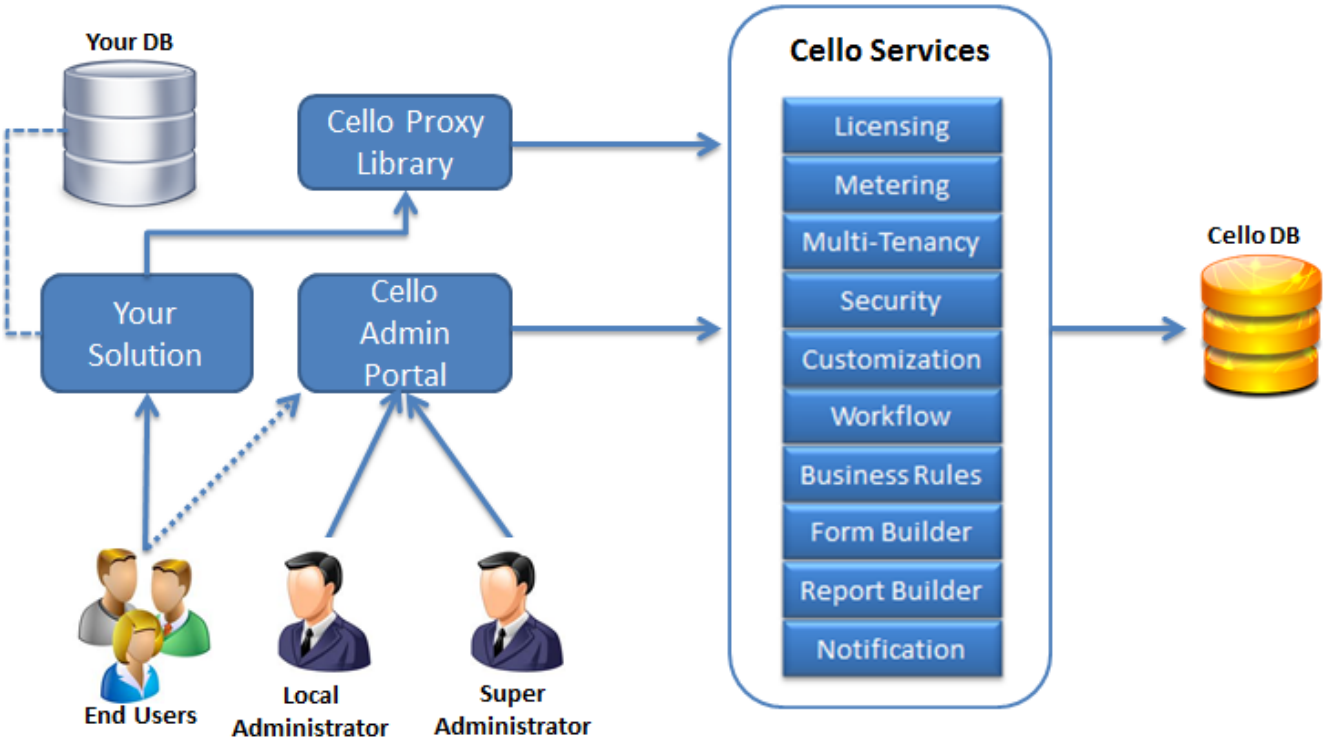
There are two major ways to consume CelloSaaS Services. They are

1. InProc
2. Service Oriented

InProc is tighter Coupling Cello Multi Tenant Application with the actual application i.e ERP, CRM, Business Management Solution etc. In the Inproc mode, both the Cello Application as well as Business Application can be deployed as a single Solution or two different solutions.

Service Oriented mode is best suitable for all sorts of application. In this model, the Core application will not be tightly coupled with the Techcello SaaS Admin Portal, hence the deployment can be done in two different servers or in the same server.

Both the approaches have Pros & Cons. Techcello doesn't mandate either of the two and the options are open to the technical/development team of the Product.

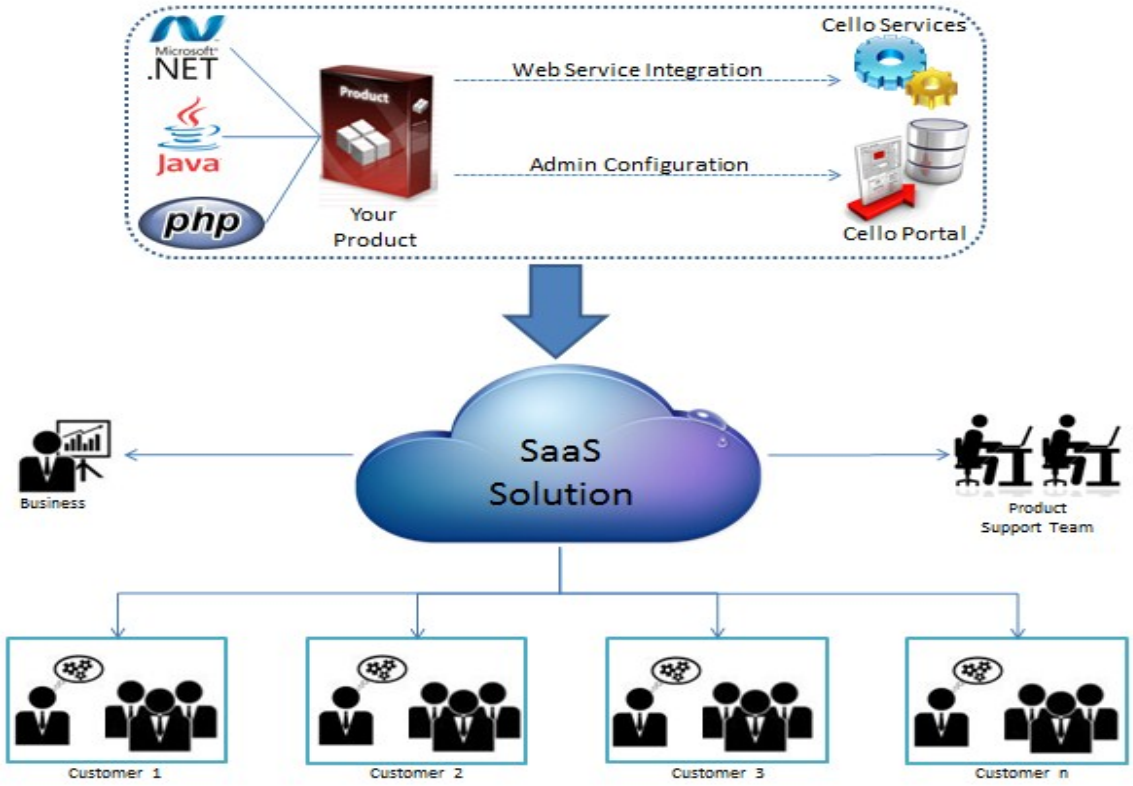


This document highlights the procedure to configure, deploy consume Cello functionalities in Services model also discuss about the enablement and utilize Cello's Web Security architecture for the application related Services.

2 Windows Communication Foundation (WCF) service

Cello Architecture is based on SOA and all its Features and Modules are completely stand alone and independent in nature. CelloSaaS service methods are exposed as web services using windows communication foundation (WCF). It utilizes transport and message level security mode which ensures proper security for the data being transmitted in motion.

This topic outlines the basic steps required to Deploy, Configure and Consume CelloSaaS WCF services that is hosted in Internet Information Services (IIS).



Techcello SOA Implementation

2.1 Hosting CelloSaaS WCF service

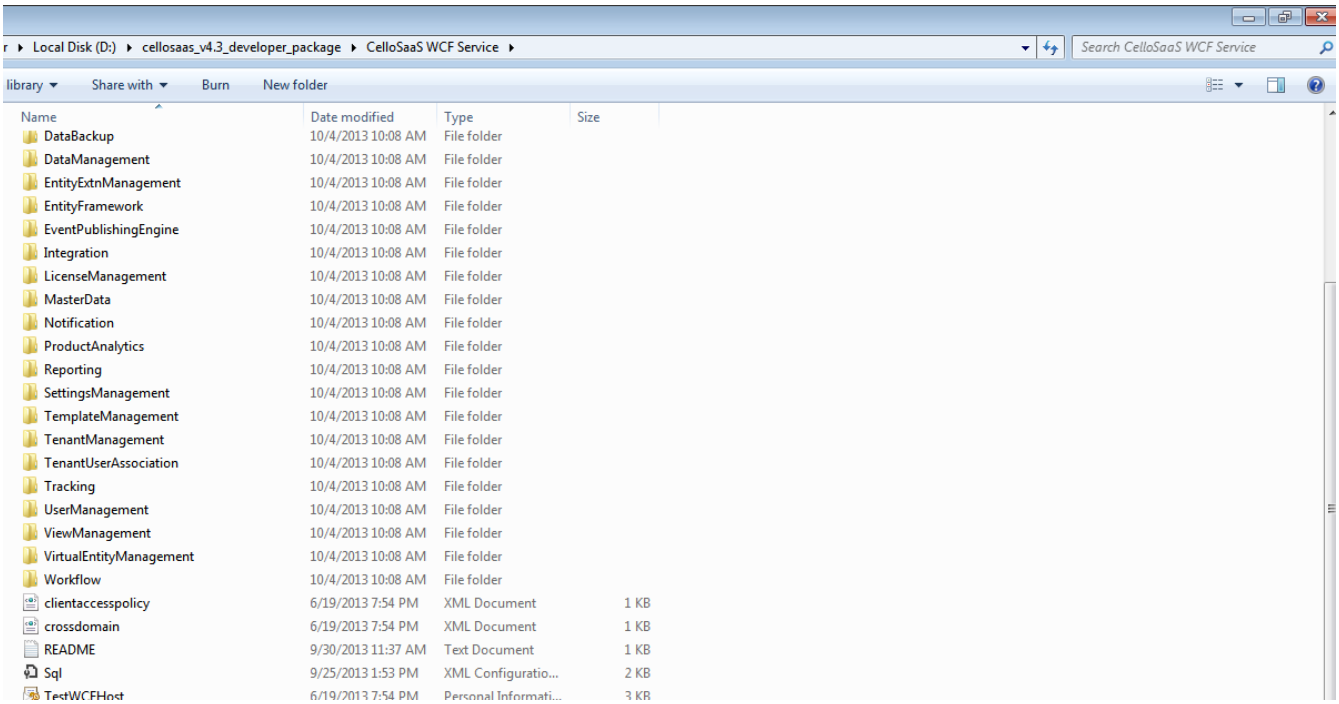
CelloSaaS Web Service Project can be found in the CelloSaaS Package. There are various ways in which you can Host the WCF Services.

They are

- Visual studio Web Development server
- Managed Windows Services

- Internet Information Services (IIS)

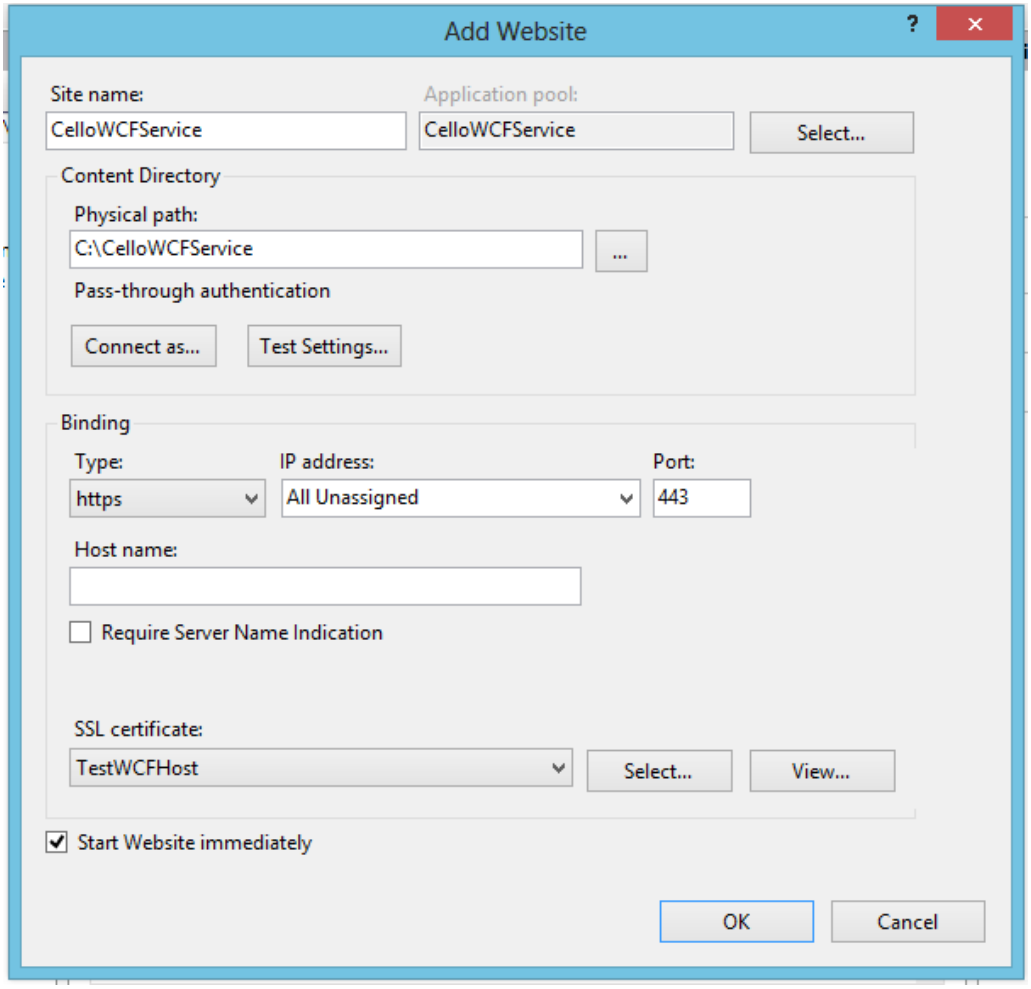
The Option # 1, is ideal for the Development phase, but during the deployment you can either choose any one of the other two options



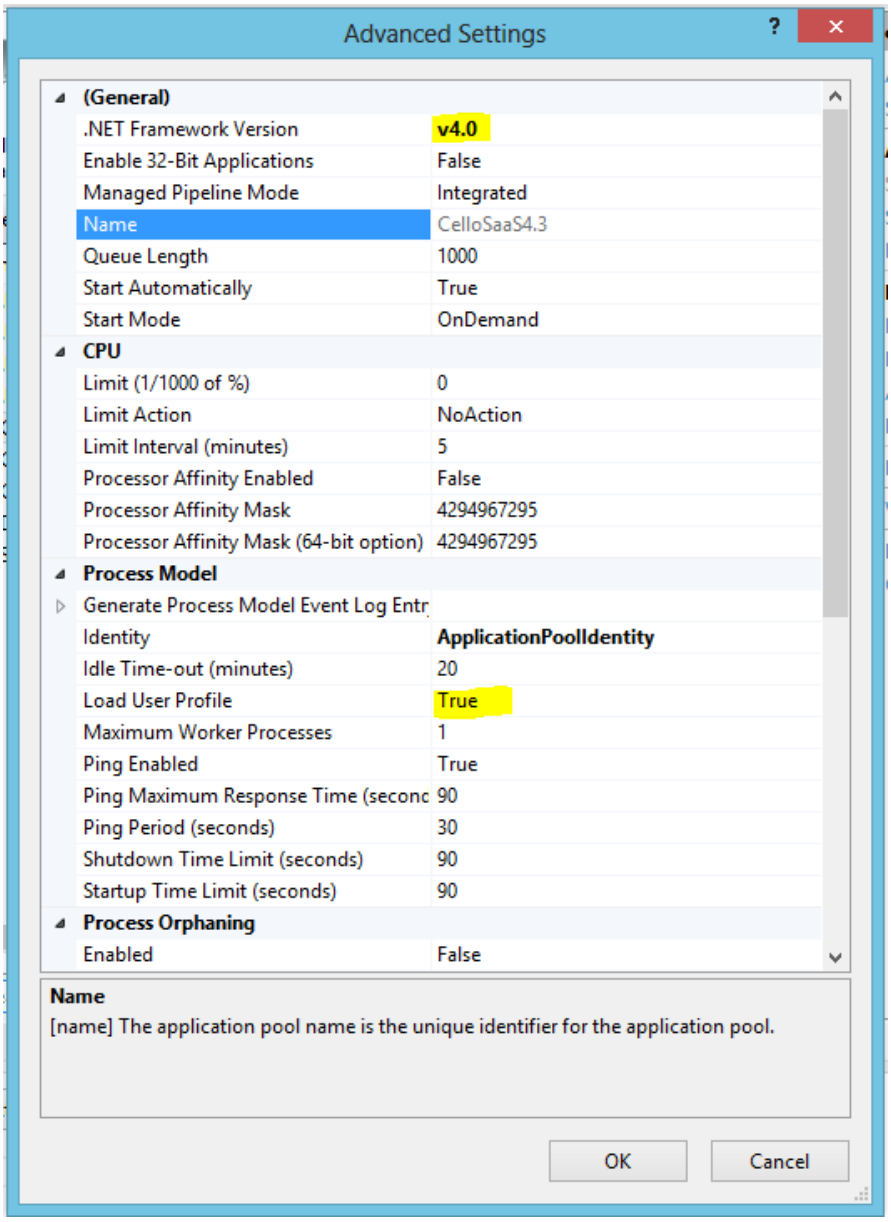
2.2 How to Host Cello Services in IIS

Hosting CelloSaaS Web Services is as similar as hosting any other WCF Services. Below are the steps to be followed to host the Cello WCF Services

1. Open IIS Manager (Windows +R Key -> Type “inetmgr”)
2. Right click on sites and select Add Web site.
3. Enter site name as “CelloWCFService”
4. Select the physical path of folder containing the
5. Choose Binding type “https” and port 443. You can change the port if you wish.
6. Choose a valid SSL certificate. (Normally www.domainname.com)
7. Click OK.
8. Change the application pool settings as indicated in the below picture.



9. Highlight the site in the Website Explore and click on "Advance Settings"



10. Change the ".Net Framework" to 4.0

11. Change the Load User Profile to "True" (Mandatory to verify CelloSaaS License)

2.3 Encryption Certificate Installation

CelloSaaS WCF service can be configured to operate in various security modes, they are

- **Message** (Access via http, SOAP message encryption applied)
- **Transport** with message credential (Access via https, both transport and SOAP message encryption applied)

During the development mode, it is perfectly alright to host the Cello Services or Application Services in HTTP mode, but the same is not advisable in the Production mode, So Cello Recommends using Security Certificates to be installed in the production environment.

If you do not wish to access the Cello services via https please host the site as described above in http binding.

To enable **Transport** security, .Net WCF runtime requires a valid security certificate file in place (including private key). For development purpose we can create self signed certificates using many tools available.

Reference:

http://technet.microsoft.com/en-us/library/cc753127(v=ws.10).aspx

http://www.akadia.com/services/ssh_test_certificate.html

2.4 How to deploy the Test Security Certificate

CelloSaaS ships a self-signed certificate "TestWCFHost.pfx" along with the Services Project which could be used for development purpose without needing to create your own certificate.

Below are the steps to be followed to install the Security Certificate.

Step1: Open a command prompt and enter the following:

Note:

1. Modify the path to point to the Visual Studio installed folder.
2. Modify the names as per you're organization.
3. Change the certification password too (pass@123), editable strings are highlighted in yellow.

Step1: call "D:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\vcvarsall.bat" x86

makecert -n "CN=Techcello Certificate Authority" -cy authority -a sha1 -sv "TechcelloPrivateKey.pvk" -r "TechcelloCA.cer"

makecert -n "CN=www..domainname.com" -ic "TechcelloCA.cer" -iv "TechcelloPrivateKey.pvk" -a sha1 -sky exchange -pe -sv "DomainPrivateKey.pvk" "www.domainname.com.cer"

cert2spc TechcelloCA.cer TechcelloCA.spc

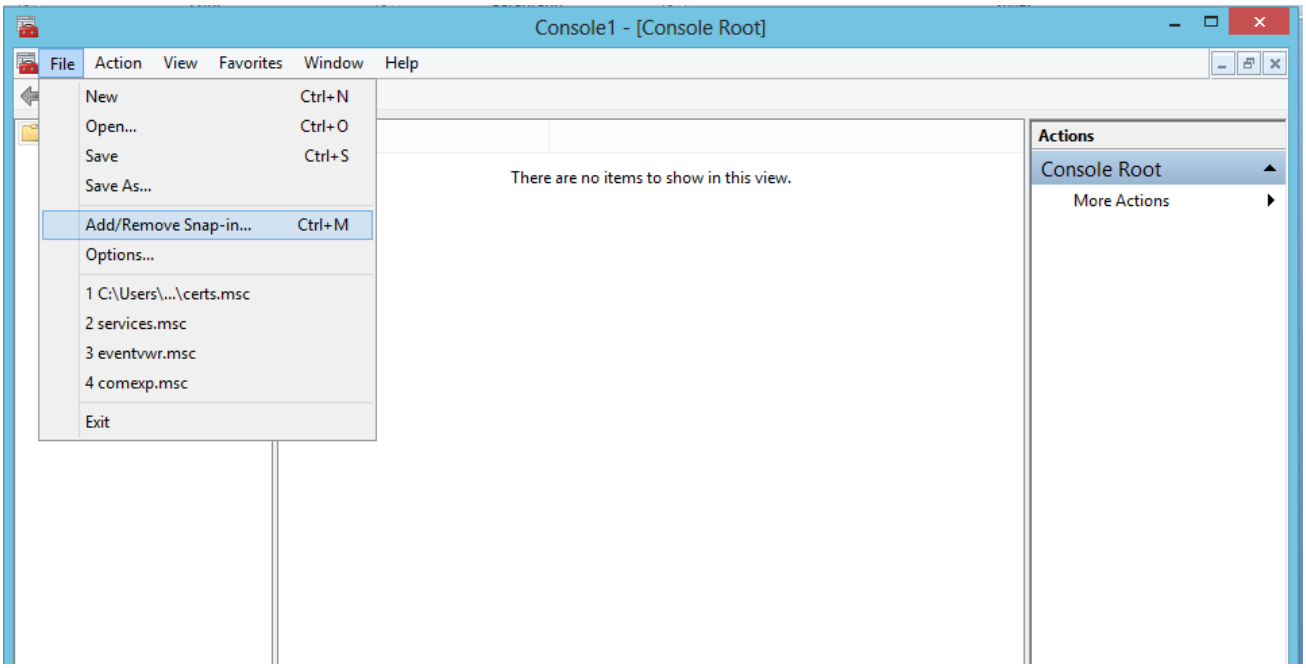
cert2spc www.domainname.com.cer www.domainname.com.spc

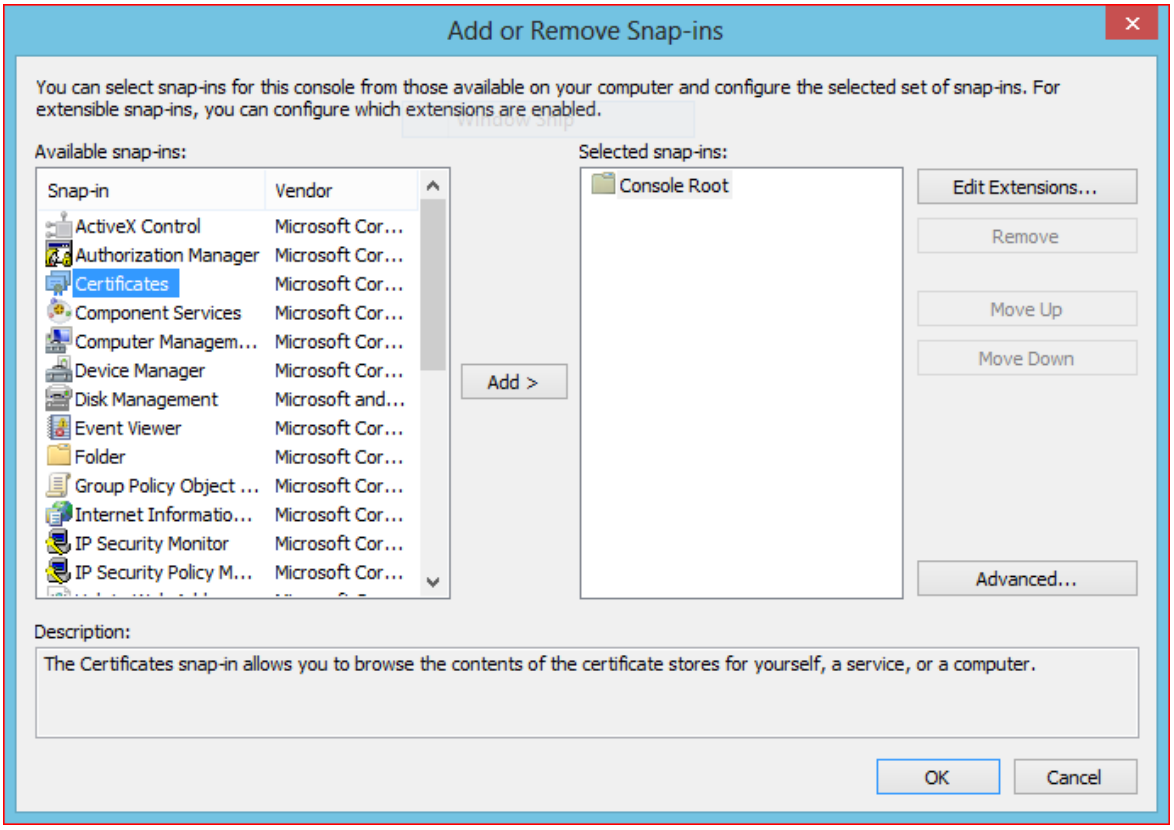
pvk2pfx -pvk "DomainPrivateKey.pvk" -spc "www.domainname.com.spc" -pfx "www.domainname.com.pfx" -pi pass@123

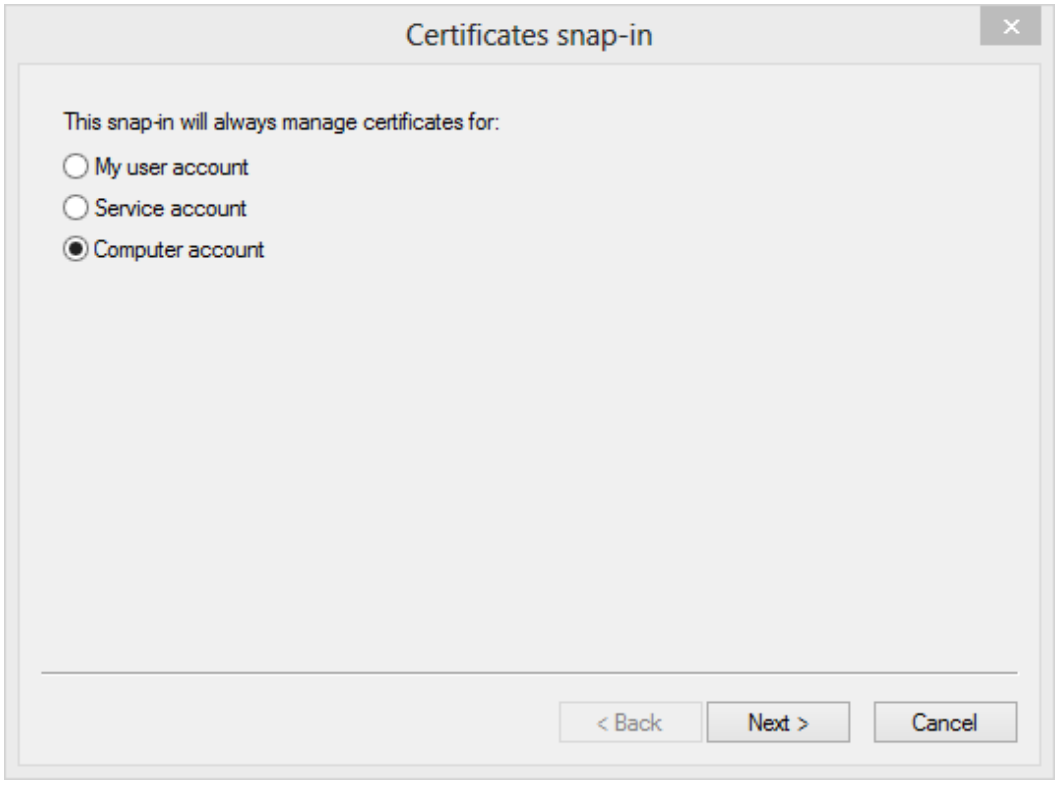
pvk2pfx -pvk " DomainPrivateKey.pvk" -spc "www.domainname.com.spc" -pfx "DomainCA.pfx" -pi pass@123

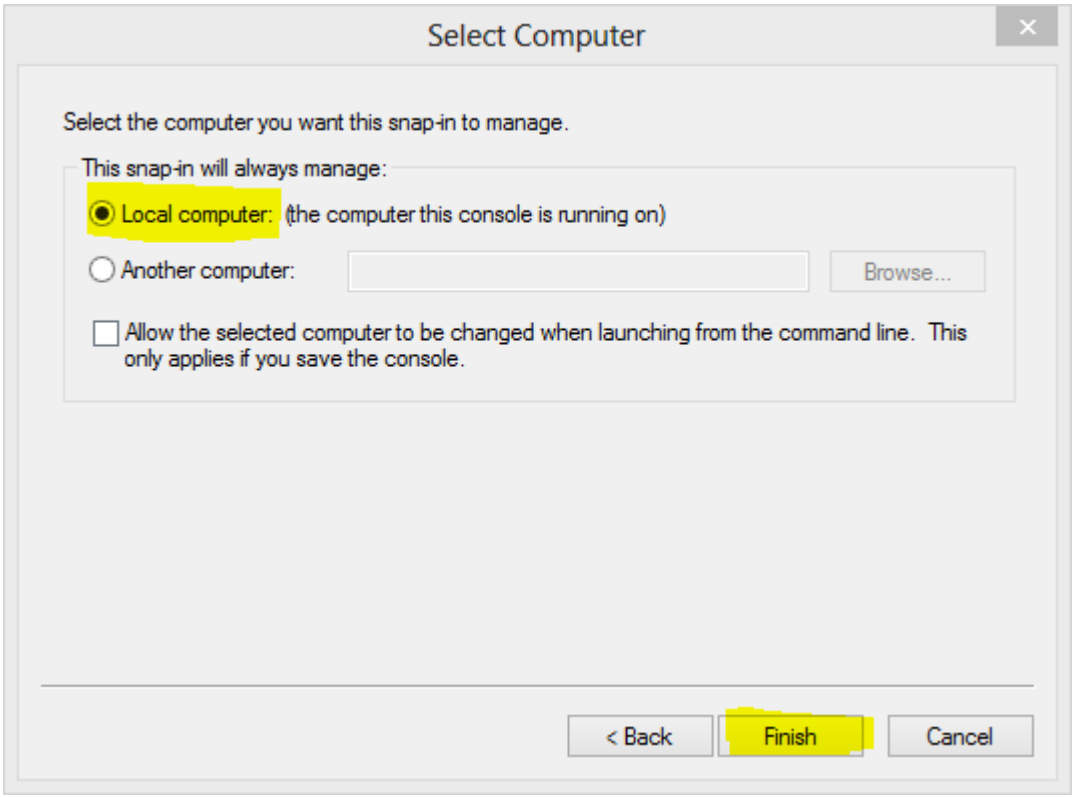
Step2: Install this certificate in the hosted machine following the below procedure.

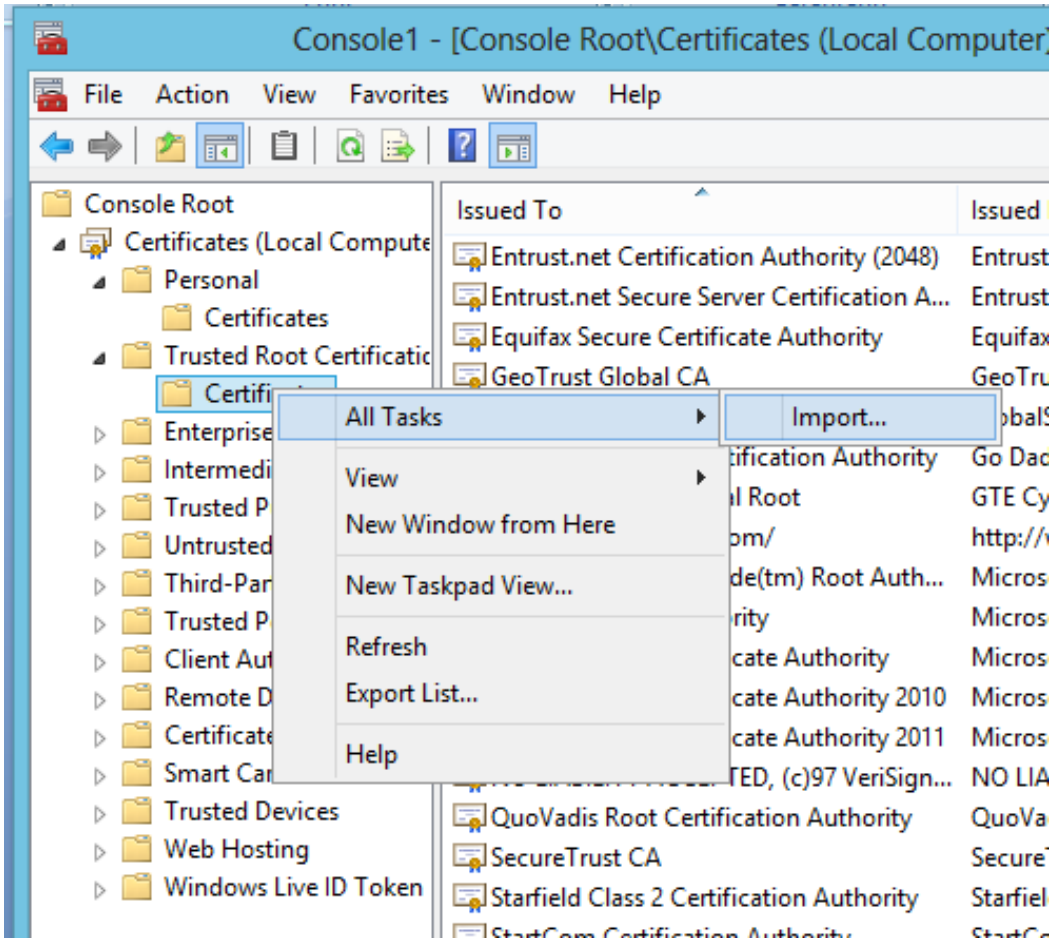
1. Open mmc (Win+R key, type mmc, press enter)
2. File -> Add/Remove Snap-in
3. Double click Certificates, choose Computer Account, Choose Local Computer, Click Finish
4. Import the certificates in Trusted root and personal stores
5. Given permission for IIS to access the private key in personal store
6. Close the mmc console

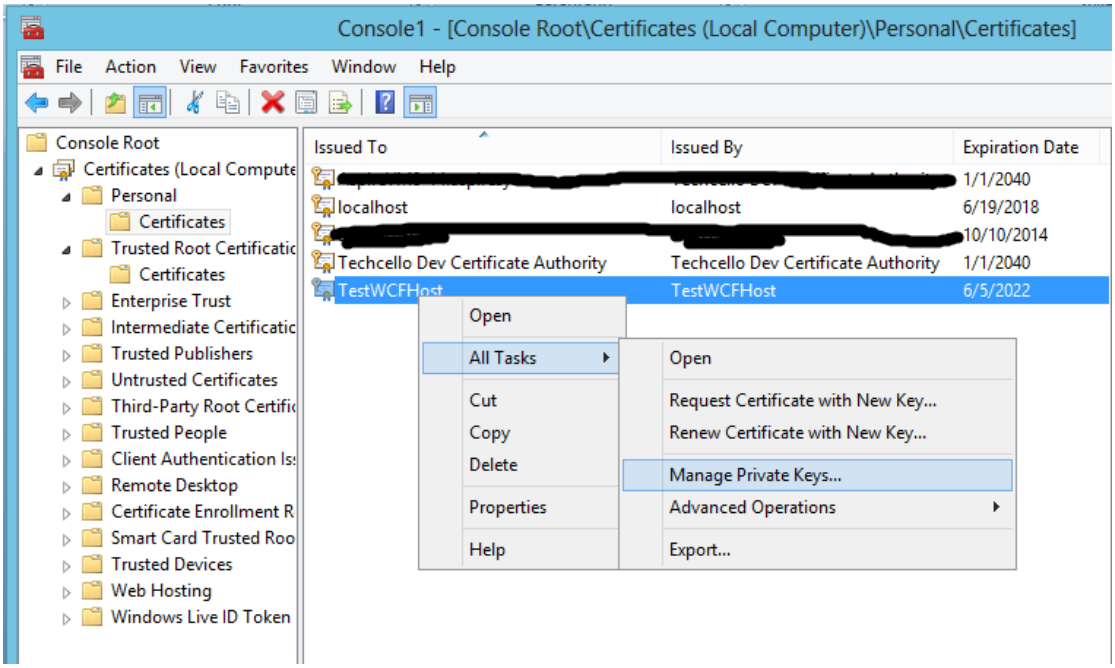

Console1 - [Console Root]
File
Action
View
Favorites
Window
Help
New Ctrl+N
Open... Ctrl+O
Save Ctrl+S
Save As...
Add/Remove Snap-in... Ctrl+M
Options...
1 C:\Users\...\certs.msc
2 services.msc
3 eventvwr.msc
4 comexp.msc
Exit
There are no items to show in this view.
Actions
Console Root
More Actions

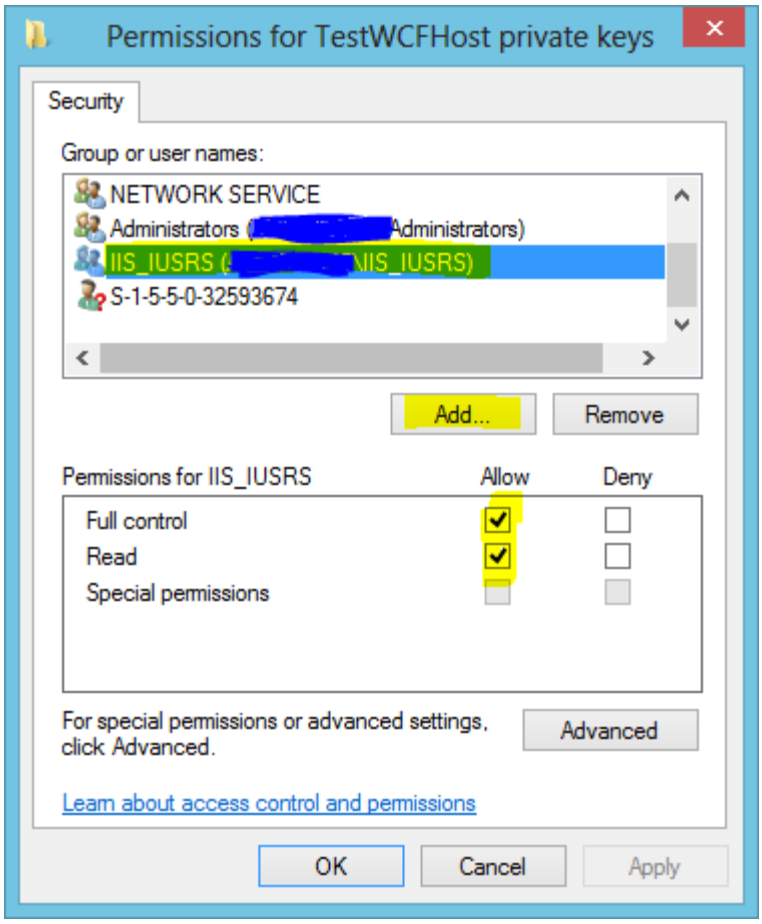

Add or Remove Snap-ins
You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.
Available snap-ins:
Snap-in
Vendor
ActiveX Control Microsoft Cor...
Authorization Manager Microsoft Cor...
Certificates Microsoft Cor...
Component Services Microsoft Cor...
Computer Managem... Microsoft Cor...
Device Manager Microsoft Cor...
Disk Management Microsoft and...
Event Viewer Microsoft Cor...
Folder Microsoft Cor...
Group Policy Object ... Microsoft Cor...
Internet Informatio... Microsoft Cor...
IP Security Monitor Microsoft Cor...
IP Security Policy M... Microsoft Cor...
Add >
Selected snap-ins:
Console Root
Edit Extensions...
Remove
Move Up
Move Down
Advanced...
Description:
The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.
OK
Cancel


Certificates snap-in
This snap-in will always manage certificates for:
My user account
Service account
Computer account
< Back
Next >
Cancel


Select Computer
Select the computer you want this snap-in to manage.
This snap-in will always manage:
Local computer: (the computer this console is running on)
Another computer:
Browse...
Allow the selected computer to be changed when launching from the command line. This only applies if you save the console.
< Back
Finish
Cancel


Console1 - [Console Root\Certificates (Local Computer)]
File Action View Favorites Window Help
Console Root
Certificates (Local Compute
Personal
Certificates
Trusted Root Certificatic
Enterprise
Intermedi
Trusted P
Untrusted
Third-Par
Trusted P
Client Aut
Remote D
Certificate
Smart Car
Trusted Devices
Web Hosting
Windows Live ID Token
Issued To
Issued
Entrust.net Certification Authority (2048)
Entrust.net Secure Server Certification A...
Equifax Secure Certificate Authority
GeoTrust Global CA
All Tasks
Import...
View
New Window from Here
New Taskpad View...
Refresh
Export List...
Help
QuoVadis Root Certification Authority
SecureTrust CA
Starfield Class 2 Certification Authority


Console1 - [Console Root\Certificates (Local Computer)\Personal\Certificates]
File Action View Favorites Window Help
Console Root
Certificates (Local Compute
Personal
Certificates
Trusted Root Certificatic
Certificates
Enterprise Trust
Intermediate Certificatic
Trusted Publishers
Untrusted Certificates
Third-Party Root Certific
Trusted People
Client Authentication Iss
Remote Desktop
Certificate Enrollment R
Smart Card Trusted Roo
Trusted Devices
Web Hosting
Windows Live ID Token
Issued To
Issued By
Expiration Date
1/1/2040
localhost
localhost
6/19/2018
10/10/2014
Techcello Dev Certificate Authority
Techcello Dev Certificate Authority
1/1/2040
TestWCFHost
TestWCFHost
6/5/2022
Open
All Tasks
Cut
Copy
Delete
Properties
Help
Open
Request Certificate with New Key...
Renew Certificate with New Key...
Manage Private Keys...
Advanced Operations
Export...



Navigate to ***web.config*** file and ensure/change the following line matches the new certificate details are updated.

```
<serviceCertificate findValue="TestWcfHost" storeLocation="LocalMachine" storeName="My"
x509FindType="FindBySubjectName"/>
```

Change the binding to match the which security mode you wish to use

Message Security:

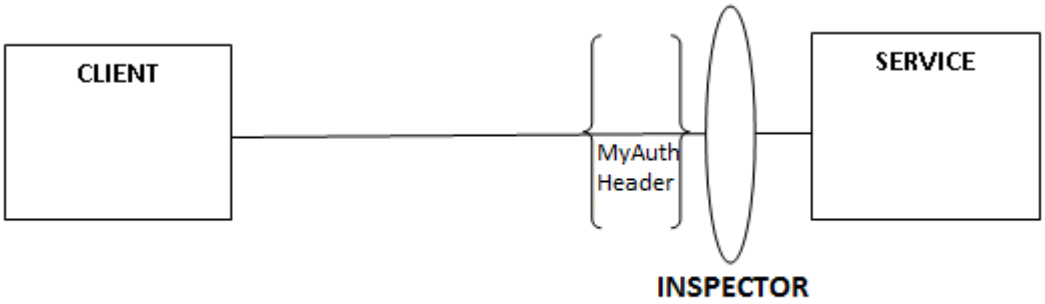
```
<wsHttpBinding>
  <binding name="userAuthetication" maxBufferPoolSize="2147483647"
maxReceivedMessageSize="2147483647">
    <readerQuotas maxDepth="2147483647" maxStringContentLength="2147483647"
maxArrayLength="2147483647" maxBytesPerRead="2147483647" maxNameTableCharCount="2147483647"/>
    <security mode="Message">
      <message clientCredentialType="UserName" establishSecurityContext="false"
negotiateServiceCredential="true"/>
      <transport clientCredentialType="None"/>
    </security>
  </binding>
</wsHttpBinding>
```

Transport with Message Credential Security:

```
<wsHttpBinding>
  <binding name="userAuthetication" maxBufferPoolSize="2147483647"
maxReceivedMessageSize="2147483647">
    <readerQuotas maxDepth="2147483647" maxStringContentLength="2147483647"
maxArrayLength="2147483647" maxBytesPerRead="2147483647" maxNameTableCharCount="2147483647"/>
    <security mode="TransportWithMessageCredential">
      <message clientCredentialType="UserName" establishSecurityContext="false"
negotiateServiceCredential="true"/>
      <transport clientCredentialType="None"/>
    </security>
  </binding>
</wsHttpBinding>
```

2.5 Introduction to Message Inspector

CelloSaaS utilizes Message Inspector extension of WCF to inject Security Context on the Web Service requests and responses. The Message inspectors will automatically append the Roles Privilege required to access the particular service as well as the Page Level privileges to access the given service.



1. Functional Security
2. Page Level Security
3. Metering usage

```
<behaviorExtensions>
        <add name="messageInspector"
type="CelloSaaS.Services.WCF.LoggingBehaviorExtensionElement, CelloSaaS.Services"/>
        <add name="meteringInspector" type="CelloSaaS.Services.WCF.MeteringExtensionElement,
CelloSaaS.Services"/>
        <add name="securityInspector"
type="CelloSaaS.Services.WCF.BLSecurityExtensionElement, CelloSaaS.Services"/>
        <add name="policyInjectionInspector"
type="CelloSaaS.Services.WCF.PolicyInjectionExtensionElement, CelloSaaS.Services"/>
      </behaviorExtensions>
```

While enabling WCF Services for the application related functions you may follow the standard mechanism of decorating attributes such as

1. Operation Contract
2. Service Contract
3. Data Contract

Cello WCF Message Inspector can be used to intercept and inspect the messages coming in or going out of the service and append the Security Context on top of the service headers. This will help developers to avoid repeated Security Validations such as Check User/Role Privilege Validation etc.

Scenario:

While building an Employee Management Solution, there might be a Service which list out all the Employees information, this method might have to be restricted based on certain Privileges, similarly you might develop 100s of methods, So as a developer, you have to validate whether a service can be allowed to access or not.

```
[OperationContract]
EmployeeDetails GetEmployees();
```

2.6 How to Configure Message Inspector

Message Inspectors can be simply configured in the Configuration File i.e. web.config which is a one time activity.

2.6.1 Via Configuration

Open the ***Web.Config*** file under the Web Services Project or Folder and search for the below configuration Snippet. By default, the Message Inspector is ON and mandatory, in case if you don't want to enable this feature simply comment this out.

```
<extensions>
      <behaviorExtensions>
        <add name="messageInspector"
type="CelloSaaS.Services.WCF.LoggingBehaviorExtensionElement, CelloSaaS.Services"/>
        <add name="meteringInspector" type="CelloSaaS.Services.WCF.MeteringExtensionElement,
CelloSaaS.Services"/>
        <add name="securityInspector"
type="CelloSaaS.Services.WCF.BLSecurityExtensionElement, CelloSaaS.Services"/>
        <add name="policyInjectionInspector"
type="CelloSaaS.Services.WCF.PolicyInjectionExtensionElement, CelloSaaS.Services"/>
      </behaviorExtensions>
    </extensions>
```

```
    <behaviors>
      <endpointBehaviors>
        <behavior name="messageInspectorBehavior">
          <messageInspector/>
          <meteringInspector/>
          <securityInspector/>
          <!--<policyInjectionInspector/>-->
        </behavior>
      </endpointBehaviors>
      <serviceBehaviors>
        <behavior name="ServiceBehavior">
          <serviceMetadata httpGetEnabled="true" httpsGetEnabled="false"/>
          <serviceCredentials>
            <clientCertificate>
              <authentication certificateValidationMode="None" revocationMode="NoCheck"/>
            </clientCertificate>
            <serviceCertificate findValue="TestWcfHost" storeLocation="LocalMachine"
storeName="My" x509FindType="FindBySubjectName"/>
            <userNameAuthentication userNamePasswordValidationMode="Custom"
customUserNamePasswordValidatorType="CelloSaaS.Services.WCF.SecretKeyClientValidator,
CelloSaaS.Services"/>
            <issuedTokenAuthentication allowUntrustedRsaIssuers="true"/>
          </serviceCredentials>
          <serviceDebug includeExceptionDetailInFaults="true"/>
        </behavior>
        <behavior name="">
          <serviceMetadata httpGetEnabled="true"/>
          <serviceDebug includeExceptionDetailInFaults="false"/>
        </behavior>
        <behavior name="ServiceBehaviourDetails">
          <serviceMetadata httpGetEnabled="true"/>
          <serviceDebug includeExceptionDetailInFaults="true"/>
        </behavior>
        <behavior name="noMetadataConfig">
          <serviceMetadata httpGetEnabled="false"/>
          <serviceDebug includeExceptionDetailInFaults="false"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
```

3 Testing the deployed Service

1. Open a browser and navigate to the Web Service hosted url
2. You will see many directories like AccessControlManagement, UserManagement, etc.,
3. Navigate to any folder and click the .svc file, https://www.domainname.com/UserManagement/UserDetailsService.svc
4. You will see the WSDL in your browser.

4 Creating service client

Read through the link to know how to: Add, Update, or Remove a Service Reference in your application

CelloSaaS WCF Services can be authenticated in two ways. They are

1. Custom username/password authentication.
2. Tenant wise app secret shared key [This secret shared key can be used if the calling application is trusted. Do not use this to expose this for public users to access this.]

4.1 UserName/Password Validation

Make sure the ***Web.Config*** read as below to go with Custom username/password authentication.

```
<userNameAuthentication userNamePasswordValidationMode="Custom"
customUserNamePasswordValidatorType="CelloSaaS.Services.WCF.ClientValidator,
CelloSaaS.Services"/>
```

4.1.1 Sample

```
client.ClientCredentials.UserName.UserName = "company code\username"; (in clear text)
client.ClientCredentials.UserName.Password = "password";
```

4.2 Shared Secret key validation:

Make sure the ***Web.Config*** read as below to go with Tenant wise app secret shared key

```
<userNameAuthentication userNamePasswordValidationMode="Custom"
customUserNamePasswordValidatorType="CelloSaaS.Services.WCF.SecretKeyClientValidator,
CelloSaaS.Services"/>
```

To Read about Tenant Wise Secret Key click here

4.2.1 Sample

```
client.ClientCredentials.UserName.UserName = "tenantId(guid)\username";
client.ClientCredentials.UserName.Password = "encrypted shared-key";
```

Note:

With the Test certificates that are not signed by a valid certification authority, the application will throw the Runtime Error as ("secure communication error") while trying to consume the service in development environment.

To avoid the above error, it is suggested to add the below code snippet to ignore this error.

```
System.Net.ServicePointManager.ServerCertificateValidationCallback += (se, cert, chain, sslerror) =>
        {
            return true;
        };
```

4.3 Sample code for fetching tenant details

```
System.Net.ServicePointManager.ServerCertificateValidationCallback += (se, cert, chain, sslerror) =>
        {
          return true;
        };

List<Tenant> tenantDetails;
        WcfTenantService.TenantServiceClient client=new WcfTenant Service.
TenantServiceClient();
        client.ClientCredentials.UserName.UserName = "company/admin@company.com";
        client.ClientCredentials.UserName.Password = "company";
        tenantDetails = client.GetAllTenantInfoDetails().ToList()
```

4.4 Introduction to CelloSaaS Proxy Layer

Proxy Layer provides surrogates of objects that we are trying to access, in another words, with Proxy Layer, we no need to take care where exactly the actual object is, we just make a call on the surrogate, and the surrogate will talk to the local/remote object.

By hiding details of accessing to other objects, Proxy Layer grants the business application with high flexibility and portability, it also let developer focus on his part and make fewer mistakes.

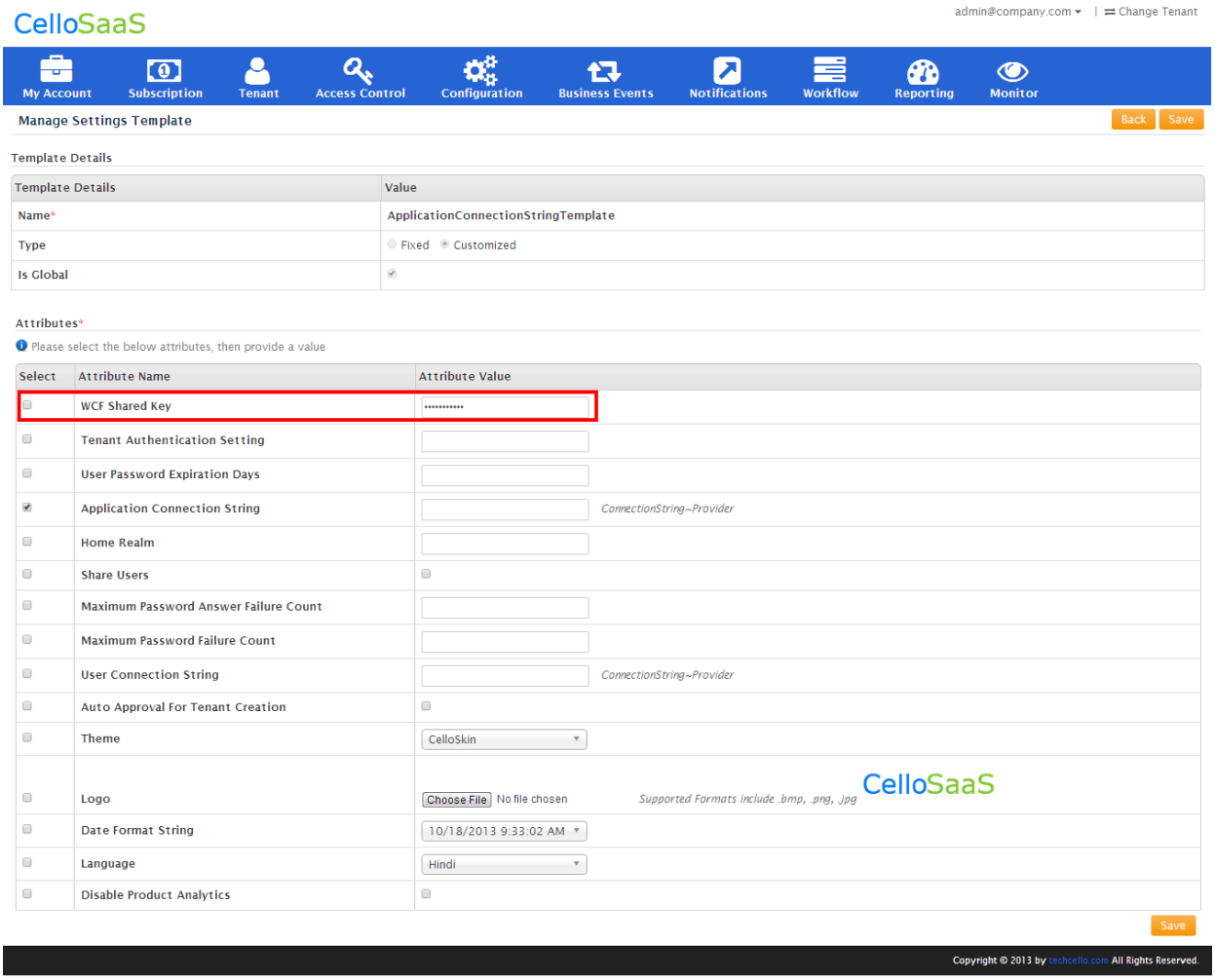
Irrespective of the consumption model i.e. InProc mode or services mode, the application layer need not worry if we follow the Proxy layer.

4.5 Secret key for validating users via services

Apart from authenticating Service calls with Username and Password, Cello also provides an elegant feature to authenticate a group of users of a tenant against a secret key. With this authentication, we can simply replace the regular username/password with the configured Secret key by Tenant.

This is a tenant wise setting of the secret key. The tenant can set a WCF shared key in the tenant setting page. This key will be used to authenticate CelloSaaS WCF service instead of sending password in user credential.

Since this is a secret key, the input is provided via the password field in the Tenant Settings view page.



5 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