VPass client

VPass client configuration guide

This document describes the VPass client applications configuration options

14. 12. 2014

Table of Contents

Client applications 2
configSections 2
connectionStrings 2
vpassClientCoreConfigurationSection 2
exhibitCommunicationConfiguration 4
vpassClientConfigurationSection 4
entityFramework 8
appSettings 8
startup 9
system.net 9
system.serviceModel 9
runtime 11
system.data 12
system.diagnostics 12
VPass Client modules 13
Startup Modules 13
Client Modules (Core Modules) 14

Client applications

Currently there are two client applications. Each is for different purpose. One application is with UI and the other one is Windows service for the 3rd party UI application to enable communication with the VPass server and Exhibit driver. Each section will have specification if it applies to both modes or if it's specific to only one application.

configSections

Valid for both client applications.

```
<configSections>
    <section name="vpassClientCoreConfigurationSection"
type="VPass.Client.Core.Configuration.VPassClientCoreConfigurationSection,
VPass.Client.Core" />
    <section name="exhibitCommunicationConfiguration"
type="VPass.Client.Communication.Configuration.ExhibitCommunicationConfiguration,
VPass.Client.Communication" />
        <section name="entityFramework"
type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework,
Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
requirePermission="false" />
  </configSections>
```

Example above is common for both client applications. This section may reqire additional section registration depending on used modules of the application.

UI client specific section:

```
<section name="vpassClientConfigurationSection"
type="VPass.Client.Support.Configuration.VPassClientConfigurationSection,
VPass.Client" />
```

This section is required for the UI client.

connectionStrings

Standard section. Configures connection string for the entity Framework used in the offline store.

```
<connectionStrings>
    <add name="OfflineStoreDBEntities"
connectionString="metadata=res://*/ServerCommunication.OfflineStore.OfflineStoreDBEnti
ties.csdl|res://*/ServerCommunication.OfflineStore.OfflineStoreDBEntities.ssdl|res://*
/ServerCommunication.OfflineStore.OfflineStoreDBEntities.msl;provider=System.Data.SqlS
erverCe.4.0;provider connection string=&quot;data
source=|DataDirectory|\Data\OfflineStoreDB.sdf&quot;"
providerName="System.Data.EntityClient" />
  </connectionStrings>
```

vpassClientCoreConfigurationSection

Custom section. This section is common for both client applications.

Section setups the basic parameters of the core context of the client applications and registers the startup modules and client modules. See the Developer documentation on haw to create modules for the client.

Key	Value	Note
exhibitId	Integer	Setups the exhibit id from the VPass web configuration. This value is setup automatically with the activation of the client.
maxUser	Integer	Maximum of logged user. Default I s1.
minUser	Integer	Minimum of logged user. Default is 1.
offlineStore	Seub element	Contains information for the offline message store. Currently empty.
Application	Subelement	Contains information for the attached application. Setup automatic.
startupModules	Element collection	Contains list of the startup modules.
Startup module element	Collection element	Use add keyword to add new element
Name	String, unique in collection	Name of the element. Free text. Have to be unique in the colletion.
moduleType	String, fully qualified class name	Fully qualified startup module class name.
coreModules	Element collection	Contains list of the Client modules registered for the client.
Core modules element	Collection element	Use add keyword to add new element.
Name	String, unique in collection	Name of the module, Free text. Has to unique in the collection.

Example:

```
<vpassClientCoreConfigurationSection exhibitId="1" maxUsers="1" minUsers="1" >
    <offlineStore storeConnectionStringName="" />
    <application appType="HTML" path="http://localhost/131" />
    <startupModules>
      <add Name="config"
moduleType="VPass.Client.Core.Modules.StartupConfigurationModule, VPass.Client.Core"
/>
      <add Name="browser"
moduleType="VPass.Client.Support.Modules.BrowserCompatibilityStartupModule,
VPass.Client"/>
    </startupModules>
    <coreModules>
      <add Name="clientConfig"
moduleType="VPass.Client.Support.Modules.ConfigurationCoreModule, VPass.Client" />
      <add Name="cleanUpModule"
moduleType="VPass.Client.Core.Modules.TempProfileCleanupModule, VPass.Client.Core" />
      <add Name="exhibitLogin"
moduleType="VPass.Client.Support.Modules.ExhibitLoginModule, VPass.Client" />
```

```
    </coreModules>
```

For the list of modules please refer to the module section.

exhibitCommunicationConfiguration

Custom section. This section is common for both client applications. Setup automatic.

This section registers the communication manager for the exhibit driver.

Key	Value	Note
communicationManagerType	String, Full qualified class name	Sets up how the client will communicate with the exhibit driver. Auto setup with activation.

Exhibit driver list

VPass contains several possible drivers. But new driver can be created and added to the system. Please refer to the Developer documentation for details.

Type	Note
VPass.Client.Core.NoExhibitDriver, VPass.Client.Core	Use when the client is not connected to any driver or application.
VPass.Client.Communication.TcpExhibitCommunication, VPass.Client.Communication	Use for communication over TCP protocol
VPass.Client.Communication.Wcf.WcfExhibitCommunication, VPass.Client.Communication.Wcf	User for communication via WCF.
VPass.Client.Support.ExhibitCommunication. WpfExhibitAdvancedCommandsDriver, VPass.Client	Use only in UI client. Use for communication with external application over WCF.
VPass.Client.Support.ExhibitCommunication. TcpExhibitAdvancedCommandsDriver, VPass.Client	Use only in UI client. Use for communication with external application over TCP.

vpassClientConfigurationSection

Custom section. This section is for UI application only. Setup partially automatic.

Key	Value	Note
commonCMSContentKey	String	Code value from the CMS content list in the web application. Automatic setup.
defaultCategoryCode	String	Code value from the visitor category list in the web app. Automatic setup.
dataDirectory	String	Name of the directory that contains XML data files
imagesDirectory	String	Name of the directory for the images.

ticketRegex	Regular expression	Agaist this expression are evaluated the codes from the tickets.
theme	String, fully qualified component name	Full path to theme file. Automatic setup. Value is from the theme list in the web app.
homeAction	String	Name of the cation from the menuActions list in the configuration. This action is performed when when default screen is needed.
qrCameraIndex	Integer	Index of camera for the QR reader. Default 0.

applicationUI

Subelement for the 3rd party application setup.

Key	Value	Note
top	integer	Distance in pixels from the top of the screen for the application.
left	integer	Distance in pixels from the left of the screen for the application
width	integer	Width of the application in the pixels.
height	integer	Height of the application in the pixels.

actvityDemo

Subelement. Sets the time of the inactivity for automatic logout and the settings for the demo module.

Key	Value	Note
secondsOfInactivity	integer	number of seconds after the user is logged out and demo module is started.
demoModuleType	string, fully qualified class name	the type of the demo module to be started after the inactivity.

social

Defines the values for connection to social networks and which networks are allowed from the VPass client.

Key	Value	Note
isActive	Boolean	true/false if the sharing is allowed in general. Auto setup.
facebookShareAllowed	Boolean	true/false if the Facebook

		share is allowed Auto setup.
twitterShareAllowed	Boolean	true/false if the twitter share is allowed Auto setup.
emailShareAllowed	Boolean	true/false if the share via email is allowed. Auto setup.
multimediaCommonMessageResourceCode	string	Resource string code for the message that is posted with the image/video. Auto setup.
facebook	sub element	contains values for connection to the Facebook. Auto setup.
key	string	Key from the Facebook app. Obtained from the Facebook. Auto setup.
secret	string	Secret value. Obtained from Facebook. Auto setup.
twitter	sub element	Contains values for connection to the twitter. Auto setup.
key	string	Key from the twitter. Auto setup.
secret	string	Secret value from the twitter. Auto setup.

profile

Subelement. Contains configuration of the profile screen.

Key	Value	Note
isActive	Boolean	true/false if the profile is allowed. Auto setup.
allowActualVisitSection	Boolean	true/false. If the visit recapitulation in the profile should be visible. Auto setup.
allowImageGallery	boolean	true/false. If the image gallery section should be visible in the profile.
allowLastVisitSection	boolean	true/false. If the last visit section should be visible on the profile page.
allowVideoGalery	boolean	tre/false. If the video gallery should be visible on the profile page.

hallOfFame

Subelelemnt. Sets up the hall of fame properties.

Key	Value	Note
isActive	Boolean	true/false if the hall of fame is active on this exhibit.
exhibitResultMessageFormatResourceCode	string	Resource code for the message for sharing.

menuActions

Collection of the menu actions of the VPass client. Does not auto configure on activation.

This defines what actions will appear on the right side on the client UI. **Name in the collection is important** as it defines the style and the CMS content value. Action is added to the collection by add keyword.

Some of the actions can appear on the screen based on the client context status (user is logged in, is online/offline etc.).

Key		
Name	string	name of the action. Style and text is defined by the value in the name. Has to unique in the collection. For CMS action name has to contain the CMS content code.
type	string	fully qualified name of the menu action class.

Example:

```
<add Name="info" type="VPass.Client.Support.Menu.CMSContentAction, VPass.Client" />
```

languages

Subelement. Contains information and configuration for the language selector of the VPass client and defines default language. Does not auto configure on activation. Please note that the Resource file have contain resources for all the defined languages.

Key	Value	Note
systemLanguagePath	string, path	Path to the XML file with string resources that contains system text
defaultCulture	string, culture code	contains default culture code/name. For all codes please refer to the MSDN.
languageItems	element collection	contains list of available languages for the client

language element	element	use add keyword to add new element.
Name	string	Name of the Language as it will appear on the selection screen. Has to be unique in the collection.
Culture	string, culture code	Contains culture code/name. For all codes please refer to the MSDN.

Example:

```
<languages systemLanguagePath="Data\SystemLanguages.xml" defaultCulture="cs-CZ">
      <languageItems>
        <add Name="Česky" Culture="cs-CZ" />
        <add Name="English" Culture="en-GB" />
        <add Name="German" Culture="de-DE" />
      </languageItems>
    </languages>
```

entityFramework

Entity Framework section. For detailed description please refer to the Entity framework documentation. Should not be changed. Valid for both client applications.

```
<entityFramework>
    <defaultConnectionFactory
type="System.Data.Entity.Infrastructure.SqlCeConnectionFactory, EntityFramework">
      <parameters>
        <parameter value="System.Data.SqlServerCe.4.0" />
      </parameters>
    </defaultConnectionFactory>
    <providers>
      <provider invariantName="System.Data.SqlServerCe.4.0"
type="System.Data.Entity.SqlServerCompact.SqlCeProviderServices,
EntityFramework.SqlServerCompact" />
    </providers>
  </entityFramework>
```

appSettings

Standard section. For details refer to the MSDN documentation.

Section can contain other keys for the specific modules used in the client, this is described in each module.

Key	Value	Note
Ehibit_LogoPath	string, path	Path to the image/icon displayed in the Hall of fame.
Ehibit_Name	string	Name of the exhibit that is displayed in the Hall of fame.

startup

Standard section. Do not change.

Example:

```
<startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>
```

system.net

Standard section. For details refer to the MSDN documentation. Contains configuration fro the SMTP server used for email sharing. If email sharing is not required this section can be removed. Values have to be obtained from the administrator of the SMTP server.

Key	Value	Note
from	string	Email that will appear as from in the send email.
host	string	Name of the SMTP server e.g. smtp.google.com
username	string	username that is passed to the SMTP.
password	string	Password for the user passed to the SMTP.
port	integer	Stadard value is 25.

Example:

```
<system.net>
    <mailSettings>
      <smtp from="[email address of the sender]">
        <network host="[smtp server]" userName="[user name for SMTP]"
password="[password for SMTP]" port="25" />
      </smtp>
    </mailSettings>
  </system.net>
```

system.serviceModel

Standart section. For details refer to the MSDN documentation of the WCF. Should not be changed. Contains configuration for the WCF communication with the server and with the exhibit driver via WCF. Valid for both client applications.

Setup these values only:

Client endpoints

- **Name = "ExhibitDriver"** – configures the WCF communication for the Exhibit driver, VPass client side. Can be changed as needed as long as the exhibit driver uses the same communication setup.
- **Name =" httpExhibitDriver"** – configures the secondary exhibit driver communications. Used by secondaryDriverModule.

- **Name=" BasicHttpBinding_IVisitorService"** – replace only vpass.local to IP address or name of your VPass server in the **address** value.
- **Name=" BasicHttpBinding_IClientSystemService" -** replace only vpass.local to IP address or name of your VPass server in the **address** value.

Services:

- **name="VPass.Client.Communication.Wcf.Service" -** configures the WCF communication for the Exhibit driver, VPass client side. Can be changed as needed as long as the exhibit driver uses the same communication setup.
- **name="VPass.Client.Communication.Wcf.SecondaryService"**– configures the secondary exhibit driver communications. Used by secondaryDriverModule.
- **name="VPass.Client.Core.ServerCommunication.Services.SystemService" – do not change** unless you change the sever setup as well. Has to be the same for all the clients
- **name="VPass.Client.Core.ServerCommunication.Services.MultimediaDownloadService" – do not change**

Behaviors and Bindings setup can be changed to modify the existing endpoints and service behavior. Any other changed are not recommended without consulting the Development team and without changing the server configuration as well.

Example:

```
<system.serviceModel>
    <standardEndpoints>
      <webHttpEndpoint>
        <standardEndpoint name="NewStandardEndpoint0" />
      </webHttpEndpoint>
    </standardEndpoints>
    <behaviors>
      <serviceBehaviors>
        <behavior name="NewBehavior0" />
        <behavior name="debug">
          <serviceDebug includeExceptionDetailInFaults="true" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <bindings>
      <netTcpBinding>
        <binding name="NewBinding0" receiveTimeout="01:20:00" sendTimeout="00:20:00"
transactionFlow="false" maxReceivedMessageSize="2000000000" />
      </netTcpBinding>
      <basicHttpBinding>
        <binding name="BasicHttpBinding_IVisitorService"
maxReceivedMessageSize="2000000000" />
        <binding name="BasicHttpBinding_IClientSystemService"
maxBufferPoolSize="2147483647" maxBufferSize="2147483647"
maxReceivedMessageSize="2147483647" messageEncoding="Text">
          <readerQuotas maxDepth="2000000" maxStringContentLength="2147483647"
maxArrayLength="2147483647" maxBytesPerRead="2147483647"
maxNameTableCharCount="2147483647" />
        </binding>
        <binding name="HttpStreaming" maxReceivedMessageSize="67108864"
transferMode="Streamed" />
```

```
        <binding name="baseHttp" maxReceivedMessageSize="2000000000"/>
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint address="net.tcp://localhost:50124" binding="netTcpBinding"
bindingConfiguration="" contract="ExhibitService.IExhibitService" name="ExhibitDriver"
kind="" endpointConfiguration="" />
      <endpoint
address="http://localhost/VPass.Tools.WebExhibitDriver/ExhibitService.svc"
binding="basicHttpBinding"
                bindingConfiguration="baseHttp"
contract="ExhibitService.IExhibitService"
                name="httpExhibitDriver" />
      <endpoint address="http://vpass.local/Services/VisitorService.svc"
binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_IVisitorService"
contract="VisitorServerService.IVisitorService"
name="BasicHttpBinding_IVisitorService" />
      <endpoint address="http://vpass.local/Services/ClientSystemService.svc"
binding="basicHttpBinding"
bindingConfiguration="BasicHttpBinding_IClientSystemService"
contract="ServerSystemService.IClientSystemService"
name="BasicHttpBinding_IClientSystemService" />
    </client>
    <services>
      <service behaviorConfiguration="NewBehavior0"
name="VPass.Client.Communication.Wcf.Service">
        <endpoint address="net.tcp://localhost:50123" binding="netTcpBinding"
bindingConfiguration="NewBinding0" contract="VPass.Communication.IControlerService" />
      </service>
      <service behaviorConfiguration="NewBehavior0"
name="VPass.Client.Communication.Wcf.SecondaryService">
        <endpoint address="net.tcp://localhost:50126" binding="netTcpBinding"
bindingConfiguration="NewBinding0" contract="VPass.Communication.IControlerService" />
      </service>
      <service name="VPass.Client.Core.ServerCommunication.Services.SystemService"
behaviorConfiguration="debug">
        <endpoint address="http://localhost/vpasclient/systemservice.svc"
binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_IVisitorService"
                  name="HttpSystemService"
contract="VPass.Client.Core.ServerCommunication.Services.ISystemService" />

      </service>
      <service
name="VPass.Client.Core.ServerCommunication.Services.MultimediaDownloadService"
behaviorConfiguration="debug">
        <endpoint address="http://localhost/vpasclient/MultimediaDownloadService.svc"
binding="basicHttpBinding" bindingConfiguration="HttpStreaming"
name="HttpMultimediamService"
contract="VPass.Client.Core.ServerCommunication.Services.IMultimediaDownloadService"
/>
      </service>
    </services>
```

runtime

Standard section. **Do not change**. For details refer to the MSDN documentation.

Example:

```
<runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed"
culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-6.0.0.0" newVersion="6.0.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
```

system.data

Standard section. **Do not change**. For details refer to the MSDN documentation.

Used for configuration connection to the offline store by entity framework.

Example:

```
<system.data>
    <DbProviderFactories>
      <remove invariant="System.Data.SqlServerCe.4.0" />
      <add name="Microsoft SQL Server Compact Data Provider 4.0"
invariant="System.Data.SqlServerCe.4.0" description=".NET Framework Data Provider for
Microsoft SQL Server Compact" type="System.Data.SqlServerCe.SqlCeProviderFactory,
System.Data.SqlServerCe, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=89845dcd8080cc91" />
    </DbProviderFactories>
  </system.data>
```

system.diagnostics

Standard section. For details refer to the MSDN documentation. Can be changed as need.

Example:

```
<system.diagnostics>
    <sources>
      <source name="Information" switchName="SourceSwitch"
switchType="System.Diagnostics.SourceSwitch">
        <listeners>
          <add name="myListener" />
          <remove name="Default" />
        </listeners>
      </source>
      <source name="Exceptions" switchName="SourceSwitch"
switchType="System.Diagnostics.SourceSwitch">
        <listeners>
          <add name="myListener" />
          <remove name="Default" />
        </listeners>
      </source>
    </sources>
    <sharedListeners>
      <add name="myListener" type="System.Diagnostics.TextWriterTraceListener"
initializeData="TextWriterOutput.log" />
    </sharedListeners>
```

```
    <trace autoflush="true" indentsize="4">
      <listeners>
        <add name="myListener" />
      </listeners>
    </trace>
  </system.diagnostics>
```

VPass Client modules

VPass client application can contains any number of the Startup or Client modules to extend the VPass client functionality. This section contains list of the modules and their additional configuration if needed. Both client application contain both type of modules but only certain modules make sense for each type of the application. For example the Windows service mode dos not need the modules that interacts with the UI components.

Startup Modules

Startup module is executed only once at the startup of the client application

Name	Type
ExhibitStartupModule	VPass.Client.Core.Modules.ExhibitStartupModule, VPass.Client.Core
StartupConfigurationModule	VPass.Client.Core.Modules.StartupConfigurationModule, VPass.Client.Core
BrowserCompatibilityStartupModule	VPass.Client.Support.Modules.BrowserCompatibilityStartupModule, VPass.Client

ExhibitStartupModule

If the application type is EXE it will start the specified application by its path in the configuration.

StartupConfigurationModule

on the start of the client check if the **exhibitId** is setup and if the exhibit is active on the server. If it is not configured and the exhibit is active on the sever it downloads the configuration and configures the client.

If exhibit is not active on the server it will switch the client to maintenance mode.

If **exhibitId** is not configured and there is no active Exhibit for the IP address of the client computer the client is switched to maintenance mode.

BrowserCompatibilityStartupModule

If the **HTML** application is used e.g. HTML content is hosted inside the Client application this module configures the compatibility mode of the used WebBrowser component. WebBrowser component is set as default to IE7 compatibility mode.

Module needs this key in the **appSettings** section:

Key	Value	Note

browserCompatibilityMode	integer	browser compatibility mode number. Use 11001 for IE11 or 10000 for IE10 compatibility

Client Modules (Core Modules)

Client(or Core) module is created on the application startup and its disposed on the application end. It can execute and reacts to core or UI events or add other communication possibilities and otherwise extend the client functionality.

Name	Type
ExhibitStartupModule	VPass.Client.Core.Modules.ExhibitStartupModule, VPass.Client.Core
MultimediaModule	VPass.Client.Core.Modules.MultimediaModule ,VPass.Client.Core
TempProfileCleanupModule	VPass.Client.Core.Modules.TempProfileCleanupModule ,VPass.Client.Core
ConfigurationCoreModule	VPass.Client.Core.Modules.ConfigurationCoreModule ,VPass.Client.Core
ExhibitLoginModule	VPass.Client.Support.Modules.ExhibitLoginModule ,VPass.Client
SecondaryExhibitDriverModule	VPass.Client.Support.Modules.SecondaryExhibitDriverModule ,VPass.Client

ExhibitStartupModule

If registered as Core module it reacts on configuration change in the application section and after the configuration is set it restarts the connected application if the path has changed.

MultimediaModule

When the application data of type Video is received this module copies the video to its own directory under its own name and modifies the video path to the new location.

TempProfileCleanupModule

When the Visitor logs in on the client the profile data as images are downloaded to the Temp directory in the client. This module reacts on the close of the application and deletes the directory with the Temp data of the Visitors.

ConfigurationCoreModule

This module I sprat of the UI application and react on the Configuration recived event of the Core Context and saves and configures the UI client section.

ExhibitLoginModule

When the application data are not displayed and used by CMS pages with the Save to server button, this module reacts on application data received event of the Core Context and replaces the Save button action. The reaction is only on data collection that contains code "**SaveToProfile**". If the data collection contains the "**TicketCode**" the rest of the data is directly saved to server. Otherwise the login screen is displayed.

The module will react on data code "**GetVisitor**". On this request it will display the login screen reads the code and load the profile from the server (online status, otherwise is empty) and returns the Visitor profile data to the application. This will not cause visitor to login in the client and no login log for the visitor is created.

SecondaryExhibitDriverModule

This module can be used when the client needs to communicate with more than one driver. This can happen when the client is connected to the hardware and 3rd party application. This module uses only WCF communication.

This module can have its own submodules for data modification. If there is no special code in the received application data it assumes that the data are required to save to profile and reacts the same way as the **ExhibitLoginModule**.

If "**GetPicture**" code is received the along with the module snaps the picture from the specified camera and saves the picture to received path. Needed codes along with the GetPicture:

- **GetPictcture** – required. Otherwise it will be saved to profile.
- **Path** – required. If not present the picture will not be taken.
- **CamIndex** – optional. Specifies the index of the camera to use. Default value is 0.
- **Width** – optional. Width of saved picture in pixels. But the real resolution depends on the camera settings.
- **Height** – optional. Height of saved picture in pixels. But the real resolution depends on the camera settings.

This module can be extend with the data modification to enable the same modifications as are possible for the primary driver data. There are some restrictions – only one data modification class can be present and the modification cannot stop the save to profile workflow.

Configuration of the module

As this module is most complex it requires great deal of configuration.

configSections:

- Register the **secondaryExhibitDriverModuleConfigurationSection**

```
<section name="secondaryExhibitDriverModuleConfigurationSection"
type="VPass.Client.Support.Modules.SecondaryExhibitDriverModuleConfigurationSection
, VPass.Client"/>
```

secondaryExhibitDriverModuleConfigurationSection:

create the configuration section in the config file. This section can be emty as it contains only one key that is optional, but it has to be present.

Key	Value	Note
appDataModificationType	string, fully qualified class name	type of the modification, optional.

Example:

```
<secondaryExhibitDriverModuleConfigurationSection
appDataModificationType="VPass.Client.Support.Modules.UniversalHttpImageDataModificati
on ,VPass.Client"/>
```

or

```
<secondaryExhibitDriverModuleConfigurationSection />
```

system.serviceModel:

- ↳ Endpoint – add endpoint with name of **httpExhibitDriver** – see the system.serviceModel configuration chapter.

```
<endpoint address="http://localhost/VPass.Tools.WebExhibitDriver/ExhibitService.svc"
binding="basicHttpBinding" bindingConfiguration="baseHttp"
contract="ExhibitService.IExhibitService" name="httpExhibitDriver" />
```

- ↳ Service – add the secondary service configuration

```
<service behaviorConfiguration="NewBehavior0"
name="VPass.Client.Communication.Wcf.SecondaryService">
        <endpoint address="net.tcp://localhost:50126" binding="netTcpBinding"
bindingConfiguration="NewBinding0" contract="VPass.Communication.IControlerService" />
   </service>
```

Make sure that the application driver uses the same setup as the secondary driver.