SMS Gateway Integration Technical Document

Version 1.0
December 2015
# Table of Contents

- Introduction.......................................................................................................................... 3
- Purpose ..................................................................................................................................... 3
- Scope ....................................................................................................................................... 3
- Sample Code .......................................................................................................................... 3
- What are SMS Gateway? ........................................................................................................ 3
- Integration of SMS gateway in iVend Retail ........................................................................ 4
- Add a new SMS gateway ....................................................................................................... 4
- Checklist ............................................................................................................................... 4
- Steps for creating a SMS gateway ......................................................................................... 4
- SMS deployment and Test ..................................................................................................... 12
Introduction
iVend Retail provide a functionality to send sms to customers. We have integrated six sms gateways with the product. If a retail wish to use any other gateway he can integrate the same.

Purpose
This document is intended for partners who are willing to undertake source code modification under license from CitiXsys to pursue customizations should be reading this document.

This document provides a complete overview of SMS gateway integration, their integration methodology in iVend Retail as well as guide to implementing a new SMS gateway.

Scope
The scope of this document is to highlights the changes required in the SMS gateway integration. The following systems are to be covered:

- Management Console
- Scheduler Service
- Loyalty Customer Portal

Sample Code
Refer to the attached code for adding a new SMS gateway to iVend Retail.

What are SMS Gateway?
An SMS gateway allows a computer to send or receive Short Message Service (SMS) transmissions to or from a telecommunications network. Most messages are eventually routed into the mobile phone networks. Many SMS gateways support media conversion from email and other formats.
Integration of SMS gateway in iVend Retail

- iVend is completely written in Microsoft .NET
- The SMS gateway design and implementation is modular.
- Each SMS gateway has its own API (Application Programming Interface)
- Send SMS using the request.
- Receive a Response and check the Response codes to ensure that you sent the request to the Cloud Communications Platform correctly
- It is possible to add a new SMS gateway. However, these are based on a set of specifications.

Add a new SMS gateway

Checklist

- A Visual Studio.NET 2013 license for development
- A DevXpress 15.2.3 (or higher) license for development
- Contractual agreement with the third party SMS gateway provider e.g. www.Nexmo.com
- Test environment (test account) for testing the developed SMS gateway
- Familiarity with Microsoft.NET development language, tools and technologies

Steps for creating a SMS gateway

- Create a new project in Visual Studio 2013.
- Ensure that the Assembly Name and the Default Namespace starts with `CXS.Messaging.<ProviderName>` where `<ProviderName>` is the actual name of the provider e.g. Nexmo.
- Make sure that the settings of the project is similar to the details shown in the sample illustration below:

```
<table>
<thead>
<tr>
<th>Assembly name:</th>
<th>Default namespace:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CXS.Messaging.Nexmo</td>
<td>CXS.Messaging.Nexmo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Target framework:</th>
<th>Output type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>.NET Framework 4</td>
<td>Class Library</td>
</tr>
</tbody>
</table>
```

Create a new project `CXS.Messaging.<ProviderName>`

- Add references to the following iVend Retail libraries from management console installed directory (`C:\Program Files (x86)\CitiXsys\iVend Retail\ManagementConsole`):
• Add references to the following DevXpress libraries:
  - DevExpress.Data.v15.2
  - DevExpress.Utils.v15.2
  - DevExpress.XtraEditors.v15.2
  - DevExpress.XtraVerticalGrid.v15.2
• Ensure that the following system libraries are already referenced:
  - System.Core
  - System.Data
  - System.Drawing
  - System.Windows.Forms
  - System.Xml

• Change the name of the default class from ‘Class1.cs’ to ‘TextLocalMessagingSystem.cs’. Double click the class to open the code editor.
• Ensure that the class includes the following namespaces:

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using CXS.Messaging.Core;
using CXS.Platform.Xml;
```
Inherit the interface ‘IMessagingSystem’ and implement the interface.

```csharp
namespace CXS.Messaging.TextLocal
{
    public class TextLocalMessagingSystem : IMessagingSystem
    {
        ConfigInfo m_ConfigInfo = null;
        public bool Initialize(System.Xml.XmlNode configNode, System.Windows.Forms.Form ownerForm)
        {
            // Initialize logic
        }

        public bool Startup()
        {
            // Startup logic
        }

        public bool Shutdown()
        {
            // Shutdown logic
        }

        public System.Windows.Forms.Control SetupControl
        {
            // Setup Control logic
        }

        public MessagingResult SendSMS(MessagingContext authorizeContext)
        {
            // Send SMS logic
        }

        private MessagingResult ParseResponse(String responseMessage)
        {
            // Parse Response logic
        }
    }
}
```
- Implement the sendSMS method to integrate the SMS gateway.

```csharp
public MessagingResult SendSMS(MessagingContext authorizeContext) {
    MessagingResult sMSResult = new MessagingResult();
    WebClient client = null;
    Stream data = null;
    try {
        client = new WebClient();
        client.QueryString.Add("username", m_ConfigInfo.Username);
        client.QueryString.Add("hash", m_ConfigInfo.APIKey);
        client.QueryString.Add("sender", m_ConfigInfo.From);
        client.QueryString.Add("numbers", authorizeContext.ToPhoneNumber);
        client.QueryString.Add("message", authorizeContext.SMSBody);

        string baseurl = m_ConfigInfo.SecureURL;
        dto = client.OpenRead(baseurl);
        StreamReader reader = new StreamReader(data);
        var response = reader.ReadToEnd();
        dto.Close();
        reader.Close();
        return ParseResponse(response);
    }
    catch (Exception ex) {
        sMSResult.SMSResultType = MessagingResultType.Error;
        sMSResult.Message = Convert.ToString(ex.Message);
        return sMSResult;
    }
}
```

- Include the sample `ConfigInfo.cs` class as it is into the project and change the namespace appropriately. This class is used by the SMS gateway to set required properties for the SMS gateway being implemented. (Note: The properties are defined on the basis of specifications given by the provider of the SMS gateway e.g. Nexmo)

- Include the sample `CXSMessagingTextLocal.global.xml` file in your project. and ensure that the XML file is included as an embedded resource as shown below:

![XML file properties](image)

- The naming convention of the XML file is: `<ClassName>.global.xml` where the class name is the name of the only implementation class in the project that inherits the `IMessagingSystem` interface.
For the sake of this example the this class is named ‘NexmoMessagingSystem’ and hence the name of this XML file is ‘CXSNexmoMessagingSystem.global.xml’

- Under the root element/node ‘String’ there are one header level elements as shown below:

```
<Strings>
  <SetupControl>...
</SetupControl>
</Strings>
```

- The ‘SetupControl’ element is a predefined/fixed name while the other element i.e. ‘NexmoMessagingSystem’ is named exactly as the actual implementation class.

- Child elements under the ‘SetupControl’ element refer to the properties of the user control in the project representing the user interface elements. For example, assuming that the user control has the following fields defined:
The corresponding elements and their values in the ‘SetupControl’ node would be:

```xml
<Strings>
  <$SetupControl$>
    <$m_CategoryRowGeneral$>General</m_CategoryRowGeneral$>
    <$Title$>TextLocal SMS Setup</Title$>
    <$m_RowSecureURL$>Secure URL</m_RowSecureURL$>
    <$m_RowUsername$>Username</m_RowUsername$>
    <$m_RowApiKey$>API Key</m_RowApiKey$>
    <$m_RowFrom$>Long number/Short code</m_RowFrom$>
  </$SetupControl$>
</Strings>
```

You should change the element names and their values as appropriate when adding your own SMS gateway.

- The iVend Retail product embeds the graphical elements of each SMS gateway individually. This means that every SMS gateway must create the user interface part as well along with the code logic of the payment processor.

This control is known throughout the product as ‘SetupControl’ that uses the ‘XtraVerticalGrid’ control of the DevXpress Suite for building the user interface.

The elements mentioned in the ‘CXSMessagingTextLocal.global.xml’ file are mapped exactly to the row editor properties of the DevXpress ‘XtraVerticalGrid’ control as illustrated below.
To build a user interface for the new SMS gateway include the sample ‘SetupControl.cs’ file and ensure that it inherits from the ‘BaseMessagingSetupView class and the ‘IGlobalizeControl’ interface.

```csharp
namespace CXS.Messaging.TextLocal
{
    public partial class SetupControl : BaseMessagingSetupView, IGlobalizableControl
    {
        List<ConfigInfo> m_list = null;

        public SetupControl()...

        protected override void OkButtonClickHandler(object sender, System.EventArgs e)...

        public override void SetupView(XNode node)...
    }
}
```

- Add program initialization routines in the constructor and override the ‘SetupView’ method to bind the grid to the data source. Call the ‘FireDone’ method in the ‘OkButtonClickHandler’ event to raise and bind the event.
- Build the project and ensure that there are no compile time errors.
SMS gateway deployment and test

- Ensure that the assembly for the SMS gateway is copied in the ‘Install’ location of iVend Retail (Management Console & Scheduler Service).
- Select the class name from the drop down.
- Click on “SMS Configuration” to provide the sms configuration details and click on “Ok” button to commit the changes in database.

Click on “Test SMS” button and enter the mobile numbers then click on “Ok” button to send the test message on entered mobile number.