|  |
| --- |
|  |
| Message Notification System Guide |
|  |

|  |
| --- |
|  |

Contents

[Introduction 2](#_Toc358158969)

[Installation 3](#_Toc358158970)

[Prerequisites 3](#_Toc358158971)

[Installation Steps 3](#_Toc358158972)

[Creating the Database 3](#_Toc358158973)

[Configuring the SSIS Packages 3](#_Toc358158974)

[Executing the Packages 7](#_Toc358158975)

[Setting up notifications 7](#_Toc358158976)

[Exception List 8](#_Toc358158977)

[Adding Exceptions 8](#_Toc358158978)

[File formats 8](#_Toc358158979)

[Execution Steps 10](#_Toc358158980)

[Troubleshooting 13](#_Toc358158981)

[Event Log 13](#_Toc358158982)

[Common Problems 14](#_Toc358158983)

# Introduction

This solution uses a SSIS Package to send various types of messages to JMMB customers.

# Installation

## Prerequisites

This system has been designed to run on the Windows platform and has been tested on Windows Server 2003. It utilizes SQL Server 2008 R2 Integration Services and will not work on lower versions of Integration Services. An installation SQL Server 2008 R2 Database Server is also required for successful implementation.

## Installation Steps

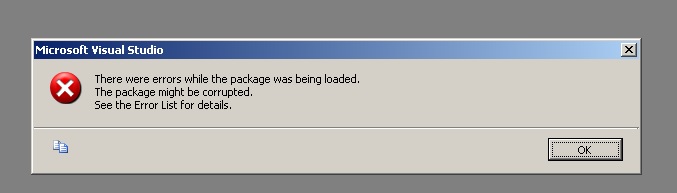
1. Extract the contents of the **JMMB\_CustomerNotification.zip** file to a suitable location like c:\. It contains a folder called **JMMB\_CustomerNotification**.

## Creating the Database

1. Identify a MSSQL instance (install one if necessary) to host the system support/configuration database.
2. There is a file called **JMMB\_CustomerNotification\_1.0.0.0.sql** in the **JMMB\_CustomerNotification** folder. Run it against the MSSQL instance that was identified in step 1 above. It will by default, create a database called **JMMB\_CustomerNotification***.*

## Configuring the SSIS Packages

1. Open the **JMMBCustomerNotification.dtproj** project file with SQL Server Business Intelligence Development Studio.
2. Open both package (**JMMBCustomerNotificationPackage** and **JMMBBulkCustomerFileUpdatePackage**) and change the connections settings for connection **SSISConfiguration** in the **Connection Manager** to point to the **JMMB\_CustomerNotification** database. This will allow it to pick up the rest of the configuration for each package. NOTE: in opening the packages you might get an error message like the image below, this is because the configurations where not found. It should not occur again after the SSISConfiguration is configured.



1. The **JMMB\_CustomerNotification** database has the **SSISConfigurations** table. This table contains the configuration values for anything in the package that would need to be configured in a new environment. Below is a table of the configuration values, the package they belong to and their purpose. Please change the **ConfiguredValue** column to match your environment.

|  |  |  |  |
| --- | --- | --- | --- |
| ConfigurationFilter | PackagePath | Description and Comments | Sample ConfiguredValue |
| JMMBBulkCustomerFileUpdatePackage | \Package.Connections[CustomerDB].Properties[ConnectionString] | Connection String to the database with the CLIENT table containing all the customers’ information. This needs to change to point to the relevant database servers. | Data Source=.;Initial Catalog=JMMB\_CustomerNotification;Integrated Security=True;Application Name=SSIS-JMMBCustomerFileUpdatePackage-{9B5AE74D-0478-47E3-B4EA-4D6D70B15696}LocalHost.JMMB\_CustomerNotification; |
| JMMBBulkCustomerFileUpdatePackage | \Package.Connections[JMMBCustomerNotificationPackage.dtsx].Properties[ConnectionString] | Path to the JMMBCustomerNotificationPackage.dtsx Package | C:\SSIS\_BithdayNotification\JMMBCustomerNotificationPackage.dtsx |
| JMMBBulkCustomerFileUpdatePackage | \Package.Connections[NotificationDB].Properties[ConnectionString] | Connection String to the JMMB\_CustomerNotification database | Data Source=.;Initial Catalog=JMMB\_CustomerNotification;Integrated Security=True;Application Name=SSIS-JMMBBulkCustomerFileUpdatePackage-{C31436C9-D856-460B-BCEF-B7FF04B1AABA}LocalHost.JMMB\_CustomerNotification; |
| JMMBBulkCustomerFileUpdatePackage | \Package.Variables[User::MissingInformatio006EFolder].Properties[Value] | Folder path to store logs of the customers with missing information, must end with backslash | C:\customerfiles\missinginfo\ |
| JMMBCustomerNotificationPackage | \Package.Connections[LocalHost.JMMB\_Notification].Properties[ConnectionString] | Connection String to the JMMB\_CustomerNotification database | Data Source=.;Initial Catalog=JMMB\_CustomerNotification;Integrated Security=True;Persist Security Info=False;Application Name=SSIS-JMMBCustomerNotificationPackage-{3501384A-658D-48EF-A080-3A45CF3BDEF0}LocalHost.JMMB\_Notification; |
| JMMBCustomerNotificationPackage | \Package.Connections[SMSService].Properties[ConnectionString] | Url of the SMS Web Service | http://localhost/AlertService.svc |
| JMMBCustomerNotificationPackage | \Package.Variables[User::ArchivePath].Properties[Value] | Folder path to store processed files, must end with backslash | c:\customerfiles\archive\ |
| JMMBCustomerNotificationPackage | \Package.Variables[User::JMMBEmailAddress].Properties[Value] | Email address to send emails as | DoNotReply@JMMB.com |
| JMMBCustomerNotificationPackage | \Package.Variables[User::SMTPServer].Properties[Value] | SMTP Server Url | mail.server.com |
| JMMBCustomerNotificationPackage | \Package.Variables[User::TemplateEndDelimiter].Properties[Value] | Ending text delimter that will wrap tokens, like Coloum1 | \_} |
| JMMBCustomerNotificationPackage | \Package.Variables[User::TemplateStartingDelimiter].Properties[Value] | Starting text delimter that will wrap tokens, like Coloum1 | {\_ |
| JMMBCustomerNotificationPackage | \Package.Variables[User::WSDLFileLocation].Properties[Value] | SMS Service WSDL File Path | C:\SMSService.wsdl |

# Executing the Packages

For the normal scenario the **JMMBBulkCustomerFileUpdatePackage** package will be the only package that needs direct execution, it in turn will execute **JMMBCustomerNotificationPackage** package. The **JMMBBulkCustomerFileUpdatePackage** has two purposes:

* Update the Customer files with the latest contact information.
* Log which customers have necessary information missing (e.g. missing telephone numbers while requiring Text channel).

It then executes the **JMMBCustomerNotificationPackage** which:

* Send the notifications based on the template for each notification type and channel.
* Archives the processed files.

The **JMMBCustomerNotificationPackage** package uses the **Notifications** table to determine which notifications should be sent and the template they should use. Both packages take into consideration an exception list (the **Exceptions** table) that represent customers desire not to receive notification on a particular channel (or not at all).

## Setting up notifications

Notifications are setup via the **Notifications** table. The user specifies the name of the notification, the customer file paths (**CustomerIdFilePath** is optional), the templates and the subject for email templates. Each notification has two file paths. The **CustomerIdFilePath** has the ID of the customers that should be notified. The **JMMBBulkCustomerFileUpdatePackage** creates a new file (name and path specified by **CustomerContactFilePath**) with current contact information from the Client table in the AutoId database. If **CustomerIdFilePath** fileis not supplied then the **CustomerContactFilePath** file must be supplied with the telephone number/email address to which notification should be sent. An email subject is also needed for notifications that will send emails.

## Exception List

The user can specify that a certain customer should not receive notification on a certain channel and that certain email address or telephone numbers should not receive notifications. Users can specify either the customer number or contact information that will be ignored (telephone number or email). If the customer number is supplied then the channel from which they are exempt from must be supplied. Possible values are:

* Email – Do not send email
* Text – Do not send text
* Both – Do not send any notification

If the telephone number or email is supplied, then it is always exempted from notification.

## Adding Exceptions

There are two ways of adding exceptions:

**Based On Customer ID**

1. Insert row with CustomerID and Channel.

**Based On Contact Information**

1. Insert a row with telephone number and/or email address.

## File formats

There two file formats of importance, the formal for the Customer Id file (path specified by **CustomerIdFilePath** field in the Notifications database) and that for the Customer contact file (path specified by **CustomerContactFilePath** field in the Notifications database). Both formats must conform to the conventions of comma separated values (csv) file specification. Below is a description of the fields in each of these two file formats.

|  |  |
| --- | --- |
| Customer Id File Format | |
| Field Name | **Description** |
| CustomerId | This is a unique identifier for the customer. The data in this field is usually used to lookup additional details about a customer from the relevant systems, e.g. AutoId |
| NotificationChannel | Method of notification. The information in this field will determine how the customer will be notified. Valid values are ***Email***, ***Text*** or ***Both***. |
| Column1 to Column10 | 10 user defined content. These ten fields contain the relevant data to be used in the creation and formatting on the Texts and Emails. As an example, the user may choose to store the customer first name in Column1, the last name in Column2, the customer balance in Column3 and any other piece of information needed to construct an email or text message (up to Column10). |

|  |  |
| --- | --- |
| Customer Contact File Format | |
| Field Name | **Description** |
| CustomerId | This is a unique identifier for the customer. The data in this field is usually used to lookup additional details about a customer from the relevant systems, e.g. AutoId |
| TelephoneNumber | The telephone number to which notification will be sent |
| Email | The email address to which notification will be sent |
| NotificationChannel | Method of notification. The information in this field will determine how the customer will be notified. Valid values are ***Email***, ***Text*** or ***Both***. Texts are sent to the number specified in the TelephoneNumber field (see above) and emails are sent to the email address specified in Email (see above) |
| Column1 to Column10 | 10 user defined content. These ten fields contain the relevant data to be used in the creation and formatting on the Texts and Emails. As an example, the user may choose to store the customer first name in Column1, the last name in Column2, the customer balance in Column3 and any other piece of information needed to construct an email or text message (up to Column10). |

## Execution Steps

These steps must be done after the installation steps (specified above) are successfully carried out. There are two scenarios catered for by the packages.

**Scenario 1 - Customer Id File specified**

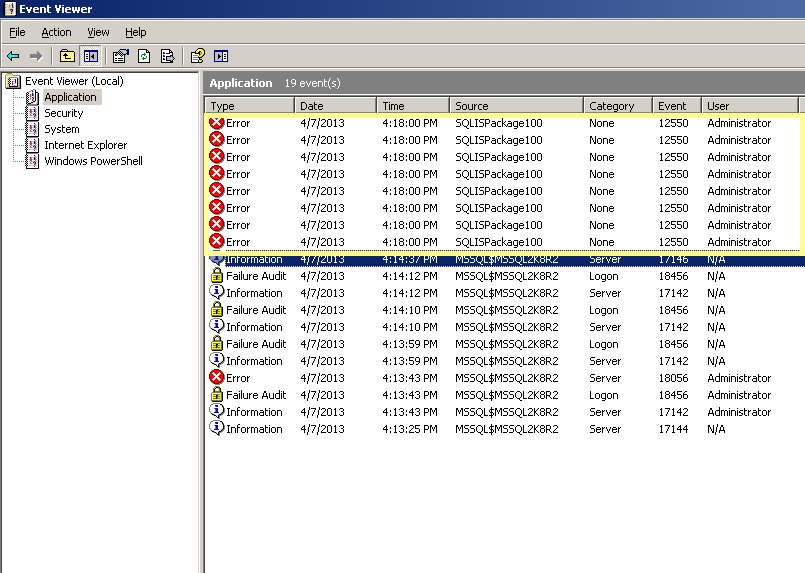
1. Create a file in accordance with the Customer Id File Format specification detailed above.
2. Update (insert a row to) the Notifications table in the **JMMB\_CustomerNotification** database.
   1. Update the Name field with a suitable but unique name for the particular notification (for example BirthdayNotification).
   2. Update the Description field with a relevant description (for example Sends birthday greetings to customers on their birthdays). Description is optional.
   3. Update the **CustomerIdFilePath** field with the path to the file created in step 1 above.
   4. A file will be created by the **JMMBBulkCustomerFileUpdatePackage**, update the **CustomerContactFilePath** field with the required path and name for this output file. This is required. Please note that Column1 – Column10 fields from the Customer Id File from step 1 above are copied directly to this output file.
   5. Update the TextTemplate field. The content in this field will contain placeholders for the data from the input file in step 1. The place holder for Column1 will be {\_Column1\_}, {\_Column2\_} for Column2 and so on. For example, if the user defined fields Column1 and Column2 contained first and last name respectively (the content is determined and documented by the person who creates the input file and may vary from file to file), the TextTemplate field might contain “Happy birthday {\_Column1\_} {\_Column2\_}”. The package will then process this replacing {\_Column1\_} and {\_Column2\_} with data from the input file.
   6. Update the EmailTemplate field similar to step e above using relevant placeholders. The content of this field maybe be html for richly formatted content.
   7. Update the Subject field for content to be delivered by email. Placeholders can also be used in this field.
   8. Set the Active field to 1(true) to mark the row as Active. Rows with fields set to 0(false) will skipped when the package is executed.
3. Execute the **JMMBBulkCustomerFileUpdatePackage** package.

**Scenario 2 - Customer Contact File specified**

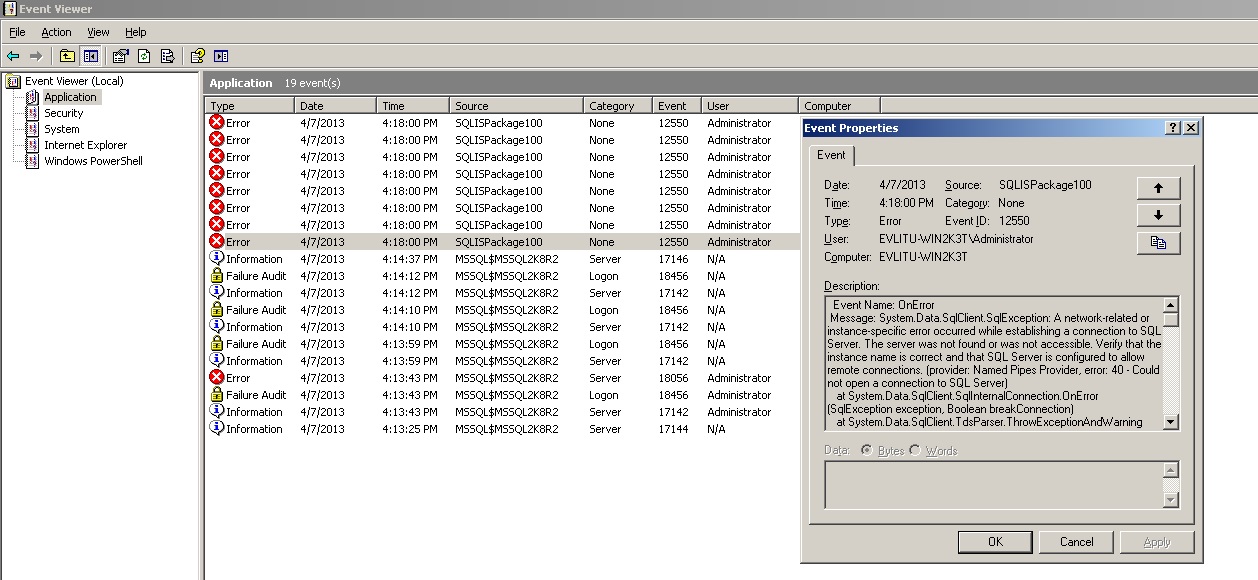
1. Create a file in accordance with the Customer Contact File Format specification detailed above.
2. Update (insert a row to) the Notifications table in the JMMB\_CustomerNotification database.
   1. Update the Name field with a suitable but unique name for the particular notification (for example BirthdayNotification).
   2. Update the Description field with a relevant description (for example Sends birthday greetings to customers on their birthdays). Description is optional.
   3. Update the CustomerIdFilePath field with null.
   4. Update the CustomerContactFilePath field with the file created in step 1 above.
   5. Update the TextTemplate field. The content in this field will contain placeholders for the data from the input file in step 1. The place holder for Column1 will be {\_Column1\_}, {\_Column2\_} for Column2 and so on. For example, if the user defined fields Column1 and Column2 contained first and last name respectively (the content is determined and documented by the person who creates the input file and may vary from file to file), the TextTemplate field might contain “Happy birthday {\_Column1\_} {\_Column2\_}”. The package will then process this replacing {\_Column1\_} and {\_Column2\_} with data from the input file.
   6. Update the EmailTemplate field similar to step e above using relevant placeholders. The content of this field maybe be html for richly formatted content.
   7. Update the Subject field for content to be delivered by email. Placeholders can also be used in this field.
   8. Set the Active field to 1 to mark the row as Active. Rows with fields set to 0 will skipped when the package is executed.
3. Execute the **JMMBCustomerNotificationPackage** package.

# Troubleshooting

## Event Log

When errors happen in the package, it is logged to the event under Application log. The source is SQLISPackage100.

It normally logs more than one event because the failure is recorded at various failure points. In the image below, the package had a problem connecting to the database as seen in the Description section of the event.



## Common Problems

If errors occur here are some things you can check before escalating it:

* Ensure that the user running the package has read/write permission to the folders, wsdl file and **JMMBCustomerNotificationPackage** package file.
* Ensure that the csv files are not being use by another program.
* If the wsdl file is changed then the **JMMBCustomerNotificationPackage** package will have to be updated.
* Ensure databases are accessible with configured connection strings.