|  |
| --- |
| Photo displaying partial image of two pie charts on a canvas-textured page |
| FACS SMS  Process Document |
| |  |  |  | | --- | --- | --- | | **"Anish J. Xavier" <axavier@convergentusa.com>** | 9/6/19 | Features and Functions | |



**Copyright Information**

This document is protected by copyright law and may not be reproduced or distributed either in part or in total. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Convergent USA. Under the law, reproducing includes translating into another language or format.

The licensee is not allowed to pass on the software or the accompanying written materials to third parties or make them otherwise available without prior written agreement of the licensor.

Information in this documentation refers to possible product extensions or to avail­able accessories is not legally binding, especially because the products are subject to continuous adaptation and because the information may also relate to future develop­ment. The contents of this documentation can change without prior notice and does not represent any legal obligation on the part of Convergent USA.

Convergent USA cannot be made liable for the correctness of information in this documentation or for damages resulting from the use of this information or the impossibility of using this information.

All other legal regulations for using the software and the corresponding documentation are set out the applicable license agreement.

All other product and company names mentioned in this documentation are trademarks of their respective companies.

You are welcome to suggest to us how to improve this documentation. Please e-mail us at [ConvergentUSA-PMO@convergentusa.com](mailto:ConvergentUSA-PMO@convergentusa.com)

**Copyright © 2019 Convergent USA, all rights reserved**.

|  |  |
| --- | --- |
| **Author** | |
| **Created By** | **Date** |
| Veerendra Dwivedi | 28/11/2019 |
| **Reviewed By** | **Date** |
| Anish J. Xavier | 28/11/2019 |
|  |  |
| **Approved By** | **Date** |
|  |  |

**Table of Content**

[Overview 4](#_Toc25798660)

[Introduction to SMS 4](#_Toc25798661)

[Key Benefits of SMS Marketing for Debt Collection 4](#_Toc25798662)

[Useful SMS Templates for Debt Collection 5](#_Toc25798663)

[COI SMS Implementation 5](#_Toc25798664)

[Flow Diagram 6](#_Toc25798665)

[FACS to SMS Application 6](#_Toc25798666)

[FACS User Defined Windows 7](#_Toc25798667)

[Pre-Validation 7](#_Toc25798668)

[SMS Processing 7](#_Toc25798669)

[1. Bulk Insert Process 7](#_Toc25798670)

[2. FACS\_SEND\_SMS: 8](#_Toc25798671)

[3. Twilio response and update: 8](#_Toc25798672)

[4. TwiliologfileRead: 8](#_Toc25798673)

[5. Name-Mismatch\_SENT\_TO\_FACS: 9](#_Toc25798674)

[FACS Events 9](#_Toc25798675)

[FACS Functions 9](#_Toc25798676)

[Database Table 11](#_Toc25798677)

[TB FACS\_File2Process 11](#_Toc25798678)

[Staging\_SMS\_Input\_From\_Facs 11](#_Toc25798679)

[Staging\_SMS\_Input\_From\_FACS\_DUPLICATES 11](#_Toc25798680)

[TBStaging\_SMS\_Input\_From\_FACS\_STOP 12](#_Toc25798681)

[TB Send\_SMS 12](#_Toc25798682)

[TB InBound\_SMS\_Response 13](#_Toc25798683)

[TB inBound\_SMS\_Status 13](#_Toc25798684)

[TB Send\_SMS\_error\_history 14](#_Toc25798685)

[Send\_SMS\_history 15](#_Toc25798686)

[F\_SMS\_ClientClass\_ClientName 16](#_Toc25798687)

[F\_SMS\_Text\_ByState 16](#_Toc25798688)

[F\_SMS\_Text\_ByState\_Transaction\_History (for historical records pre-update) 16](#_Toc25798689)

[Stored Procedures used 17](#_Toc25798690)

[Using SQL Function 18](#_Toc25798691)

# Overview

Convergent is a leader in providing Account Receivable management services to various industries. Convergent leverages industry leading FACS platform from Ontario systems for managing customer accounts. As a part of it business process of reaching out to borrowers, Convergent leverages SMS text messaging by building a custom application that integrates with Twilio.

# Introduction to SMS

**Short message service (SMS)** will play a very vital role in the future business areas whose are popularly known as mobile banking, organizational marketing system etc. For this future, SMS could make a mobile device in a business tool as it has the availability and the effectiveness.

SMS, as used on modern devices, originated from radio telegraphy in radio memo pagers that used standardized phone protocols. These were defined in 1985 as part of the Global System for Mobile Communications (GSM) series of standards. The first SMS message was sent in 1992.

The protocols allowed users to send and receive messages of up to 160 characters (when entirely alpha-numeric) to and from GSM mobiles. Although most SMS messages are mobile-to-mobile text messages, support for the service has expanded to include other mobile technologies, such as ANSI CDMA networks and Digital AMPS.

SMS is also employed in mobile marketing, a type of direct marketing. According to one market research report, as of 2014, the global SMS messaging business was estimated to be worth over $100 billion, accounting for almost 50 percent of all the revenue generated by mobile messaging.

**Here are the best applications of SMS services for debt collection**.

**Text payment notifications**. Automatically send payment reminders to customers with the help of SMS software. Text payment reminders can help you reduce collection costs and encourage customers to pay their debts.

**Follow up** on payment confirmations. Don’t forget to thank your customers after they complete a payment. This will improve your relationship and increase their loyalty.

**Schedule** payment reminders. Believe it or not, some people simply forget to pay their debts. You can prevent this by sending scheduled reminders ahead of time. You can automatically schedule weekly or daily reminders.

**Send SMS** messages with account details. For debt collection services, clarity is key. Send personalized SMS messages to clients with information about their account and provide all the tools they may require.

**Send useful information** via text. You can automatically send useful information such as payment plans, debt amounts, interest rates, etc. Full transparency and constant updates will make it easier to collect the entire debt.

## Key Benefits of SMS Marketing for Debt Collection

**Lift response rates**. Where debt collection is involved, obtaining a response from customers is key. Many clients refuse to answer calls or emails but respond to texts.

**Reach all clients instantly**. You can reach all your clients immediately by sending bulk SMS messages. You can also enable delivery receipts in order to identify deregistered numbers or failed deliveries.

**Improve customer relationships**. Debt collection is stressful for you as well as the client. Who knows what unfortunate circumstances may have led to a missed payment? Text messaging provides a non-invasive means of communication that will not endanger your relationships with customers.

**Save time**. You can save a lot of time with SMS debt collection by taking full advantage of automated messaging software and templates. These templates contain concise instructions that customers can follow.

**Create a sense of urgency**. Text reminders create a sense of urgency that reinforce the importance of timely payments. Make sure you include due dates and other useful information to persuade customers to submit their payments on time.

## Useful SMS Templates for Debt Collection

Check out our text message templates for debt collection notices that replicate official emails sent by debt collectors and tell us what you think.

**Soft Reminder**

“Dear \*client\*, we would like to remind you that the amount \*sum\* was due for payment on \*date\*. To avoid further costs please forward the payment no later than \*date\*.”

**Strong Reminder**

“Dear \*client\*, we have still not received the amount \*sum\*” that was due on \*date\*. Should your payment be received within the next 7 days, we will not take court action. We urge you not to ignore this last reminder.”

**Final Reminder**

“Dear \*client\*, despite our previous reminders we have still not received the payment. We regret to inform you that we have no other choice but to undertake legal action against you in order to retrieve the debt.”

**Thank you, Message,**

“Dear \*client\*, your payment of \*sum\* has been accepted and your debt has been cleared. Thank you!”

SMS debt collection has many benefits. If you’re ready to implement these strategies for your business, feel free to contact us. We’d love to help.

# COI SMS Implementation

COI has built a custom application that allows it to send SMS messages to its borrowers.

# Flow Diagram

# 

# FACS to SMS Application

On a daily basis FACS will export a file to the SMS application, this file will have the following elements and will be used for sending out SMS messages. The file will be created based on Logic block that will pick up accounts in 3TXT status. These accounts will be excluded from dialing on the dialer. Accounts will be moved to the 3TXT status based on Tactics that operations team will run. The file will have the following elements

* + Account Number
  + Namespace
  + Type of message
  + Mobile Number
  + Call Back number
  + Message ID
  + Client ID
  + Client Class

The file would reside on: **wasql01p.e-r-solutions.com** under **L:\FACSSMS\FacsInputFiles**\ with filename format as SMS\_Output\_yyyymmdd.txt

# FACS User Defined Windows

FACS User defined Window (UDW)will track SMS activity on accounts. This window will also be used when we want to send messages that need manual consent.

|  |  |  |
| --- | --- | --- |
| Field | Heading | Description |
| 1 | SMS COUNTER | Number of SMS Sent on this account |
| 2 | SMS SEND DATE | Last SMS Sent date |
| 3 | SMS CONSENTFLAG | Flag to be set when manual SMS needs to be sent |
| 4 | UNSUBSCRIBE DTE | Date on when STOP message is received |
| 5 | CONSENTEDMOBILE | Mobile Number for Manual SMS |
| 6 | CELL MONITOR FL | Flag if using Cell Phone Monitoring Flag |
| 7 | CELL SCRUB DATE | Flag is using Cell Phone Scrubbing Service |
| 8 | SEND TO SMS Y N | Flag set when account has been sent to SMS APP |
| 9 | TYPE OF MESSAGE | Message ID |
| 10 | NAME MISMATCH | Date when we received Name Mismatch from SMS Service |
| 11 | SMS FAILURE | Date when SMS Message failed |
| 13 | SMS FAIL TWILIO | Date when messages failed to Twilio |
| 14 | SMS Response | Date when we got a incoming message back |
| 20 | SMS FAILURE REASON | Character field with the reason code why SMS Failed |
| 21 | MATCH MISMATCH | Character field to show Boku response |

## Pre-Validation

Pre-validation or pre-processing refers to the process of selecting accounts, extracting pertinent information and getting the data ready to use for next steps.

Business places accounts in a ‘3TXT’ disposition (by running a Tactic), based on pre-defined logic, which indicates that they are okay to send SMS to. A flat file populated with the required fields is then generated (refer to [**FACS to SMS Application**](#_FACS_to_SMS)). This file is then moved to a specific directory. The flat file is ingested into a staging table in SQL Server using the stored procedure FACS\_BulkLoad2Staging.

SQL Server name: *wasql01p.e-r-solutions.com*

Database: *FACSSMSDB*

## SMS Processing

During testing, SMS processing is carried out using jobs created in Task Scheduler, however, the production run would involve Operations choosing options from an interactive web-package (a container with the website and all associated directories and files) and running texting campaigns on their own. In essence, the activities elucidated below would run automatically with minimal user intervention.

Processing broadly involves the following programs interacting with the FACS application

### Bulk Insert Process

This process takes the raw files from FACS, inserts it into staging table, removes duplicates and accounts that we have received a message from and inserts the data in the [Send\_SMS] table. This table will be used by the FACS\_SEND\_SMS application for sending out the message. This process calls the following stored procedures.

* FACS\_BulkLoad2Staging
* FACS\_Get\_FACS\_File2Process
* FACS\_BulkLoad2Staging\_DeleteDuplicates
* Insert\_Staging\_to\_Send\_SMS

### FACS\_SEND\_SMS:

It calls the Boku API and receives score for the accounts in the [SEND\_SMS] table.

Accounts that qualify for SMSAuthorize and Boku score are sent to Twilio with the message. Accounts that do not qualify for SMSAuthorize and Boku score are inserted in the table [SEND\_SMS\_Error\_history]. Accounts that qualify and have successful messages accepted by Twilio are inserted in to [Send\_SMS\_history] table.

The fields that are updated in SEND\_SMS are

* 1. FirstNameScore
  2. LastNameScore
  3. CorrelationID
  4. BokuResponse- Entire JSON object

This process calls the following stored procedures:

Get\_PhoneList\_For\_Send\_SMS,

Insert\_SMS\_Send\_error\_History

Insert\_SMS\_Send\_History

Update\_SMS\_Send\_Status

This process calls the following FACS API: **SMSAuthorize**

This process calls the following functions: Fn\_ReplaceClientNamePhoneForSMS

### Twilio response and update:

We have 2 webservices that receives messages from Twilio.

<https://peco.convergentusa.com/FACSSMS/api/GetSmsResponse> - This webservice gets updates from Twilio on message delivery status. Each one of these webservices writes a log file.

Location of log file -C:\inetpub\wwwroot\FACSSMS\Logs

<https://peco.convergentusa.com/FACSSMS/api/GetSmsInBoundMsg> - This webservice gets any return messages being sent by the borrower.

Location of log file -C:\inetpub\wwwroot\FACSSMS\Inbound\_Logs

### TwiliologfileRead:

This application reads the log files created in Step 3, updates the tables and calls the FACS webservices

Logfile -GetSmsResponse

This calls the procedure Insert\_inbound\_SMS\_Status which inserts a row for every status update in [Inbound\_SMS\_status] table.

This program calls

1. the FACS function **SMSSENT** for ‘delivered’ status.
2. the FACS function **SMSFailure** for ‘undelivered’ status

See FACS function definitions in [FACS Functions](#_FACS_Functions) section.

Logfile -GetSmsInBoundMsg

This calls the procedure Insert\_Inbound\_SMS\_response which inserts a row into the table [Inbound\_SMS\_Response] with every message we get from the borrower.

This program calls

1. the FACS function **WriteNote** for delivered status.
2. the FACS function **SMSStop** for undelivered status

**We only call these WriteNote and SMSStop function when we receive an incoming message from the borrower.**

### Name-Mismatch\_SENT\_TO\_FACS:

This application

1. Executes the stored procedure GET\_SEND\_SMS\_NameMismatchSentToFACS
2. Calls the FACS Function **NameMismatch** to update FACS with Name mismatch
3. Lastly, executes Update\_Send\_SMS\_NameMisMatch\_Status\_ForFACS procedure that updates error history when name-mismatch is updated on FACS

# FACS Events

Below are the list of events and related FACS actions that will be taken as these events come in

|  |  |  |
| --- | --- | --- |
| **Event** | **Description** | **FACS Result** |
| SMS Validity | First step in the application to check if the account is still valid for Texting | Checks FACS account if it is still valid for SMS, if it is a failure then  Add UDW 165 and field 12 increment counter 1 and write note “SMSAuthorize Failure” |
| Boku Name Match | Event is triggered when we get a response back from Boku with Score where first name and last name score are greater than 5 | Boku Scrub :Name match phone # <<phone#>> Add UDW 165 field 22 with current Date . Add UDW 165 field 21 add note Name match |
| Boku Name Mismatch | Event is triggered when we get a response back from Boku with Score where first name or last name score is less than 5 | Boku Scrub :Name Mismatch phone # <<phone#>> Add UDW 165 field 22 with current Date . Add UDW 165 field 21 add note Name Mismatch Add UDW 165 field 10 with current date |
| Outbound SMS Sent | When we get a success response from Twilio that message had been delivered | This function increments the UDW field “SMS COUNTER” UDW 165 field 01 by 1 UDW 165 field “SMS SEND DATE” field 02 with current date. Add Note SMSSvc: Text sent to <<phone #>> at <<date sent>> <<time sent including time zone>> Add Note SMSVc -Actual Message being sent |
| Outbound Failed Text-Twilio | When we get a failure response from Twilio | This function updates the UDW field “SMS FAILURE” UDW 165 field 13 with current date  SMSSvc: \*ERROR\*:SMS Twilio Failure to <<phone #>>  Add UDW with error code |
| Outbound Failed Text-Carrier | When we get a failure response from Twilio-Carrier | This function updates the UDW field “SMS FAILURE” UDW 165 field 11 with current date  SMSSvc: \*ERROR\*:SMS Delivery Failure to <<phone #>>  Add UDW with error code |
| Inbound SMS Received | When the customer responds to the SMS message by texting back | SMSSvc: Message text received from <<phone #>> Update UDW 165 Field 14 with date Based on Actual message change disposition See Table Below |

##### Keywords and suggested actions for inbound SMS responses

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Keyword** | **Action** | **Disposition** | **Treat as Stop** | **Update Notes Y** | **Phone Flag Change** |
| Payment, Pay, Paid | Check for Consumer Payment | 3500 | N | Y | N |
| Fraud, Dispute, Error, I don’t owe | Client Inquiry - Needs review | 3500 | Y | Y | N |
| Legal, Sue, Suit, lawsuit, attorney, lawyer | Needs Review | 35SM | Y | Y | Y-! |
| Stop, Sto, Opt Out | Cease SMS | 35SM | Y | Y | N |
| Cease, Do not text, Wrong Number, Invalid | Cease SMS | 35SM | y | Y | Y- B |
| Bankrupt, Bankruptcy, Banko, BK | Cease Action | 3500 | Y | Y | N |
| Dead, Deceased, Died | Cease Action | 3500 | Y | Y | N |
| Driving | Nothing Needed | Same | N | Y | N |
| Response Greater than 30 characters | Likely needs manual review | 35SM | N | Y | N |
| Profanity - Fuck, Shit, Bitch, (Change note to say received Profanity) | Needs review | 35SM | Y | Y | N |
| Who is this | Needs review | 3SMS | N | Y | N |
| All other Response - If we have more than one word | Needs Review | 35SM | N | Y | N |

# FACS Functions

The SMS application will update the FACS application with the following messages functions

* **Function**: SMSAuthorize- First step in the application to check if the account is still valid for Texting   
  **Description**: Checks FACS account if it is still valid for SMS
  + **Parameter(s)**: *FACSAcct#*
  + **Example** **Call**: <http://wafacs:57772/csp/blu/User.FACSDigital/SMSAuthorize/12345>
* **Function**: SMSStop – When the customer responds to the SMS message by texting back   
  **Description**: SMSSvc: Message text received from <<phone #>> Update UDW 165 Field 14 with date Based on Actual message change disposition See Table Below   
  **Parameter(s)**: *FACSAcct#, PhoneNumber, InboundMessage*
  + **Example** **Call:** <http://wafacs:57772/csp/RED/User.FACSDigital/SMSStop/5368104/3109025157/InBound> message
* **Function**: SMSFailure- When we get a failure response from Twilio
  + **Description**: This function updates the UDW field “SMS FAILURE” UDW 165 field 13 with current date SMSSvc: \*ERROR\*:SMS Twilio Failure to <<phone #>>
  + **Parameter(s)**: *FACSAcct#, PhoneNumber, ErrorCode*
  + **Example** **Call**: <http://wafacs:57772/csp/RED/User.FACSDigital/SMSFailure/5368104/3109025157/30003>
* **Function**: NameMatch- Event is triggered when we get a return back form Boku with Score and firstname and last name are greater than 5    
  **Description**: Boku Scrub :Name match phone # <<phone#>> Add UDW 165 field 22 with current Date . Add UDW 165 field 21 add note Name match    
  **Parameter(s)**: *FACSAcct#*, *PhoneNumber*

**Example** **Call**: <http://wafacs:57772/csp/RED/User.FACSDigital/SMSNameMatch/44813896/8182977443>

* **Function**: NameMismatch – Event is triggered when we get a return back form Boku with Score and firstname or last name are less than 5

**Description**: Boku Scrub :Name Mismatch phone # <<phone#>> ,Add UDW 165 field 22 with current Date . Add UDW 165 field 21 add note Name Mismatch, Add UDW 165 field 10 with current date   
**Parameter(s)**: *FACSAcct#, PhoneNumber*

* + **Example** **Call**: <http://wafacs:57772/csp/RED/User.FACSDigital/SMSNameMisMatch/5368104/3109025157>
* **Function**: Delivered – When we get a success from Twilio that message had been delivered   
  **Description**: This function increments the UDW field “SMS COUNTER” UDW 165 field 01  by 1

UDW 165 field “SMS SEND DATE”  field 02 with current date. Add Note SMSSvc: Text sent to <<phone #>> at <<date sent>> <<time sent including time zone>> Add Note SMSVc -Actual Message being sent   
**Parameter(s)**: *FACSAcct#, PhoneNumber,* *Message Sent*

* + **Example** **Call**: <http://wafacs:57772/csp/RED/User.FACSDigital/SMSDelievered/5368104/3109025157/Actual> Message
* **Function**: Undelivered– When we get a failure response from Twilio-Carrier

**Description**: This function updates the UDW field “SMS FAILURE” UDW 165 field 11 with current date SMSSvc: \*ERROR\*:SMS Delivery Failure to <<phone #>> Add UDW  165 field 20 with error code   
**Parameter(s)**: *FACSAcct#, PhoneNumber, ErrorCode* **Example** **Call**: <Http://wafacs:57772/csp/RED/User.FACSDigital/SMSUndelivered/5368104/3109025157/30003>

# Database Table

## TB FACS\_File2Process

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Column Type** | **Constraint** | **Description** |
| [modifiedDateTime] | [varchar](100) |  | Timestamp of file generation |
| [FileType] | [varchar](100) |  | File type |
| [fileSize] | [varchar](100) |  | Size of the file in bits |
| [sFileName] | [varchar](100) |  | Name of the generated file |
| [EST\_Timestamp] | [varchar](100) |  | - |

## Staging\_SMS\_Input\_From\_Facs

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Column Type** | **Constraint** | **Description** |
| [NameSpace] | [varchar](200) |  | Directory of FACS used |
| [ClientClass] | [varchar](200) |  | Short name for client from FACS |
| [ClientId] | [varchar](200) |  | Client ID from FACS |
| [FACSAcct#] | [varchar](200) |  | FACS account number assigned to borrower. It is a unique field. |
| [CellPhone#] | [varchar](50) |  | Borrower’s cell phone number, from Phone1 field in FACS |
| [SMSText#] | [varchar](50) |  | SMSText key. Determines type of message (with or without client name) |
| [LastName] | [varchar](200) |  | Borrower’s Last Name |
| [FirstName] | [varchar](200) |  | Borrower’s First Name |
| [StateCode] | [varchar](50) |  | 2 letter state code |

|  |  |  |  |
| --- | --- | --- | --- |
| Staging\_SMS\_Input\_From\_FACS\_DUPLICATES | | |  |
| **Column Name** | **Column Type** | **Constraint** | **Description** |
| [DateInserted] | [datetime] |  | Datetime record is inserted to table |
| [FacsNum] | [varchar](200) |  | FACS account number assigned to borrower. It is a unique field. |
| [CellPhone] | [varchar](200) |  | Borrower’s cell phone number, from Phone1 field in FACS |
| [deleteflg] | [int] |  | Flag set to 0 or 1. Entry deletes records deemed duplicate. 0 to delete, 1 to keep |
| [FacsFileName] | [varchar](1000) |  | Name of the generated file |
| [Reason] | [varchar](1000) |  | Reason for deletion. Eg. “Duplicate Cell Phone Field” |

|  |  |  |  |
| --- | --- | --- | --- |
| TBStaging\_SMS\_Input\_From\_FACS\_STOP | |  |  |
| **Column Name** | **Column Type** | **Constraint** | **Description** |
| [DateInserted] | [datetime] |  | Datetime record is inserted to table |
| [FacsNum] | [varchar](200) |  | FACS account number assigned to borrower |
| [CellPhone] | [varchar](200) |  | Borrower’s cell phone number, from Phone1 field in FACS |
| [Body] | [varchar](max) |  | Retrieved from table InBound\_SMS\_Response if cell phone number is present in the table |
| [FacsFileName] | [varchar](1000) |  | Name of the generated file |

## TB Send\_SMS

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Column Type** | **Constraint** | **Description** |
| [ID] | [int] | NOT NULL, | Index |
| [DateInserted] | [datetime] |  | Datetime record is inserted to staging table |
| [NameSpace] | [varchar](50) |  | Directory of FACS used |
| [ClientClass] | [varchar](200) |  | Short name for client from FACS |
| [ClientId] | [varchar](200) |  | Client ID from FACS |
| [FACSAcctNum] | [int] |  | FACS account number assigned to borrower |
| [CellPhone] | [varchar](50) |  | Borrower’s cell phone number |
| [SMSTextKey] | [int] |  | SMSText key. Determines type of message (with or without client name) |
| [LastName] | [varchar](50) |  | Borrower’s Last Name |
| [FirstName] | [varchar](50) |  | Borrower’s First Name |
| [StateCode] | [varchar](50) |  | 2 letter state code |
| [BokuFirstNameScore] | [int] |  | Name-match score assigned by Boku for first name |
| [BokuLastNameScore] | [int] |  | Name-match score assigned by Boku for last name |
| [BokuCorrelationID] | [varchar](100) |  | Unique ID to call the Boku API |
| [BokuResponse] | [varchar](2000) |  | Boku Response in JSON |
| [NameMisMatchSentToFACS] | [int] |  | Legacy column. No longer used |
| [SmsSid] | [varchar](200) |  | Unique ID provided by Twilio |
| [TwilioStatus] | [varchar](200) |  | Whether accepted by Twilio |
| [SMSSentStatus] | [int] |  | Flag set to 1 if FACS SMSAuthorize does not return change in account status/disposition |
| [SMSSentDateTime] | [datetime] |  | Time stamp when SMS |
| [ResponseMsg] | [varchar](2000) |  | Given by Twilio Response as Json |
| [FacsFileName] | [varchar](1000) |  | FACS file name generated by FACS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TB InBound\_SMS\_Response | | | |  |
| **Column Name** | **Column Type** | | **Constraint** | **Description** |
| [id] | [int] | | NOT NULL, IDENTITY (1,1) | Index |
| [DateTimeInserted] | [datetime] | |  | Datetime response is recorded by Twilio |
| [PhoneNumber] | [varchar] (50) | |  | Borrower’s cell phone number including country code |
| [SmsSid] | [varchar] (100) | |  | Unique ID provided by Twilio |
| [MsgSid] | [varchar] (100) | |  | Unique ID for each message assigned by Twilio |
| [Body] | [varchar](max) | |  | Body of SMS response text |
| [ResponseSentToFACS] | [int] | |  | Flag indicating whether response has been sent to FACS. 1=sent, 0= not sent |
| [StopSentToFACS] | [int] | |  | If response is received stop flag updated on FACS. 1=sent, 0= not sent |
| [JasonObject] | [varchar](max) | |  | Response as JSON object |
| TB inBound\_SMS\_Status | |  |  |  |
| **Column Name** | | **Column Type** | **Constraint** | **Description** |
| [id] | | [int] | Primary Key, IDENTITY (1,1) | Index, primary key |
| [DateTimeInserted] | | [datetime] |  | Datetime status is recorded by Twilio |
| [PhoneNumber] | | [varchar](50) |  | Borrower’s cell phone number |
| [SmsSid] | | [varchar](100) |  | Unique ID provided by Twilio |
| [SmsStatus] | | [varchar](100) |  | Unique ID for each message assigned by Twilio |
| [StatusSentToFacs] | | [int] |  | For each status, flag indicating whether it has been sent to FACS. 1=sent, 0= not sent |
| [JasonObject] | | [varchar](max) |  | JSON object containing Twilio’s status with other details |

|  |  |  |  |
| --- | --- | --- | --- |
| TB Send\_SMS\_error\_history | |  |  |
| **Column Name** | **Column Type** | **Constraint** | **Description** |
| [ID] | [int] | Primary Key, IDENTITY (1,1) | Index. Primary key |
| [DateInserted] | [datetime] | Primary Key | Datetime record is inserted to table |
| [NameSpace] | [varchar](50) |  | Directory of FACS used |
| [ClientClass] | [varchar](200) |  | Short name for client from FACS |
| [ClientId] | [varchar](200) |  | Client ID from FACS |
| [FACSAcctNum] | [int] |  | FACS account number assigned to borrower |
| [CellPhone] | [varchar](50) |  | Borrower’s cell phone number |
| [SMSTextKey] | [int] |  | SMSText key. Determines type of message (with or without client name) |
| [LastName] | [varchar](50) |  | Borrower’s Last Name |
| [FirstName] | [varchar](50) |  | Borrower’s First Name |
| [StateCode] | [varchar](50) |  | 2 letter state code |
| [BokuFirstNameScore] | [int] |  | Name-match score assigned by Boku for first name |
| [BokuLastNameScore] | [int] |  | Name-match score assigned by Boku for last name |
| [BokuCorrelationID] | [varchar](100) |  | Unique ID to call the Boku API |
| [BokuResponse] | [varchar](2000) |  | Boku Response in JSON |
| [NameMisMatchSentToFACS] | [int] |  | Legacy column. No longer used |
| [SmsSid] | [varchar](200) |  | Unique ID provided by Twilio |
| [TwilioStatus] | [varchar](200) |  | Whether accepted by Twilio |
| [SMSSentStatus] | [int] |  | Flag set to 1 if FACS SMSAuthorize does not return change in account status/disposition |
| [SMSSentDateTime] | [datetime] |  | Flag set to 1 if FACS SMSAuthorize does not return change in account status/disposition |
| [ActualMessage] | [varchar](2000) |  | NA |
| [ResponseMsg] | [varchar](2000) |  | NA |
| [FacsFileName] | [varchar](1000) |  | Name of the generated flat file associated with this run |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Send\_SMS\_history |  | |  | | |  |
| **Column Name** | **Colomn Type** | | **Constraint** | | | **Description** |
| [ID] | [int] | | Primary Key, IDENTITY (1,1) | | | Index. Primary key |
| [DateInserted] | [datetime] | | Primary Key | | | Datetime record is inserted to table |
| [NameSpace] | [varchar](50) | |  | | | Directory of FACS used |
| [ClientClass] | [varchar](200) | |  | | | Short name for client from FACS |
| [ClientId] | [varchar](200) | |  | | | Client ID from FACS |
| [FACSAcctNum] | [int] | |  | | | FACS account number assigned to borrower |
| [CellPhone] | [varchar](50) | |  | | | Borrower’s cell phone number |
| [SMSTextKey] | [int] | |  | | | SMSText key. Determines type of message (with or without client name) |
| [LastName] | [varchar](50) | |  | | | Borrower’s Last Name |
| [FirstName] | [varchar](50) | |  | | | Borrower’s First Name |
| [StateCode] | [varchar](50) | |  | | | 2 letter state code |
| [BokuFirstNameScore] | [int] | |  | | | Name-match score assigned by Boku for first name |
| [BokuLastNameScore] | [int] | |  | | | Name-match score assigned by Boku for last name |
| [BokuCorrelationID] | [varchar](100) | |  | | | Unique ID to call the Boku API |
| [BokuResponse] | [varchar](2000) | |  | | | Boku Response in JSON |
| [NameMisMatchSentToFACS] | [int] | |  | | | Legacy column. No longer used |
| [SmsSid] | [varchar](200) | |  | | | Unique ID provided by Twilio |
| [TwilioStatus] | [varchar](200) | |  | | | Whether accepted by Twilio |
| [SMSSentStatus] | [int] | |  | | | Flag set to 1 if FACS SMSAuthorize does not return change in account status/disposition |
| [SMSSentDateTime] | [datetime] | |  | | | Flag set to 1 if FACS SMSAuthorize does not return change in account status/disposition |
| [ActualMessage] | [varchar](2000) | |  | | | Body of actual message sent to borrower |
| [ResponseMsg] | [varchar](2000) | |  | | | Twilio response as JSON object for SMS attempt |
| [FacsFileName] | [varchar](1000) | |  | | | Name of the generated flat file associated with this run |
|  | |  | |  |  |

## 

## F\_SMS\_ClientClass\_ClientName

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Colomn Type** | **Constraint** | **Description** |
| [Namespace] | [varchar](100) |  | Directory of FACS used |
| [ClientClass] | [varchar](10) | Primary Key | Short name for client from FACS |
| [Sms\_ClientName] | [varchar](500) |  | Friendly name for client |
| [Call\_back\_Phone] | [varchar](500) |  | LiveVox TFN for borrower to call back on |
| [PrefixForAccount] | [varchar](2) |  | Single character prefix for account number to designate namespace |
| [UseBokuWebAPI] | [varchar](1) |  | If Y, will use Boku name-match and get score. Otherwise do not use service |
| [DoNameMatch] | [varchar](1) |  | If Y then records will be filtered based on name-match threshold |
| [Twilio\_AccountSid] | [varchar](100) |  | Unique ID associated with Twilio account used |
| [Twilio\_AuthToken] | [varchar](100) |  | Token provided by Twilio for authorization to use service |
| [Twilio\_messagingServiceSid] | [varchar](100) |  | For Twilio to assign long codes per their algorithm |

## F\_SMS\_Text\_ByState

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Column Type** | **Constraint** | **Description** |
| [Namespace] | [varchar](200) | Primary Key | Directory of FACS used for client |
| [ClientClass] | [varchar](200) | Primary Key | Short name for client from FACS |
| [SMSTextKey] | [int] | Primary Key | Determines type of message (with or without client name) |
| [State] | [varchar](200) |  | State name |
| [StateCode] | [varchar](50) | Primary Key | 2 letter state code |
| [MsgSeq] | [int] |  | Not used |
| [SMSTextMessage] | [varchar](max) |  | Template body of message to be used by State |
| [user] | [varchar](500) |  | Username that added the record |
| [DateInserted] | [datetime] |  | Datetime record was inserted |

## F\_SMS\_Text\_ByState\_Transaction\_History (for historical records pre-update)

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Column Type** | **Constraint** | **Description** |
| [Namespace] | [varchar](200) |  | Directory of FACS used for client |
| [ClientClass] | [varchar](200) |  | Short name for client from FACS |
| [SMSTextKey] | [int] |  | Determines type of message (with or without client name) |
| [State] | [varchar](200) |  | State name |
| [StateCode] | [varchar](50) |  | 2 letter state code |
| [MsgSeq] | [int] |  | Not used |
| [SMSTextMessage] | [varchar](max) |  | Template body of message to be used by State |
| [user] | [varchar](500) |  | Username that added the record |
| [DateInserted] | [datetime] |  | Datetime record was inserted |

## Stored Procedures used

#### Used in web portal:

|  |
| --- |
| 1) **Create\_SMS\_ClientClass\_ClientName –** To create new client entries through web portal |
| 2) **Display\_By\_Date –** View SMS results by date |
| 3) **Display\_SMS\_ByState –** View SMS by state based on selection from drop down menus |
| 4) **Display\_SMS\_ClientClass\_ByState –** select client class by namespace selection in drop down. Used to copy or insert |
| 5) **Display\_SMS\_ClientClass\_ClientName –** Displays details from [F\_SMS\_ClientClass\_ClientName] |
| 6) **Display\_SMS\_Namespace\_ByState –** Displays distinct namespaces for drop-down |
| 7) **Display\_SMS\_SMSTextKey\_ByState –** Displays SMSTextKey based on selection of ClientClass and NameSpace from drop-down menus. |
| 8) **DisplayAA\_SMS\_ByState** – Displays all details from [F\_SMS\_Text\_ByState] Used in SMS process: |
| 9) **FACS\_BulkLoad2Staging:** This procedure reads the flat file placed on 'L:\FACSSMS\FacsInputFiles\ ' (wasql01p.e-r-solutions.com**)** and populates the [Staging\_SMS\_Input\_From\_Facs] table row by row |
| 10) **FACS\_BulkLoad2Staging\_DeleteDuplicates**: Deletes records with duplicate cell phone numbers and retains the record with the lowest account number |
| 11) **FACS\_Get\_FACS\_File2Process**:  This populates the table [FACS\_File2Process] with the details of the latest load. The table holds details on timestamp on the flat file, file type, size (in bits) and file name |
| 12) **Get\_InBound\_SMS\_Response**: On execution, this would call the Twilio API to pull the responses sent back by the consumers SMS was sent to. This response is loaded to [InBound\_SMS\_Response]. |
| 13) **Get\_InBound\_SMS\_Status**:  Calls the Twilio API and gets the current delivery status and populates the table [InBound\_SMS\_Status] |
| 14) **Get\_PhoneList\_For\_Send\_SMS**: Joins the tables [Send\_SMS] and [F\_SMS\_Text\_ByState] to get the state code, SMSTextKey also, values to determine whether name-match service is to be used and whether to exclude based on score. Depending on the values selected the SMS run is determined. |
| 15) **Get\_Send\_SMS\_NameMisMatchSentToFACS**:- Gets name mismatch list from [Send\_SMS\_error\_history] and send it to FACS |
| 16) **Insert\_InBound\_SMS\_Response**:  Populates the [Inbound\_SMS\_Response] table with SMS responses sent by borrowers (from Twilio) |
| 17) **Insert\_InBound\_SMS\_Status**: Populates [InBound\_SMS\_Status] with Twilio status updates (Delivered, Undelivered, Sent, Queued) |
| 18) **Insert\_SMS\_ByState**: Used to create entries for new namespace, client class and SMSTextKey in [F\_SMS\_Text\_ByState] table |
|  |
| 19) **Insert\_SMS\_Send\_error\_History**: Populates [SMS\_Send\_error\_History] in case of any errors in sending SMS (Twilio error, not authorized by FACS or if name-match score for first/last name is less than threshold) |
| 20) **Insert\_SMS\_Send\_History**: [SMS\_Send\_History] gets populated with details of accounts that are okay to send SMS to (not in the [SMS\_Send\_error\_History] table) |
| 21) **Insert\_Staging\_to\_Send\_SMS**: Inserts records from the staging table to the [Send\_SMS] table |
| 22) **Send\_SMS\_Userlogin:** Update [UserLogin] table (holds usernames for access to portal) |
| 23) **Update\_InBound\_SMS\_Response\_ForFACS**:  A log file (with borrower response) is created by the ReadLogFile exe. This log file is then loaded to the [InBound\_SMS\_Response] table. |
| 24) **Update\_InBound\_SMS\_Status\_ForFACS**: A log file (with status from Twilio) is created by the ReadLogFile exe. Details from the log file are loaded to [InBound\_SMS\_Status] table. |
| 25) **Update\_Send\_SMS\_NameMisMatch\_Status\_ForFACS:**  Updates the [Send\_SMS\_error\_history] table when name-mismatch is sent to FACS with 1 on column NameMisMatchSentToFACS |
| 26) **Update\_SMS\_ByState:** Updates the message template on the **[**F\_SMS\_Text\_ByState] table based on Namespace, ClientClass, SMSTextKey and StateCode selections on the web portal |
| 27) **Update\_SMS\_ClientClass\_ClientName:** Allows for updation of Client name, Call back number, Account number prefix, whether to use the Name-match API and if to filter accounts based on name-match score based on Client Class selection from the portal. |
| 28**) Update\_SMS\_Send\_Status**:  Updates the fields BokuFirstNameScore, BokuLastNameScore, BokuCorrelationID, BokuResponse, SMSSentStatus, SmsSid, ResponseMsg, SMSSentDateTime   on the [Send\_SMS] table. |

## Using SQL Function

|  |
| --- |
| 1. **Fn\_ReplaceClientNamePhoneForSMS:** This function creates and formats the message body for the SMS   Campaigns |
| 1. **ReplaceASCII:** Replaces special characters with their ASCII equivalents and returns it to the procedure   Get\_Inbound\_Message\_Response   1. **Fn\_NameMatch:** Gets the first name and last name scores; if both are greater than 5, passes a 1, if either   is less than 5, passes a -1 |

Thank You

Visit – [Convergent USA](https://www.convergentusa.com/)

THIS DOCUMENT IS PROPRIETARY AND FOR INTERNAL USE ONLY