4.       Provide a **brief description of the use and development history** for the **proprietary technology described in document 10.2.1** in the data room (developed by employees, contractors, founders, acquired from a third party, etc.).

----------------------------------------------------------------------------------------------------------------------------------------

SMS

**Description**: This proprietary software is for sending SMS to borrowers on a daily basis. The process starts with the extraction of data from FACS and Latitude. It then goes through a pre-validation stage. There the records are validated for state; the last date an SMS was sent to that phone number or FACS account. The software checks if the phone number belongs to the name of the person in the FACS database. This is done by a web API call in real time to an outside company (BOKU). The web api call gives us the cell phone provider for the phone number and a score for the name match to the phone number. We use “short codes” to send the SMS where we have permission from the carriers, otherwise long codes are used.

The records are then reviewed by the “clicker” group. Here a person reviews each record and clicks it to be sent. The records presented to the “clicker” are only those which can be sent at that point in time. For example, at 8:00 am PST we cannot SMS to a person in Hawaii. The SMS program checks with FACS in real time to make sure that the account is still in the correct disposition, before sending it. This is done in real-time using web api.

A custom web hook running on our servers captures status data sent by cell phone providers. This data is updated in our database and sent over to FACS in real time.

At present we have 3 types of messages we SMS.

1. We are Convergent Outsourcing, Inc. Please call us at {{PhoneNumber}}. Regarding <Client Name>, Convergent Account #: {{AccountNumber}}.
2. Please contact Convergent Outsourcing, Inc regarding an issue with your recent scheduled payment. Our phone number is {{PhoneNumber}}. Your <Client Name> Account, Convergent Account #: {{AccountNumber}}.
3. Convergent-Hello! Reminder that your payment for <Client Name> of $xxx.xx will be processed on zz/zz/zzzz. Convergent Account #: {{AccountNumber}}.

The text of the messages is maintained by operations. We have written a web app for them to change and maintain the text of the messages.

For latitude we have an Account Summary panel**-**A Latitude panel used for taking Borrower consent for the email, SMS and Calls.

* A custom module to create SMS text of various types for borrowers, in both V11 and V10 Latitude instances.
* A Webhook to collect Borrower response for V11/V10.

All communication with FACS, Latitude, Twilio and Sql server is in real-time using the REST API architecture.

**Development history**: Development for ACT SMS started in 2018 at the request of the operations group. Development for COI started in June 2019. The SMS API provider is Twilio Inc. When starting the SMS project for ACT we checked with a few providers and decided on Twilio Inc, because of their rich API set. The code base is written in C# and Microsoft Sql (TSQL). It runs on windows software and uses the IIS server. The APIs for FACS are custom written on the FACS server.

**Developers**: The entire software design coding and architecture was done by our in-house development team located in U.S. and India. Kevin Haworth was responsible for writing FACS web APIs. The India team (Veerendra, Neeraj, Vivek) were responsible for writing the code running on Windows.

**Email**

**Description:** This propriety software is for sending email from FACS and Latitude systems to borrowers daily. The process starts with the extraction of data from FACS and Latitude. The records are validated for state code and the email address. The email address is validated using a third-party web api, called by the email validation program. To increase open rates the account is scored using an AI based scoring model. The model determines if the chance of the person opening the email is high.  Using this information, the email program controls the volume of emails sent on different domains. This helps with 2 of our domains in having high a reputation that contributes to better a delivery rate.

A custom web hook running on our servers captures status data sent by SendGrid for each email. This data is updated in our database and sent over to FACS in real time.

The state language for the emails is maintained by operations. We have written a web app for them to change and maintain the text of the “state language”.

**Development history**: Development for email using SendGrid started in Aug 2019 at the request of the operations group. Previously some emails were being sent by an outside contractor (RevSpring). It was decided to bring that email sending process in-house as it was cheaper for us to send those emails. Also by doing the emails in-house the process changing the email templates would speed up. It was also decided to switch to emails from postal mail for Validation notices, validation with settlement offer notices, settlement offer notices, contact us notices, NSF (non sufficient funds) notice and card decline notices. I the receiver of the email did not open the email then a postal letter would be sent to them. This effort was to reduce the cost of communication with borrowers. The email API provider is SendGrid Inc. The code base is written in C# and Microsoft Sql (TSQL). It runs on windows software and uses the IIS server. The APIs for FACS are custom written on the FACS server.

**Developers**: The entire software design coding and architecture was done by our in-house development team located in U.S. and India. Kevin Haworth was responsible for writing FACS web APIs. The India team (Anupam, Neeraj and Vivek) were responsible for writing the code running on Windows.

**IVR**:

**Description** : The **MainIVR** collects payments from the caller; announces the mailing address and the web address if the caller so desires. The payments collected are either full or partial payments payable that day. The payments can be paid by credit cards (Visa, Mastercard or Discover) or by eCHeck (ACH). The information about the account is collected in real time from FACS using a web API. The payment information is also updated in real time within FACS using a web api.

The logs for each call are downloaded from Twilio and updated into a SQL database. These logs are moved to SnowFlake (data warehouse) daily for reporting purpose. These logs can also be used to spot check the calls that came in.

The IVR has no voice capability. The input from the caller is collected based on the keystrokes the caller makes on their phone's keypad.

**Survey IVR -** The IVR was written to collect survey data from the caller after the call with the agent has ended. The call is transferred to a Twilio (IVR) number from Avaya. The survey data collected by the IVR is stored in SQL database and reporting of statistics to the client is provided from that database. The client using this currently is Keefe.

**SecurePayment IVR -** This IVR collects the credit card or Bank information and stores it in FACS in real time. The need for this IVR arose due clients (Verizon) not wanting agents working from home to collect credit card information on their customers.

The process works as follows:

* The caller talks to the agent and decides on the number of payments and the payment method. Then they initiate a 3-way call with the IVR (agent, caller, IVR). The IVR authenticates the caller account information in real time with FACS. Then collects the credit card information. The agent is not able to hear the input the caller is making using the phone keypad. The credit card information is update to FACS in real time. The agent then works on the account with the caller.

This IVR is used by the third-party collection group at COI.

**Development history:**

The effort of developing a IVR started in May-June of 2020. We went through a proof of concept with Twilio Inc and LiveVox. Twilio was chosen because of the flexibility of their interface and ease of maintaining the code base. LiveVox had limited flexibility (their offering was take it or leave it out of the box approach). Twilio also had all the features like converting text to voice, the features available on AMAZON AWS, since it runs on that platform.

The initial development was done using their platform called “Studio”. Web APIs were written by the in-house dev team for “Studio” to be able to talk to our FACS and SQL database. Coding to make decisions was written on Twilio in Node JS using their interface to AWS lambda functions.

**Developers:** The entire software design coding and architecture was done by our in-house development team located in U.S. and India. Kevin Haworth was responsible for writing FACS web APIs. The India team (Anupam, Neeraj, Veerendra) were responsible for writing the code running on Windows and Twilio platform.

**IVA**:

**Description** : **-** This IVA collects payments from the caller, announces the mailing address and the web address if the caller so desires. The payments collected are full or partial payments payable that day. Payment arrangements with the first payment starting at a future date can also be made using this IVR.  Payments can be arranged on a weekly, bi-weekly or monthly basis. The IVR calculates the payment amount for each payment, so it adds up to the full amount due. The payments can be paid by credit cards (Visa, Mastercard or Discover) or by eCHeck (ACH). The information about the account is collected in real time from FACS using a web API. The payment information is also updated in real time within FACS using a web API.

The logs for each call are downloaded from Twilio and updated to a SQL database. These logs are moved to SnowFlake (data warehouse) daily for reporting purpose. These logs can also be used to spot check the call information that came in.

Each call is recorded. The recording is turned off when the flow reaches the point of collecting credit card or Bank account information. The recording starts after that information has been collected. A copy of the recording is kept on the cloud with Twilio, where a front-end web app resides allowing operations the ability to monitor call recordings.

The IVA has both voice recognition capability as well as input based on keystrokes. The “autopilot” technology figures out the “intent” of the caller and directs the flow to the appropriate task.

**Development history:**

The effort of developing an IVA started in November of 2020. The goal here was to give a better user experience than the IVR to the user/caller. We went through a proof of concept with Twilio Inc and DabbleLab (a firm hired to train our developers in Twilio Autopilot). Autopilot is the product on Twilio which figures out the intent of the caller based on what they are saying. The flow of the call is then directed based on the intent of the caller, as interpreted by the Autopilot BOT. The Autopilot BOT is trained to interpret the intent as more data is collected from the callers.

The BOTs created in Autopilot were connected to one another in “Studio”. Web APIs were written by the in-house dev team for Autopilot to be able to talk to our FACS and SQL database. Coding to make decisions was written on Twilio in Node JS using their interface to AWS lambda functions.

**Developers:** The entire software design coding and architecture was done by our in-house development team located in U.S. and India. Kevin Haworth was responsible for writing FACS web APIs. The India team (Anupam, Neeraj, Veerendra and Vivek) were responsible for writing the code running on Windows and Twilio platform.

**Veritax**:

**Brief Description**: This is used for downloading the borrower’s last 4 years of Income, if the borrower gives consent. It is used by collectors in Latitude to confirm the income a borrower has, so the payments can be calculated properly. The process is order creation on the Veritax site using web apis. Veritax asks the consent from the borrower. If the borrower gives consent, we get a message on our web hook to download the documents.

**Developers:** The entire software design coding and architecture was done by Microware. Currently it is being maintained by our India development team member Vivek.

**SignNow**:

**Brief Description:** We need signed documents from the borrower for his consent and payment plan. Signnow gives us the ability for the borrower to receive the document in their email and e-sign and return the document. The software is a plugin to Latitude and communicates with SignNow web apis to accomplish this.

**Development history:** Before this software was created the collector had to print and mail these documents to the borrower. With this software the collectors can get the signed documents while they are on the phone with the borrower.

**Developers:** The entire software design coding and architecture was done by Microware. Currently it is being maintained by our India development team member Vivek.