Virgil®

Architectural Summary

Version 3

September 16, 2015

Table of Contents

Background

The Virgil® platform provides a common set of operational capabilities and establishes a service-based technical architecture for all MedAvante business offerings. The platform goal is to enable MedAvante to offer a continuum of products and services that can be selected and configured to meet specific needs of clients and sponsors.

Architecture

App-services Model

At the most abstract level, the Virgil architecture is based on the *app-services model* (historically called the *client-server model*). In this model, *apps* (i.e., applications or clients) make requests of *services* in order to accomplish work. Services can satisfy the request directly or make requests of other services, as needed. How a service satisfies a request is unknown and irrelevant to the requestor.

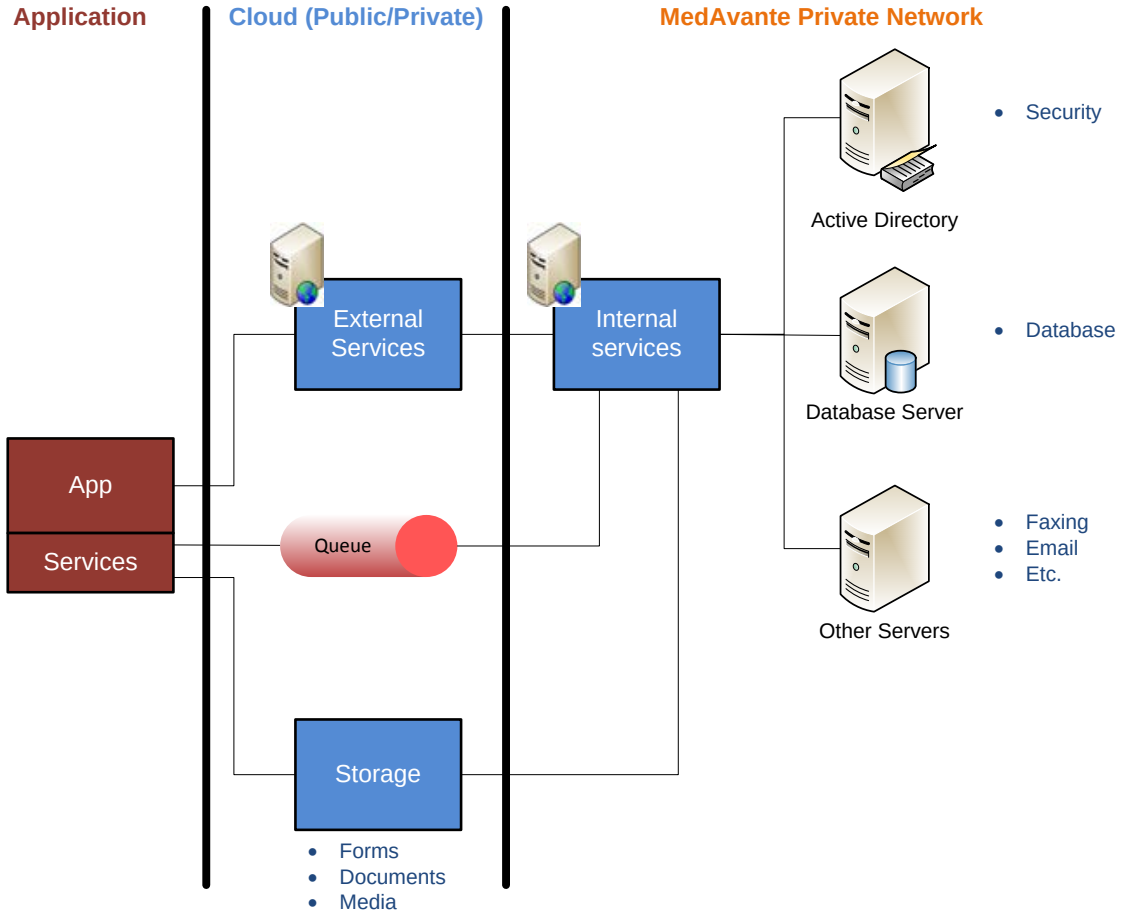
This decoupling (isolation) of functionality is a core strength of the architectural model. Platform capabilities can be added or modified in relative isolation from other components with minimal impact on existing functionality.

The most well-known use of this architectural style is the World Wide Web. In this case, the app is the user's browser and the service is the web server holding the web page or document requested by the browser. As the user interacts with the web pages, the web server makes requests of backend services to realize the web page functionality.

There are, of course, many more details to the app-services model (e.g., sending the request, verifying the user is authorized to make the request or see the result) but the basic concept is quite simple. An app makes a request to a service, and the service does something in return (e.g., return a web page, perform a calculation and return the result, update a database with study information).

Distributed Implementation

Virgil delivers its functionality through a three-tier implementation of the app-services model as shown in the diagram below.



- Application Layer – provides the user-facing functionality for Virgil.
  - All subject data and files at this layer are encrypted.
  - Applications only communicate with External Services. Where necessary (e.g., SCN) the application may be deployed with services to communicate with cloud-based storage and queues.

- Cloud Layer – contains storage, queues, and external services.
    - Storage is cloud-based (i.e., outside the MedAvante network).
        - Only files (e.g., forms, documents, and media) are stored in this layer.
        - All files at this layer are encrypted.
    - Queues are cloud-based (i.e., outside of the MedAvante network)
        - Queues are used for asynchronous communication between external and internal services.
    - External Services are managed by MedAvante but are isolated from the MedAvante network (i.e., private cloud)
        - External Services are the point of communication for all Virgil applications in order to enforce security through isolation of externally facing capabilities and capabilities that reside inside the MedAvante network.
        - Virgil applications can only make requests to the External Services that reside in the cloud layer; they cannot communicate directly with services inside the MedAvante network.
        - External Services perform no work; they only serve as the intermediary between the App and the Internal Service.
        - The sequence of operation is:
            1. The App makes a request of an External Service.
            2. The External Service relays that request to an Internal Service.
            3. The Internal Service satisfies the request (i.e., performs work).
            4. The Internal Service returns the request response (e.g., data, success flag) to the External Service.
            5. The External Service relays the request response to the App.

- MedAvante Private Network Layer – contains platform functionality and data storage.
    - All platform data (i.e., databases) is at this layer.
    - All user and application security services and data are at this layer.
    - Subject data and files at this layer are not encrypted.
    - Internal Services reside at this layer and provide the bulk of the Virgil platform functionality

Note: In order to simplify the process flow descriptions in the remainder of the document, references to service calls will assume successful service calls and only specify the function required from the service, not the details of the external/internal service communication. So, for example,

SCN requests data transfer from the Data Service.	*instead of*	1. SCN requests data transfer from the external Data Service. 2. The external Data Service communicates the request to the internal Data Service. 3. The internal Data Service performs the requested action and passes the result (response) back to the external Data Service 4. The external Data Service relays the request response to SCN.

Applications

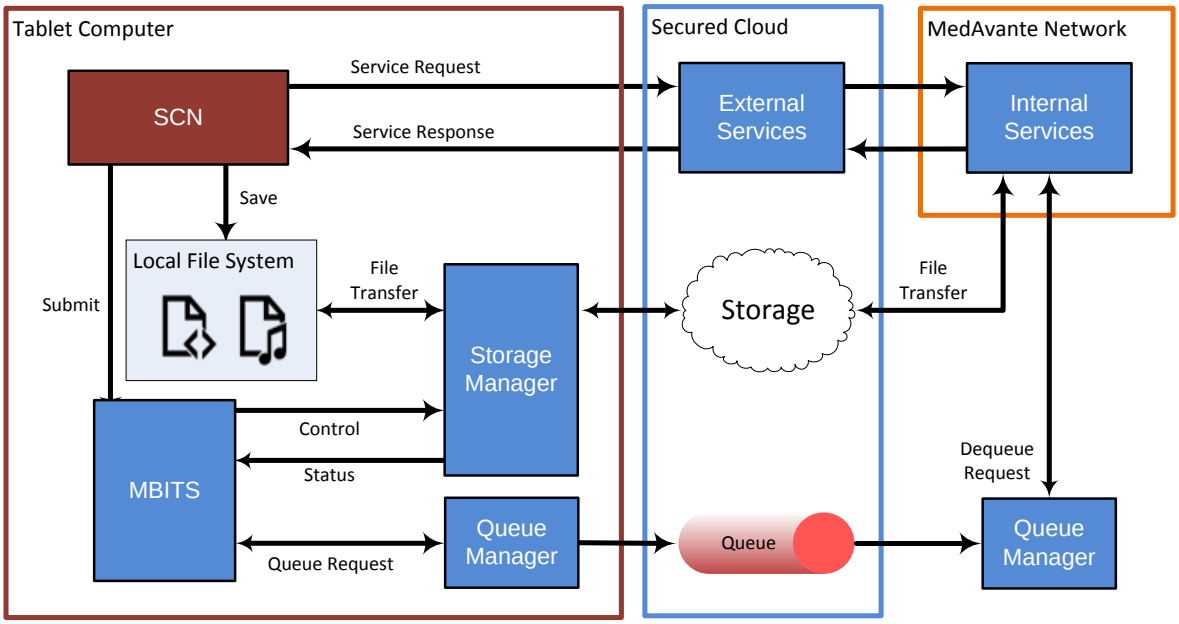
Users interact with the Virgil platform through applications. The following sections describe two applications that are key components of the current development effort (i.e., the Site Clinician Application and Virgil Portal). Planned applications are described briefly at the end of this document.

Site Clinician Application

In the Virgil platform, the site clinician performs assessments using the Site Clinician Application (SCN) on a tablet computer. While the bulk of the site clinician functionality is provided by the SCN application, the tablet also contains three services that cooperate to deliver the full range of required functionality to the tablet.

- MBITS (for MedAvante Background Intelligent Transfer Service) – coordinates form/file transfers with the Storage Manager Service and uses the Queue Manger Service to notify other Virgil components that forms have been submitted and are available for processing.
- Storage Manager Service - provides electronic form and file transfer capabilities to/from cloud-based storage.
- Queue Manager Service – provides asynchronous communication between SCN and other Virgil components.

A visualization of the tablet components is shown below to highlight significant platform components.



Leaving out some administrative/setup tasks (e.g., creating subjects and subject visits), the basic site clinician interaction can be described as follows:

1. Initial startup and login
    a. The user powers up the tablet and the file transfer service (MBITS) is automatically started.
    b. The user launches the SCN application.
    c. SCN displays the login screen and the user provides their account username and password (a.k.a., user credentials).
    d. SCN requests the Security Service to authenticate the user, passing the user credentials.
    e. The Security Service determines that the user has provided valid credentials.
    f. The Security Service responds to the original request and passes an encrypted security token back to SCN, indicating successful authentication.
        - The security token includes the user-provided credentials along with all of the user's study and site authorizations. The security token contains an expiration date in order to support offline authentication (e.g., a user can login to the app while offline for a period of time. After that, online authentication must be used to "refresh" the security token).
        - The service response would be an error flag if the user had provided improper credentials.
    g. SCN downloads any new or updated form templates, based on the user's study authorizations.
    h. SCN downloads any forms that have been assigned to the user for editing.
    i. SCN displays the study visit list (or other application screens, as appropriate).
2. Conduct assessment
    a. The user selects a form for a subject visit and starts the assessment.
    b. The user conducts the assessment, completing the form and, as configured, capturing assessment recordings (audio or video). All forms and recordings are saved to the local (tablet) file system in encrypted form, ensuring the information is only intelligible inside Virgil applications.
3. Submit assessment form
    a. The user submits the assessment form (after providing an electronic signature).
    b. SCN requests MBITS to transfer the submitted form and associated recording.

- At this point, the user may close SCN without impacting the form/file transfer and notification processes.

c. MBITS requests the Storage Manager Service to copy the submitted forms and files to the cloud-based storage (using an encrypted transfer protocol).

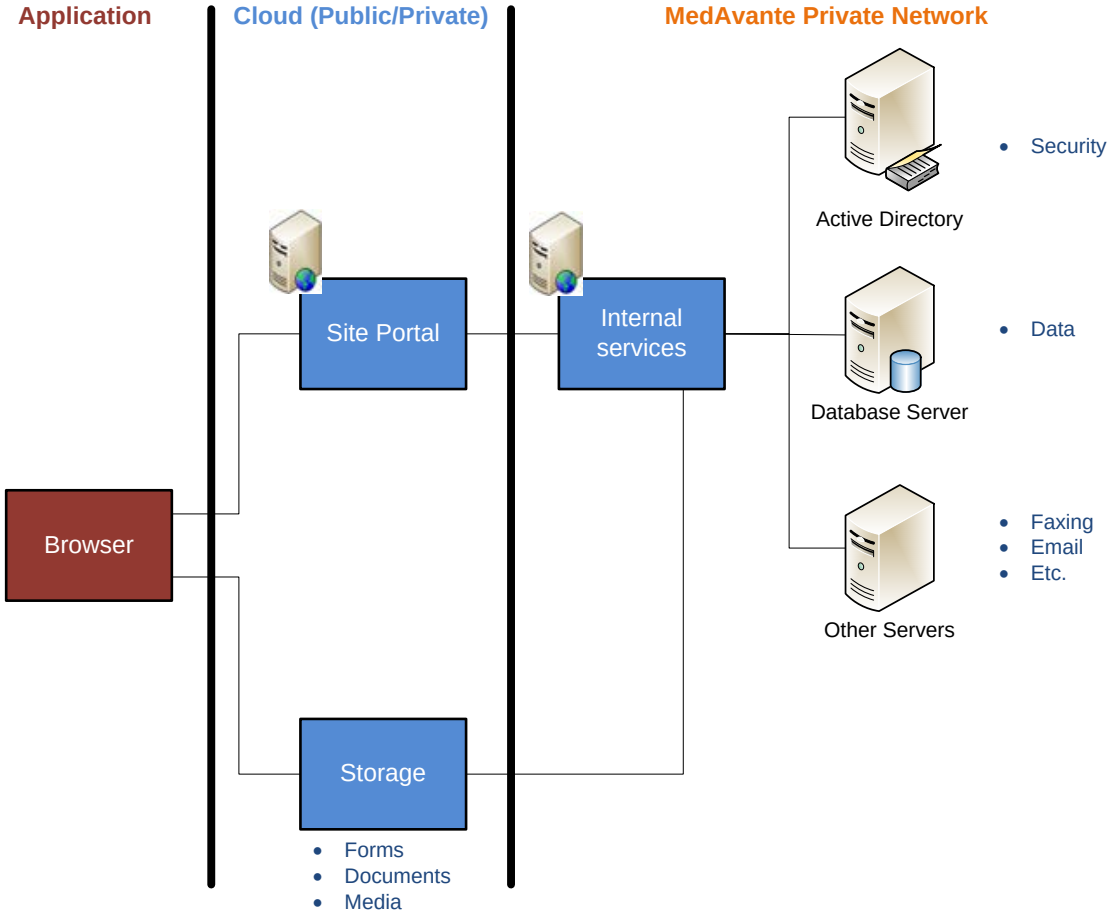
d. MBITS requests the Queue Manager Service to place a notification of the availability of submitted forms and files on the cloud-based queue.

e. The Data Hub service retrieves the notification from the queue and coordinates processing of the submitted forms (see *Example 2: Process a submitted form* below).

Virgil Portal

The Site Clinician Application supports the acquisition and submission of subject visit assessment data but it does not provide any way to view the submitted data/files, or to perform broader study-related activities. These site clinician requirements are addressed in the Virgil Portal.

The Virgil Portal is a web-based application that can be accessed via a web browser on any computer, including the Virgil tablet itself. The Virgil Portal resides at the cloud layer and is responsible for all requests to the internal services, effectively replacing the External Service component in the client-based architecture described above. A logical diagram is shown below.



The Virgil Portal provides the following high-level capabilities (partial list):

- Study Analysis/Overview
    - View Study Profile
    - View Study/Site Raters
    - View Subjects
    - View Subject Visits
    - View Assessments
- Assessment Management
    - Create/update/delete subject
    - Create/update/delete/enable/disable observer
    - Create/update/delete subject visit
    - Assign/reassign subject visit forms
    - Upload supplemental files to a subject visit (e.g., paper-based forms)
    - Manage and transcribe paper-based assessments
    - Create/delete/respond/close data queries
- Assessment Review
    - View submitted forms
    - Access submitted recordings (audio and video)
    - Access MedAvante-supplied central review feedback
- Virgil Platform Administration
    - Manage Studies
    - Manage Forms
    - Manage People
    - Manage Organizations

Services

The table below provides a description of the core services of the Virgil platform. The reader is directed to the Operational Examples section of the document for insight into how the services are orchestrated to provide business value.

Name	Provides functionality to:
Security Service	Determines who has access to the system (authentication) and the types of access they have (authorization). There are two implementations of this service. • Web-based Internal Service that handles all online authentication and constructs the encrypted security token for use on the tablet. • Tablet-based DLL that is used for authentication when the tablet is offline.
MBITS	Coordinates transfer of files from the tablet to/from cloud-based storage. Posts notification for other Virgil components of the availability of submitted forms.
Data Hub Service	The central processing and workflow component of the Virgil platform. Performs standard post-submission form processing: • Requests data extraction from forms and saves the data in database • Requests generation of standard PDF form renditions • Orchestrates all study-specific post-submission workflows such as central review.
Storage Manager Service	Responsible for uploading and downloading files from Cloud Storage.
Form Manager Service	Responsible for actions on forms: • Generates PDF rendition of form • Extracts data from form
Queue Manager Service	Queues and de-queues messages (providing asynchronous, guaranteed delivery messaging).

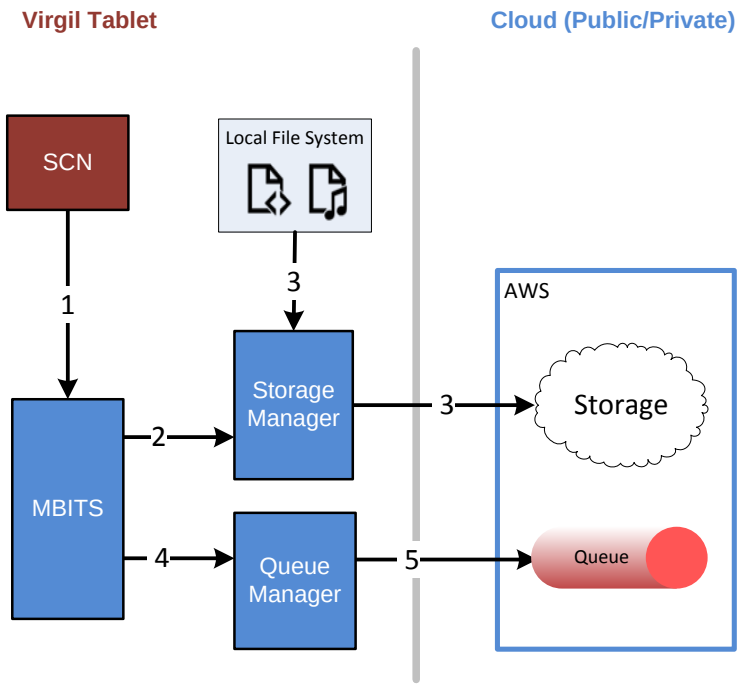
Operational Examples

This section provides descriptions of how Virgil components collaborate to provide business value.

Example 1: Submit a form

This example provides additional detail on the process described in the *Site Clinician Application* section above.

Example 1: Submit a form



The user submits the assessment form (after providing an electronic signature). Then:

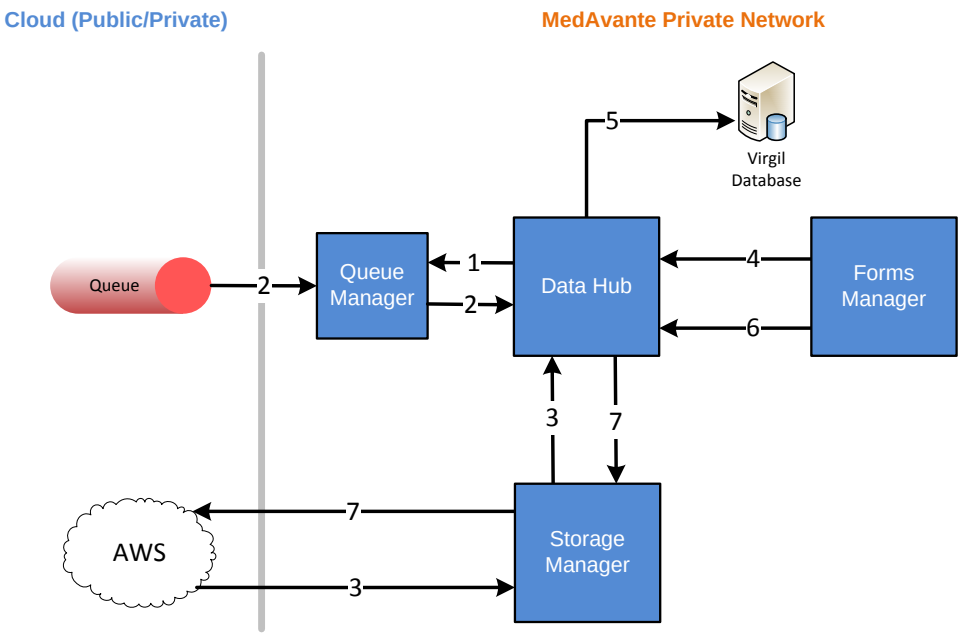
Step	Action
1	SCN requests MBITS to transfer the submitted form and associated recording. At this point, the user may close SCN without impacting the form/file transfer and notification processes.

Step	Action
2	MBITS requests STORAGE MANAGER to copy the submitted forms and files to the cloud-based storage
3	STORAGE MANAGER uses an encrypted transfer protocol to transfer the files from the local file system to Cloud Storage. The stored files are encrypted.
4	MBITS requests the Queue Manager Service to notify the platform of the availability of submitted forms and files
5	QUEUE MANAGER places the form submission notification on the cloud-based queue. The notification includes a key to the forms and files that were included in the submission.

Example 2: Process a submitted form

All submitted forms are processed with a standard set of actions, irrespective of any study-specific optional services.

Example 2: Process a submitted form



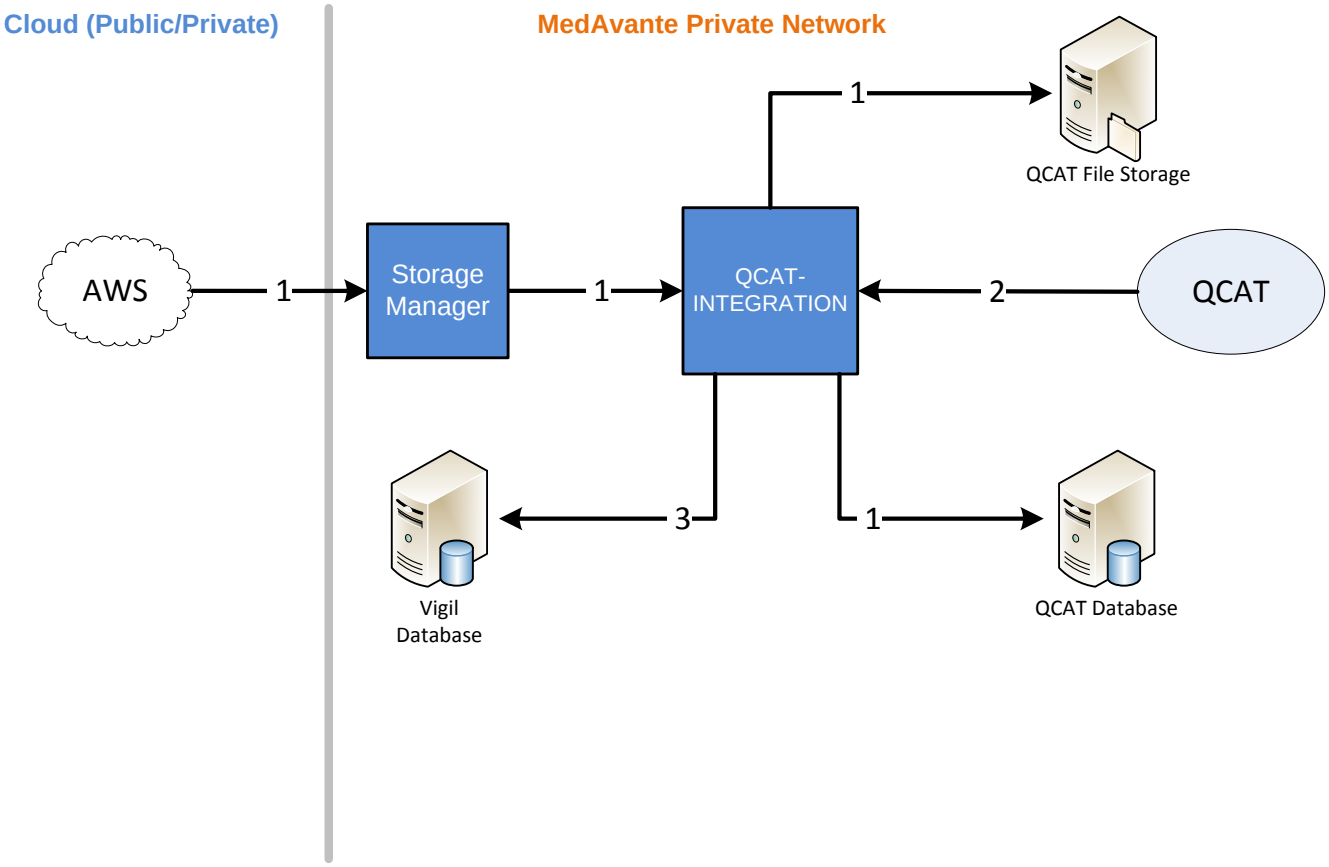
Step	Action
1	DATA HUB periodically requests QUEUE MANAGER to check the queue for messages.
2	QUEUE MANAGER retrieves the request from the queue.
3	DATA HUB requests STORAGE MANAGER to retrieve the form from Cloud Storage (which it does, bringing a copy into the MedAvante private network).
4	DATA HUB requests FORMS MANAGER to extract data item values from the form (which it does, passing the values back to DATA HUB).

Step	Action
5	DATA HUB stores the extracted data item values in the Virgil Database.
6	DATA HUB requests FORMS MANAGER to create a PDF rendition of the submitted form (which it does, passing the file back to DATA HUB).
7	DATA HUB requests STORAGE MANAGER to save the PDF rendition to Cloud Storage (which it does, completing the standard form processing workflow).

Example 3: Execute central review

This example describes what happens to a subject visit when the study has been configured for central review.

Example 3: Execute central review



QCAT-INTEGRATION periodically queries the Virgil database for study visits that are ready for central review. Then:

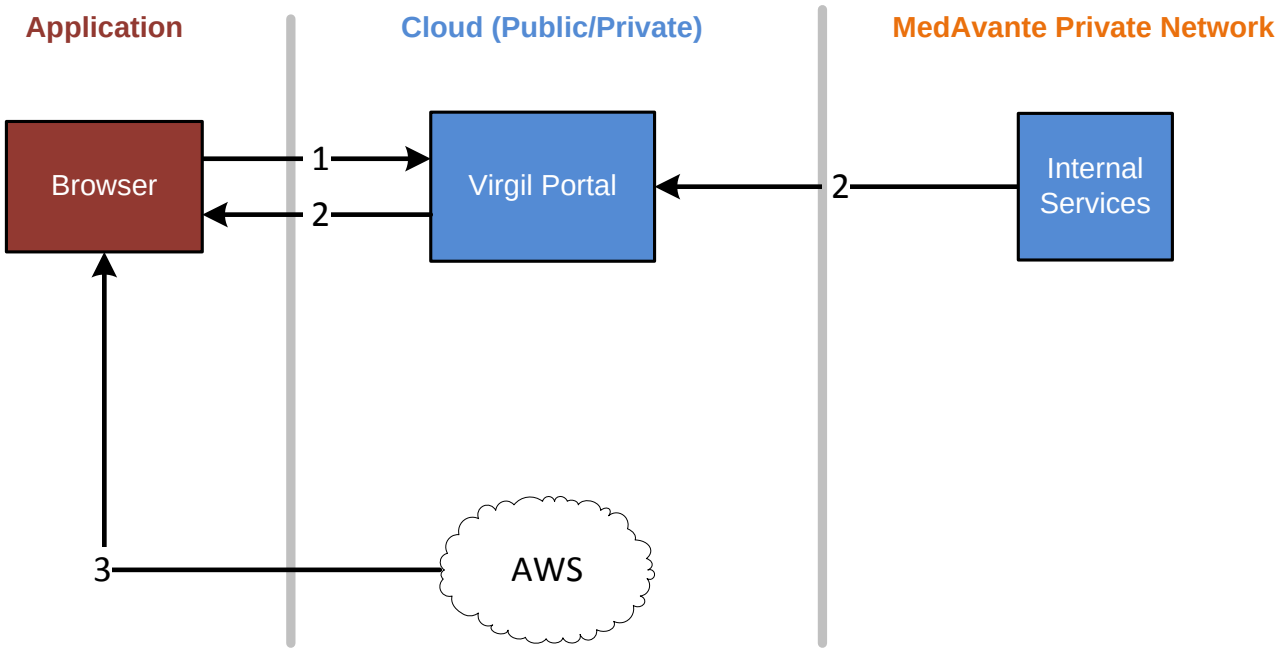
Step	Action
1	QCAT-INTEGRATION retrieves the files and creates the required database entries in the QCAT Database for the subject visit).

Step	Action
	(Not Shown) Existing QCAT capabilities are used to process the subject visit scales through review and feedback submission.
2	QCAT notifies QCAT-INTEGRATION that review has been completed on the subject visit
3	QCAT-INTEGRATION creates the required database entries in the Virgil Database for the central review.

Example 4: View an assessment form

This example describes what happens when a site clinician browses to an assessment form. (The same actions would occur with any file, including audio recordings).

Example 4: View an assessment form



Step	Action
1	User selects to view the form (i.e., the PDF rendition) in the Virgil Portal.
2	Virgil Portal requests a URL for the file, from internal services, and returns it to the BROWSER. The URL contains authentication keys to allow the user access to the file
3	BROWSER requests display of the URL (in Cloud Storage).

Step	Action
	(Not shown) The request is redirected to the closest Cloud Storage server for performance reasons. For details see *Streaming Files from Cloud Storage* below.
4	File is streamed to the BROWSER using an encrypted protocol.

Other Components

Cloud Storage

The cloud storage component of the Virgil platform is hosted on Amazon Web Services (AWS). The platform is has been developed to minimize the difficulty of changing the cloud service provider in the future but there is no reason to assume this will be necessary.

AWS is used to store all Virgil platform files. (All data resides in databases inside the MedAvante private network.)

Logically, there are four functions provided by AWS:

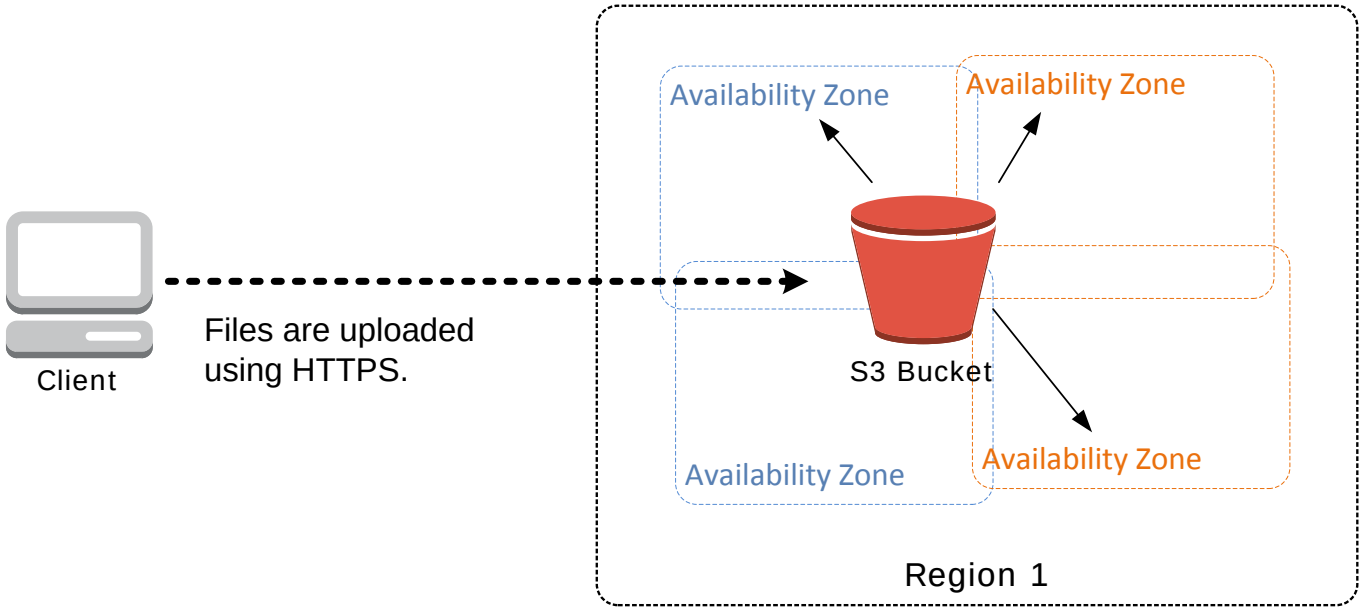
- Save file – physical movement of a file into the cloud.
- Retrieve file – physical movement of a copy of a file out of the cloud.
- Queue/dequeue message – asynchronous delivery of inter-service messages.
- Access (i.e., view/listen to) file - streaming of file contents to a user. Streaming is used for efficiency and to ensure that a copy of a file never exists outside of managed central storage (where it is encrypted).

Virgil uses three AWS services

- S3 (Simple Storage Service) – for file storage
- SQS (Simple Queue Service) – for queuing and dequeuing of messages
- CloudFront – for streaming of file content (i.e., viewing/listening to a file)

The reader is directed to on-line AWS documentation for details on these services. Two Virgil-specific scenarios are provided below with AWS service-level detail.

Save File to Cloud Storage



- Files are encrypted at rest using S3 Server Side Encryption (SSE).
- Data is replicated to each availability zone in the region
  - (2 in VA, 2 on West coast)

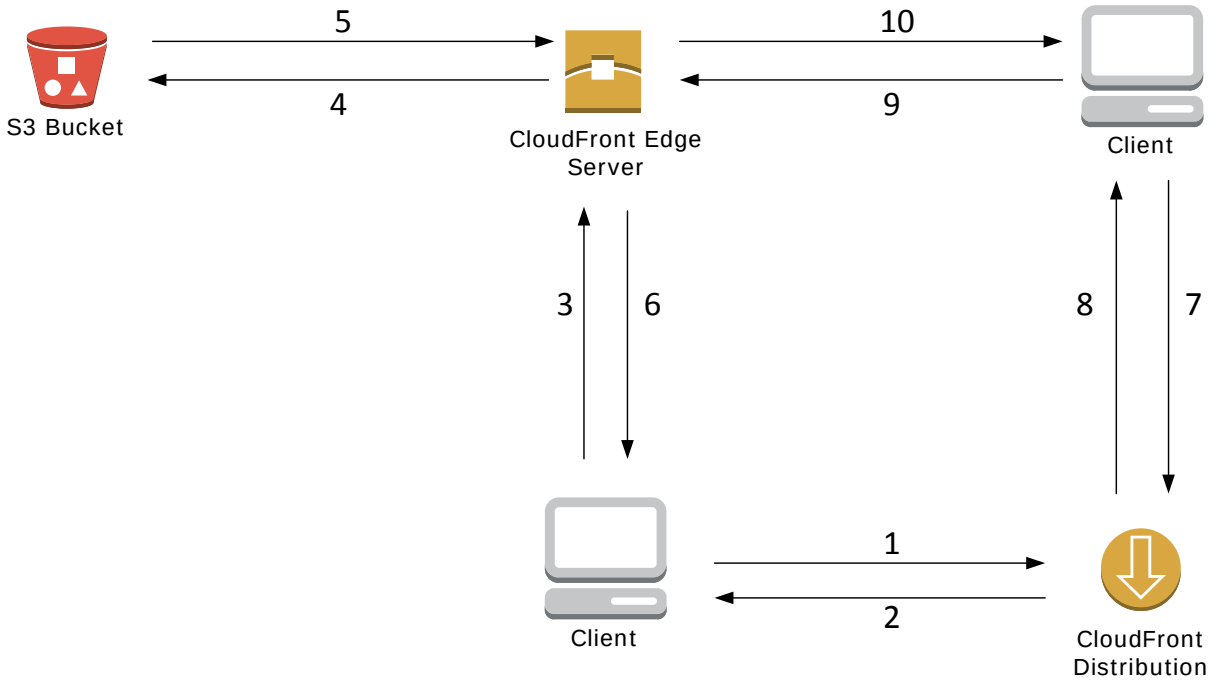
When files are loaded into S3 they are automatically copied to different servers in different geographic locations to maximize data availability.

File loss is only possible with simultaneous failure of multiple devices at four geographically separate data centers. Amazon claims 99.999999999% durability (i.e., 1 chance in 100 billion).

Virgil takes additional steps to ensure even more file security, backing up all files in a separate S3 availability zone.

Streaming Files from Cloud Storage

Operational Example 3 above describes how files are streamed to a user. A more detailed description of what happens after the client requests the file (URL) is now presented.



Steps	Action
1, 2	Client requests file from CloudFront distribution
3	DNS redirects client to the closest CloudFront Edge Server
4, 5	Edge Server checks local cache for the file; file does not exist so it is retrieved from S3 Bucket (transfer occurs over Amazon high-speed private network)
6	File is streamed to client using an encrypted protocol (RTMPE or HTTPS)
7, 8	A subsequent client request, from same region, is redirected to same Edge Server
9, 10	Edge server checks local cache for the file; finds it and streams the file to the client