OFR Data Security

# Overview

Leap Orbit and CRISP are implementing an Overdose Fatality Review (OFR) workflow solution for the Maryland Department of Health Behavioral Health Administration. The solution will:

1. Receive CSV files with OCME case data via SFTP upload.
2. Parse the files and store the parsed data in a database.
3. Provide a website where users can access, review, assign, and update the data.

The OFR solution is built on the Microsoft Azure cloud platform, and will be deployed within CRISP’s existing Azure infrastructure. CRISP will maintain full control over access to the Azure resources and to the data.

# Storage Types

The OFR solution utilizes two different types of storage: Azure Storage Blobs and CosmosDB. In addition, data is briefly stored on an FTP Server.

## Azure Storage Blobs

The CSV files are stored as Blobs in an Azure Storage account for archival purposes.

### Encryption at Rest

The Blob Service of the Storage Account is set to use Storage Service Encryption, so all files are always encrypted at rest. For more information on Azure Storage Service Encryption see <https://docs.microsoft.com/en-us/azure/storage/storage-service-encryption>.

### Encryption in Transit

The Storage Account is set to require secure transfers, so requests to send or receive data to/from the Storage Account must always use secure connections (HTTPS). For more information see <https://docs.microsoft.com/en-us/azure/storage/storage-security-guide#encryption-in-transit>.

### Authentication

Access to the Storage Account requires a 512-bit Access Key. The OFR solution needs the Access Key in order to save the files. CRISP can control which users, if any, receive the Access Key. There are two active Access Keys at a time. Microsoft recommends regularly updating (regenerating) the Access Keys by 1) switching all resources to use the second Access Key, 2) regenerating the first Access Key, 3) switching all resources back to use the new first Access Key, 4) regenerating the second Access Key. For more information see <https://docs.microsoft.com/en-us/azure/storage/storage-security-guide#data-plane-security>.

### Authorization

There is no specific Authorization needed for access to Blobs. All authenticated users can access all Blobs.

## CosmosDB

The parsed data is stored as JSON documents in CosmosDB. Authenticated users can access the CosmosDB documents via a website to view and modify them.

### Encryption at Rest

All data in CosmosDB is always encrypted at rest. This is not a setting that can be enabled or disabled. For more information, see <https://docs.microsoft.com/en-us/azure/cosmos-db/database-encryption-at-rest>.

### Encryption in Transit

All interactions with CosmosDB require encryption at the transport layer. This is not a setting that can be enabled or disabled. For more information see <https://docs.microsoft.com/en-us/azure/cosmos-db/database-security>. The website also requires secure connections, so all data sent back to end users is encrypted.

### Firewall

CosmosDB includes a firewall which, when enabled, blocks all IP addresses by default. Specific IP addresses or ranges of IP addresses must be added to allow access. The IP address of the OFR solution’s Web App must be allowed access through the firewall. CRISP can configure which additional IP addresses are allowed through the Firewall, if any. For more information see <https://docs.microsoft.com/en-us/azure/cosmos-db/firewall-support>.

### Authentication

Website users are authenticated via Active Directory B2C. This authentication is only for access to the website itself, not to CosmosDB. The API back end of the website uses a Master Key for access to CosmosDB. For more information on Master Keys, see <https://docs.microsoft.com/en-us/azure/cosmos-db/secure-access-to-data>. CRISP can utilize its existing Active Directory to grant users direct access to CosmosDB, if necessary.

### Authorization

Because it uses a Master Key, the OFR solution has full authorization to CosmosDB. CRISP can control the roles and permissions of any additional CRISP Active Directory users to whom it chooses to grant access.

## FTP Server

An Azure Virtual Machine running the third-party product CrushFtp allows users to upload files via SFTP. Uploaded files are stored on a network attached drive that is part of the Azure Storage Account’s File Service. Files only exist briefly on the attached drive. Upon completion of an upload, CrushFtp makes an API call informing that a new file is available. The API code copies the file to Blob Storage and deletes it from the File Service (and thus from the FTP server’s attached drive).

### Encryption at Rest

The File Service of the Storage Account is set to use Storage Service Encryption, so all files are always encrypted at rest. For more information on Azure Storage Service Encryption see <https://docs.microsoft.com/en-us/azure/storage/storage-service-encryption>.

### Encryption in Transit

1. User to server: CrushFtp is configured to only allow SFTP connections. Regular (unsecured) FTP connections and FTTPS connections are not accepted.
2. Server to storage: Because the File Service is part of the same Storage Account as the Blobs, it also utilizes the Storage Account setting requiring secure transfers.

### Authentication

1. User to server: CrushFtp is integrated with the same Active Directory B2C which provides authentication to the OFR Website. Users must provide valid credentials to be allowed to upload files.
2. Server to storage: The File Service uses the same 512-bit Access Key as is used for Blob Storage. The VM provides this key when it attaches the network drive. When the Access Key is changed, the drive must be reattached to the VM.

### Authorization

1. User to server. No specific Authorization is needed. All authenticated users can upload files.
2. Server to storage: No specific Authorization is needed for access to the File Service. All authenticated users can access it.