Client-Managed Synchronous Key-Based Data Encryption

Draft – Company Confidential

Navaera Sciences, LLC

November 23, 2020

Table of Contents

Table of Figures

Revision History

Author	Date	Reason For Changes	Version
Glassman, Jacob	10/19/2020	Initial Draft Completed	1
Wang, Michael	11/3/2020	Draft Updated with Corrections	2
Freund, Caroline	11/17/2020	Draft Updated with Corrections	3

Notice to Readers: This document discusses a feature that is currently experimental, or planned for implementation. This document should not be interpreted as a user guide, or specification as of its current draft version.

The remainder of this page was intentionally left blank

1. Introduction

In recent years, clients have requested additional capabilities for Navaera On-Demand that support:

1. The ability of a client to manage an encryption key that would protect data, specifically Personally Identifiable Information ("PII") that are stored within data sources that support a Navaera On-Demand or Navaera On-Premise solution; and

2. The ability to encrypt specified fields of data using an encryption key that is generated by the client, and managed using the function described in (1) above.

In this document, we will discuss the implementation of a new feature for Navaera On-Demand for *Client-Managed Synchronous Key-Based Data Encryption* that is designed to provide clients with the aforementioned capabilities.

2. Current Encryption Capabilities in Navaera On-Demand

It is important to note that Navaera On-Demand refers to Navaera's Software as a Service ("SaaS") platform, and Navaera On-Premise refers to Navaera solutions that are packaged for deployment on client infrastructure. However, the same applications are deployed in either delivery format though they may be configured differently. Since this is the case, the features discussed in this document should be considered to be available in both solutions deployed on the SaaS platform, as well as those that are deployed on client infrastructure. Throughout the remainder of this document, we will refer to 'Navaera solutions' to discuss the encryption architecture for Navaera On-Demand and Navaera On-Premise solutions holistically.

Currently, Navaera solutions rely on 'data-at-rest' encryption technologies that are external to the actual applications. Examples of which include transparent database encryption, file system encryption, and full disk encryption. While these technologies do support all statutory, regulatory, and functional needs related to encryption for data at-rest, it would not be feasible to enable clients to manage encryption keys that support these methods as such integration would require either additional connectivity or manual intervention from Navaera staff in the case of Navaera On-Demand.

2.1 Existing Key Management and Protection Infrastructure

Navaera maintains an existing key management infrastructure that uses nCipher HSM technology for the storage of key data in accordance with FIPS 140-2. This key management infrastructure supports the storage and maintenance of key data using the PKCS#11 standard and other standards. This key management infrastructure will also be used to manage encryption keys that will be collected from clients within the scope of this feature.

3. Configuration of Encrypted Fields

It is important to consider that the 'ecosystem' of Navaera Applications is often comprised of more than one actual application. Navaera solutions have been architected as 'loosely coupled' or 'microservice' components since their original releases and no single application has monolithic control of all data. This architecture creates the added challenge of not only requiring application data encryption to be managed within the scope of a single application, but also requires each application to synchronously have access to the encryption key, as well as a central directory of encrypted fields.

3.1 Password and Encryption Control Daemon

Navaera solutions currently use an existing *Password and Encryption Control Daemon ("PECD"),* which is a software application that was originally designed to manage automated administrator password updates in accordance with Navaera's Break Glass Infrastructure. The functionality of that application has been expanded to support the new functionality necessary for application data key management.

3.1.1 New Support Features in the PECD

The PECD application provides the following capabilities with respect to the application-based encryption functionality:

1. It shall be used to manage a central directory of encrypted fields across all configured applications;

2. It shall be used to check-out, and maintain a transient copy of the client-managed key for the purpose of enabling applications to encrypt and decrypt data using that key, in real-time.

3.1.2 Central Encryption Directory Configuration

Within the PECD, a configuration table, maintained in a defined database, shall be used to manage the Central Encryption Directory ("CED") and define fields within each application that will remain encrypted and will require real-time decryption for active use. In initial releases of this feature, the CED shall be a reportable table but is not planned to support client-side management. In other words, Navaera will enable clients to view the supporting data for configuration, however it is not yet planning to enable clients to be able to edit the configuration.

The CED table structure is defined using four fields, which are described in the table below:

Field	Field Description
Application Key	A key for each designated application configured in the instance. For example: NOD, IMF, FRA, DSB, etc.
Table	The table name, within the application database that contains the field for encryption

Field	The name of the field, within the application database and defined table that will be encrypted.
Encrypted	A binary indicator for whether the field will be encrypted. If the field is set to '1' it will be encrypted. Otherwise, it will not be encrypted.

The table shall be automatically populated, at application startup with a list of all fields that are present in each application database configured for the instance. By default, any field not specifically designated as encrypted, should not be encrypted. In other words, any new table/field combinations added to the table at application startup should be considered not encrypted.

4. Encryption Key Management

Support for client-management of the symmetric encryption key used to actually encrypt and decrypt data is provided through the self-service feature. A screenshot of the Encryption Keys management UI is shown below, in Figure 1.

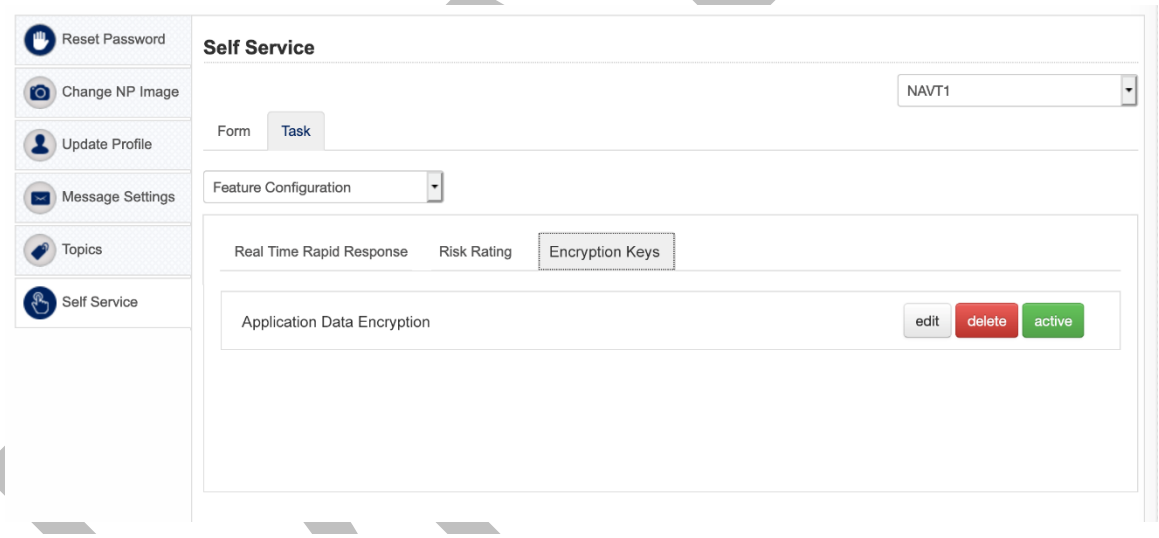

Figure 1 - Encryption Key Management Self-Service

When the user clicks 'edit,' a modal user interface appears that enables the user to enter an AES symmetric key. The key length must be a minimum of 256 bits.

4.1 Storage of Encryption Key

Upon submission of the key through self-service user interface, the key shall be submitted to the key management infrastructure for storage, and shall be associated with the client Radix. It should be noted, however that the key shall not be entirely rotated until the PECD application has been restarted. This is because in order to rotate the key, data already encrypted with an older key must first be decrypted with the old key, and then must be re-

encrypted with the new key. The PECD application is designed to orchestrate this rotation only on its startup. Key rotation considerations are discussed later in this document.

5. Encryption and Decryption Methods of Action

As discussed above, the PECD application polls the key management infrastructure on startup to determine if a new key has been provided. In the event that a new key has been updated, the system shall perform the following checks, in the following order:

1. Determine if existing data has been encrypted;
2. If the data is encrypted, then the key rotation process should be initiated; and
3. If the data is not encrypted, then the initial encryption process should be initiated.

5.1 Initial Encryption Process

If, on startup, PECD identifies the scenario where data is not already encrypted, the application shall ingest the CED, and begin the encryption process for all data enabled for encryption. Prior to persisting the newly encrypted data, the data shall be written to a temporal batch file, which shall then be persisted to the required table using BCP only after all required data for encryption has been encrypted.

5.2 Key Rotation Process

If, on startup, PECD identifies the scenario where data is already encrypted with a prior key, it should then asynchronously decrypt, and then encrypt the data in a two-threaded process. The first thread shall decrypt data into an in-memory circular buffer while the second thread shall consume data in the buffer, and encrypt it with a new key. Prior to persisting the newly encrypted data, data will be written to a temporal batch file, which will then be persisted to the required table using BCP only after all required data for encryption has been encrypted.

5.3 Real-Time Encryption/Decryption of Required Data

As mentioned earlier in this document, the component architecture of Navaera solutions, whether deployed in a SaaS or On-Premises format, will require data that is shared between components and applications decrypt data on an as-needed, real-time basis. Examples of this process are as follows:

1. In the event that the Navaera On-Demand Node application is configured to encrypt data in certain fields, it shall first check out the encryption key in-use for the Radix, and store the key in secure memory space. It shall then read the data from the input RM-XML, and then encrypt the data prior to persistence using the current key; and

2. In the event that ADX needs to read encrypted data, it shall first check out the encryption key in-use for the Radix, and store the key in secure memory space.

During a data source filter task, this key shall be used to decrypt data prior to inserting it into in-memory processing storage;

3. In the event that EnView IMF needs to decrypt data for display purposes, it shall first check out the encryption key in-use for the Radix, and store the key in secure memory space. When required for display purposes, it shall then decrypt the data prior to enabling the required data to be displayed.

Each application shall coordinate encryption or decryption needs by validating whether fields contain encrypted data or require data to be encrypted by subscribing to the CED data published by PECD.

In other words, as a general framework, PECD shall serve to manage key rotation and initialization, and also:

1. Serve subscribing applications by publishing the CED; and
2. Act as an agent to enable the symmetric key to be checked out by subscribing applications.