

---

**Raven**

**GTRI**

**Nov 13, 2023**



# CONTENTS

<b>1</b>	<b>What is Raven?</b>	<b>3</b>
1.1	End User Manual . . . . .	3
1.2	Technical Manual . . . . .	13
1.3	About Raven Team . . . . .	29



This document exists as a guide to help users understand the Raven mortality platform, providing both end-user manuals as well as technical documentation.



## WHAT IS RAVEN?

Raven is an open-source and proof of concept platform for the Medicolegal Death Investigation (MDI) FHIR Implementation Guide (IG). It is both a

1. **Testing Tool** for data validation, data comparison, data exchange via API, and any features on demand in the future, and
2. **Education Tool** for showing various examples in FHIR format and workflows in action”

The current Raven tooling and tests aid developers in implementing the MDI FHIR record format by validating the data against MDI IG guidelines and FHIR-based extended API operations. For more information on each Raven tool components, Raven’s MDI CSV schema, or MDI FHIR standard please see the corresponding sections in this document, accessible through the table of contents below or the sidebar navigation menu.

### 1.1 End User Manual

In this section, you can learn how to use and *run* the Raven platform. You can think of Raven as a playground or sandbox. Each structure has its own function and feature that a user can *play* with or even put your hands in dirt if the user wants. In Raven, you can *run* and *play*.

*Are you non-technical?* If you are a non-technical end user, then we recommend you to go to “[The MDI Standard](#)” page first and learn about Medicolegal Death Investigation (MDI) workflows

#### 1.1.1 The MDI Standard

##### What is FHIR?

[HL7 Fast Healthcare Interoperability Resources \(FHIR\)](#) is a successor to HL7’s earlier industry standards healthcare messaging, HL7v2.x and HL7v3.x. It builds upon those standards to produce a modern interoperability standard, enabling the easy exchange of healthcare records across systems.

FHIR is built around the concept of “Resources”, logically distinct entities that serve as the minimum granularity for transfer. For example, the Patient resource represents core patient demographic data and serves a focal reference for many other resources. Other resources include clinical concepts such as Condition or an Observation.

FHIR is currently up to its R5 release, though R4 is still the most prevalent of the modern releases and continues to be the release in which most development is focused. For a complete list of FHIR R4 Resources and their respective maturities, please see the [FHIR R4 Resource List](#).

## **What is MDI FHIR IG?**

The [Medicolegal Death Investigation \(MDI\) FHIR Implementation Guide \(IG\)](#) is a FHIR implementation guide detailing the proper method of using FHIR resources to construct a FHIR version of a Death and Toxicology Reporting. The MDI standard is developed to support modernization of interoperability between Coroner/Medical Examiner case management systems (CMS) and other systems such as Electronic Death Registrar Systems (EDRS) and Toxicology Lab Information and Management System (LIMS).

The Raven Platform uses the MDI IG for handling death records, importing MDI data and exporting to FHIR resources. The Raven Platform allows users to import their own data into FHIR MDI resources and store them on the Raven FHIR Server.

MDI IG is still in the draft version and being evolved as more data elements are considered. The MDI IG will follow the HL7 FHIR IG development cycles and will become mature over the development cycles.

For a more detailed breakdown of MDI contents, please see the [official MDI Implementation Guide](#).

## **Overview of MDI Workflows**

Currently, two workflows are defined in the MDI IG, MDI-to-EDRS and Toxicology-to-MDI. The MDI IG defines profiles to describe the required content structures for the workflows.

### **MDI-to-EDRS**

MDI-to-EDRS workflow represents the interoperability between MDI case management system (CMS) and state's electronic death registration system (EDRS). In MDI IG, this workflow is supported by MDI-to-EDRS profiles. As it happens in most states, the case is mostly created by funeral directors. Thus, this workflow begins with an initial case created at the EDRS. CMS first searches EDRS for a case and retrieves the case with limited decedent's demographics. CMS may update the case during the journey of the death investigation. When the investigation is completed, the case shall be certified and submitted to EDRS.

In this workflow, users can validate the MDI-to-EDRS FHIR bundle documents, load the documents, and submit to EDRS. It's highly recommended for users to first validate the FHIR data before loading to Raven. For those who do not have their own dataset or are not ready to produce the dataset, Raven allows users to search the Raven FHIR Server, load the case and play with the case. Users can explore the raw FHIR data along with the rendered data in forms.

### **Toxicology-to-MDI**

Toxicology-to-MDI workflow represents the interoperability between forensic toxicology laboratory information management system (LIMS) to an MDI case management system (CMS). In MDI IG, this workflow is supported by Toxicology-to-MDI profiles. This workflow is bidirectional. There is an initial lab order sent from CMS with samples. After lab work is performed, the lab report is sent back to CMS from LIMS. Currently, the MDI IG specifies the lab reporting direction only and uses FHIR messaging for the data exchanges.

Users can validate the Toxicology-to-MDI FHIR bundle messages and store the messages in Raven FHIR server.



## 1.1.2 Record Management

### Record Import (Importing MDI Records)

Record Importing is a Raven feature that imports the Comma-separated Values (CSV) or spreadsheet file into the MDI FHIR server in an MDI FHIR IG compliant format.

The FHIR data model is complicated and structured with multi-levels and logical references. In order to help transitioning from non-FHIR data to MDI IG compliant format, the MDI CSV format was designed. The Case Importing feature maps the pre-defined MDI CSV format to the MDI FHIR IG format and persists them in the MDI FHIR server.

---

**Note:** **Use Case:** Mapping of any case management system data (in CSV/spreadsheet format) to MDI FHIR and importing them to the Raven FHIR server

---

For the case importing, a predefined XLSX or spreadsheet template is provided to users. Users populate their data to the provided template. The user-data will be converted to the MDI FHIR IG data and imported to Raven FHIR server.

For connectathon support, the Case Importing feature in Raven will generate reference MDI FHIR IG data using connectathon testcase data so that participant-generated MDI FHIR IG data can be compared with the reference MDI FHIR IG data with the comparison tool. The case importing and comparing data are done as follows.

#### Importing Procedure

- Reads the testcases spreadsheet
- Converts the data in the testcases to MDI CSV
- Mapper maps the MDI CSV to MDI IG FHIR and stores the converted MDI FHIR IG data in the Raven FHIR Server to be used as reference data
- When participants' validated data are loaded to Raven, Comparison Tool compares the loaded data with the reference data. See "[Validation And Comparison](#)" page for more information.

#### Spreadsheet Schema

If the user cannot construct the FHIR records necessary, or are unfamiliar with the FHIR standard in general; Raven provides an excel spreadsheet XLSX template for easy of use. Users can fill in individual case data as plaintext values, and use the import case view on the RAVEN platform. RAVEN will transform the XLSX data into individual FHIR case records that adhere to the FHIR-MDI-IG standard. The template is hosted on the RAVEN base site; and a [copy](#) can be directly downloaded from the public internet.

Sections	Elements	Description	Case 1	Case 2	Case 3	Case 4	Case 5
Tracking Numbers		This Excel File can be imported into RAVEN2.0, converting into FHR and importing into RAVEN.					
		This section is for tracking separate identifiers: an identifier for the local mdi system, and the state registrar(EDRS) file number.					
	Tracking Number: Mdi Case Number	A locally unique case number from the case management system. Optional					
	Tracking Number: EDRS File Number	A locally unique case number from the state registrar(EDRS). Usually assigned once the case has been registered or submitted to the state registrar. Optional					
Decedent		This section is for decedent demographic information, all information resides in the us-core-patient resource within the document					
	Decedent Name	Primary name of the Decedent. Accepted Formats: <First Name> <Last Name> <First Name> <Middle Initial> <Last Name> <First Name> <Middle Name> <Last Name>					
	Decedent Race	Race of Decedent Accepted values are shown in the dropdown If the decedent has multiple race entries, use a comma-separated format to enter multiple races					
	Decedent Ethnicity	Ethnicity of Decedent Accepted values are shown in the dropdown					
	Decedent SexAtDeath	Decedent's gender, as determined at the time of death. Accepted values are shown in the dropdown					
	Decedent SSN	Social Security Number of decedent Accepted Formats: ###-##-#### #####					
	Decedent Age	The age of the decedent. In case of infant or fetal death, an age denomination may be used. Accepted formats: <Age Value> <Age Value> <Age Unit>					
	Decedent DOB	The date in which the Decedent was born Accepted Formats: mm/dd/yyyy mm-25-yyyy					
	Decedent Marital status						

## RAVEN Import XLSX Spreadsheet Definitions

Sections	Elements	Description
Tracking Numbers		This section is for tracking
Tracking Numbers	Tracking Number: Mdi Case Number	A locally unique case num
Tracking Numbers	Tracking Number: EDRS File Number	A locally unique case num
Decedent		This section is for decede
Decedent	Decedent Name	Primary name of the Deco
Decedent	Decedent Race	Race of DecedentAccepte
Decedent	Decedent Ethnicity	Ethnicity of DecedentAcco
Decedent	Decedent SexAtDeath	Decedent's gender, as det
Decedent	Decedent SSN	Social Security Number o
Decedent	Decedent Age	The age of the decedent.
Decedent	Decedent DOB	The date in which the Dec
Decedent	Decedent Marital status	
Decedent	Decedent Residence: Street	Primary Address of Dece
Decedent	Decedent Residence: city	
Decedent	Decedent Residence: county	
Decedent	Decedent Residence: State, U.S. Territory or Canadian Province	
Decedent	Decedent Residence: Postal Code	
Decedent	Decedent Residence: Country	
Cause And Manner of Death		This section is for the info
Cause And Manner of Death	Cause of Death Part I Line a	First line of the cause of o
Cause And Manner of Death	Cause of Death Part I Line b	Second line of the cause o
Cause And Manner of Death	Cause of Death Part I Line c	Third line of the cause of
Cause And Manner of Death	Cause of Death Part I Line d	Fourth line of the cause o
Cause And Manner of Death	Cause of Death Part I Interval, Line a	Approximate interval of t
Cause And Manner of Death	Cause of Death Part I Interval, Line b	Approximate interval of t
Cause And Manner of Death	Cause of Death Part I Interval, Line c	Approximate interval of t
Cause And Manner of Death	Cause of Death Part I Interval, Line d	Approximate interval of t
Cause And Manner of Death	Cause of Death Part II	Other contributing conditi
Cause And Manner of Death	Manner of Death	Manner of deathAccepted
Cause And Manner of Death	Date of Injury	If an injury occurred lead
Cause And Manner of Death	Time of Injury	If an injury occurred lead
Cause And Manner of Death	How Injury Occurred	A text description of the i
Cause And Manner of Death	Did Injury Occur at Work?	In the case of an injury, w

Sections	Elements	Description
Cause And Manner of Death	Decedent's Transportation Role During Injury	If an injury occurred with
Death Circumstances		This section describes sp
Death Circumstances	Location of death	Full or partial address des
Death Circumstances	Location of Injury	If an injury occurred, des
Death Circumstances	Pregnancy status	Was the decedent pregena
Death Circumstances	Did Tobacco Use Contribute to Death?	If the decedent used tobac
Jurisdiction		This section describes jur
Jurisdiction	Decedent Date of death	The date of death of the d
Jurisdiction	Decedent Time of death	The time of death of the d
Jurisdiction	Date establishment method	The circumstances of how
Jurisdiction	Date pronounced dead	The date in which the dec
Jurisdiction	Time pronounced dead	The time in which the dec
Jurisdiction	Place of death	The type of place the dec
Exam-Autopsy		This section describes the
Exam-Autopsy	Autopsy Performed?	Was an autopsy performe
Exam-Autopsy	Autopsy Results Available?	If an autopsy was perform
Chief Medical Examiner/Coroner		This section describes the
Chief Medical Examiner/Coroner	Medical Examiner Name	Name of the Medical Exa
Chief Medical Examiner/Coroner	Medical Examiner Phone Number	Phone number of the offic
Chief Medical Examiner/Coroner	Medical Examiner License Number	Medical Examiner Licens
Chief Medical Examiner/Coroner	Medical Examiner Office: Street	Primary Address of the m
Chief Medical Examiner/Coroner	Medical Examiner Office: City	
Chief Medical Examiner/Coroner	Medical Examiner Office: County	
Chief Medical Examiner/Coroner	Medical Examiner Office: State, U.S. Territory or Canadian Province	
Chief Medical Examiner/Coroner	Medical Examiner Office: Postal Code	
Certifier		This section describes the
Certifier	Certifier Name	Name of the Certifier.Acc
Certifier	Certifier Type	Is the Certifer a Physician

## Record Viewer (Viewing Cases)

The Record Viewer is a UI component which allows the browsing and viewing of Raven FHIR Server records, encompassing both MDI Case Documents (MDI to EDRS) and Toxicology Reports (LIMS to MDI). In addition to providing a user-friendly option for viewing the data present on the FHIR Server, the layout is structured from the perspective of the MDI Implementation Guide to serve as an educational tool to better understand the data structure and fields which make up the MDI to EDRS and Toxicology to MDI documents.

The screenshot displays the Raven Record Viewer interface. At the top, there's a header bar with fields for Subject Name (Rebecca Ellison), Date/Time of Death (2021-12-31), Gender (Female), and a placeholder for the MDC Case Number. Below this, there are buttons for 'Expand All Sections' and 'Collapse All Sections'. The main content area is divided into three expandable sections: Demographics, Occupation History, and Circumstances. The Demographics section is currently expanded, showing fields for Social Security Number, Gender, Date of Birth, Marital Status, Race, and Ethnicity. The Occupation History and Circumstances sections are collapsed. To the right of the main content area, there's a pane titled 'Structure' showing the JSON structure of the selected field. The JSON structure is a complex object with various nested arrays and objects, including fields like 'resourceType', 'id', 'profile', 'type', 'extension', and 'system'.

The Record Viewer also features a FHIR Resource Explorer, which allows users to select a field and see the underlying FHIR Resource structure containing the related data. The FHIR Resource Explorer will support JSON and XML formats, as well as a human readable “narrative view”.

**Note: Use Case:** Human readable display of MDI FHIR IG data with a FHIR explorer. Any cases loaded in the Raven FHIR server should be retrievable by Record Viewer. Users can use FHIR APIs to load the data.

### 1.1.3 Validation And Comparison

The Validation & ComparisonRaven feature set includes the MDI Validator and the Comparison Tool. The purpose of Validation & Comparison is to confirm MDI record conformance and validity as it is important to connectathon testing and support.

The MDI (Medicolegal Death Investigation) Validator is a web application that allows users to upload or copy-paste their MDI FHIR IG data for validation. The MDI Validator uses the HL7 FHIR validator as a core validation engine and provides a user interface (UI) wrapper that is tailored to the MDI IG.

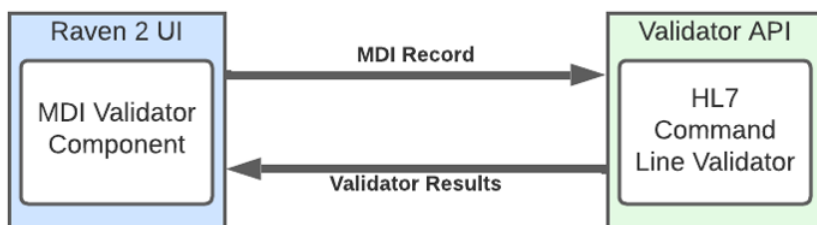
The Comparison Tool is a connectathon supporting tool that will compare pre-validated test case MDI FHIR IG data with the user generated FHIR data. Users will want to ensure that not only their data validated but also their contents in FHIR correctly populated. The Comparison Tool will provide a compressed case view with side by-side comparison of the imported record and the correct test case record. This will let users easily hone in on individual content issues and have confidence in their process.

**Note:** Connectathon Support - validation of user generated MDI FHIR IG data.

Connectathon participants can enter their FHIR documents into the MDI FHIR IG validator and review any errors. The validator confirms the users’ confidence in their external mapping as well as provides a learning experience for review and conforming to the IG.

## Architecture

These would be modules within the Raven Platform or could be used independent of Raven for testing. They rely on the Raven FHIR server to serve the data.



### 1.1.4 Workflow Simulator

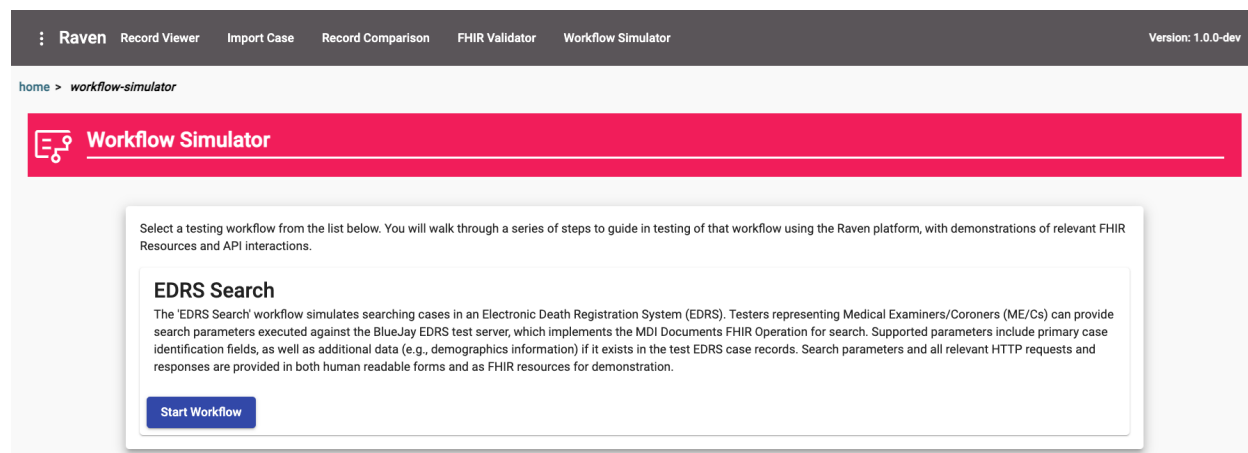
The Workflow Simulator is a module of the Raven platform that allows users to simulate data flows between Medicolegal Death Investigation (MDI) systems such as case management systems (CMS), electronic death registration systems (EDRS), and toxicology laboratory information systems (LIMS). The supported data flows are defined by interoperability use cases. There are two types of established use cases: Testing Use Cases and Operational Use Cases.

1. Testing Use Cases: Use cases that are developed for testing events to evaluate the interoperability implementation of MDI systems
2. Operational Use Cases: Use cases that are defined by users in the MDI community to standardize the operations such as search, update, certification, amendment, or messaging.

Some testing use cases can be supported by individual Raven modules such as the FHIR Validator and Record Comparison modules. Operational use cases, which are often more complex, such as the Search EDRS API workflow, can be implemented as a proof-of-concept using the Workflow Simulator prior to production development. Thus, users can use the Workflow Simulator as clear indicators and metrics to help making decisions on where to spend their resources as they rebuild for modernization and interoperability within their data ecosystems.

## Workflows/Use Cases

When opening the Workflow Simulator module, the user will be given a list of currently implemented workflows to select from. Once selected, the workflow will be loaded, presented as a step by step process in Raven.



## Search EDRS (CMS to EDRS)

### Step 1 - Import/Select Record (Optional)

The first step of the Search EDRS workflow allows the user to select an MDI to EDRS Document from the Raven FHIR Server to use to auto populate search parameters with the values from the record. Users may also import a record into the workflow as an MDI to EDRS Document Bundle in the FHIR JSON format. This step is entirely optional, and if a user wishes to proceed without a case select they can manually input all search parameters required.

The screenshot displays the 'Workflow Simulator - Search EDRS' interface. The top navigation bar includes 'Raven', 'Record Viewer', 'Import Case', 'Record Comparison', 'FHIR Validator', and 'Workflow Simulator'. The version is '1.0.0-dev'. The breadcrumb trail is 'home > workflow-simulator > search-eds'. The main header is 'Workflow Simulator - Search EDRS'.

The interface shows a progress bar with three steps: 1. Select MDI to EDRS document, 2. Configure Endpoint, and 3. Search EDRS. Step 1 is currently active.

Below the progress bar, a message states: 'Step 1 - Select or Import an MDI to EDRS Document (Optional) If provided, the Document Bundle is used to populate the Search Parameters. Otherwise, the Search Parameters may be manually specified.'

The main content area has two tabs: 'Select MDI to EDRS Document' (active) and 'Import MDI to EDRS FHIR Document Bundle'. Under the active tab, there is a 'Filter' input field, a 'Manner of Death' dropdown menu, and a 'Clear Filters' button.

A table lists search results with columns: #, Name, Gender, Date of Death, Manner of Death, and MDI Case Number.

#	Name	Gender	Date of Death	Manner of Death	MDI Case Number
1	Tacktheritrix, Jackmerius	Male	12/13/2022	Natural	000005
2	Shower-handel, Davoin	Male	12/13/2022	Accidental Death	000006
3	Buckshank, Ozmatz	Male	12/13/2022	Natural	000007
4	Washingbeard, Beezer	Male	12/13/2022	Accidental Death	000008
5	Winslow, Annie	Female	12/13/2022	Accidental Death	000006
6	Stevens, Erica	Female	12/13/2022	Natural	000007
7	Rivera, Chance	Male	12/13/2022	Accidental Death	000008
8	Thacker, Erica	Female	12/13/2022	Natural	000007

The second screenshot shows the 'Import MDI to EDRS FHIR Document Bundle' tab. It contains instructions: 'The data provided should be a valid FHIR MDI to EDRS Document Bundle. Please validate your resource prior to using it in the workflow simulator or unexpected behavior may occur.' and 'Click the "Input MDI to EDRS Document Bundle" button below to paste in your complete Document Bundle resource or click "Select File" to select a plain text (JSON) file from your local drive.'

There are two buttons: 'Input MDI to EDRS Document Bundle' and 'Select File'.

Below these buttons, there is a section titled 'Selected MDI to EDRS Case' with three input fields: 'Decedent Name:', 'Date/Time of death:', and 'MDI Tracking Number:'. A 'Proceed to Configure Endpoint' button is at the bottom right.

### Step 2 - Configure Endpoint

After the user decides on whether they would like to use an existing record, they are taken to the Configure Endpoint step. This part of workflow is the configuration of the FHIR endpoint for testing the search functionality against an Electronic

Death Registration System (EDRS). Users may select between a pre-registered Endpoint or a Custom Endpoint. Pre-registered endpoints are configured in Raven and will typically provide open testing endpoints, including the Raven BlueJay server which acts as a test EDRS. Selecting a pre-registered endpoint requires no additional configuration from the user. For custom endpoints, users may provide a non-registered testing endpoint and setup basic authorization as needed. Custom endpoints are not recorded in any form by the Raven platform, and their use is entirely the responsibility of the user. Please note that the Raven platform is a single page application based in a web browser, and using custom endpoints may result in the user's browser recording sensitive information separate from the Raven platform. (This should be managed by the user in coordination with their organization's internal IT policies.)

home > workflow-simulator > search-edrs

**Workflow Simulator - Search EDRS**

1 Select MDI to EDRS document 2 **Configure Endpoint** 3 Search EDRS

Step 2 - Configure the EDRS endpoint. The BlueJay test server is provided as a default.

**Selected MDI to EDRS Case**

Decedent Name:

Date/Time of death:

MDI Tracking Number:

Please be aware that for custom endpoint configuration users are responsible for all security considerations. Raven does not store any user information, though some web browsers may attempt to store sensitive data input into form fields.

☒ Registered Endpoint ☐ Custom Endpoint

Select Endpoint  
BlueJay

[View Server Capability Statement](#)

[View Server \\$mdi-documents Operation Definition](#)

[Back](#) [Proceed to search](#)

### Step 3 - Search EDRS (API Interaction)

The final step of the Search EDRS workflow is the execution of search parameters against the identified EDRS endpoint. The potential parameters fields are data driven and populated automatically based on the FHIR MDI Implementation Guide “MDI Documents” Operation Definition. Users may select any number of parameter fields they wish to use. If a record was selected or imported during step 1, the parameters will attempt to have their values automatically populated. As the user enters data or modifies the parameter fields, an example of the FHIR Parameters resource is shown for demonstration purposes which matches the current state of the parameters HTML form. This allows users building reference implementations a model to which they can refer in their own development, tying a standard HTML style form to the underlying FHIR resource it will produce. Once satisfied with their search parameters, users may connect to the EDRS and attempt to find matching records.

Raven
Record Viewer
Import Case
Record Comparison
FHIR Validator
Workflow Simulator
Version: 1.0.0-dev

home > workflow-simulator > search-edrs

Workflow Simulator - Search EDRS

1 Select MDI to EDRS document
2 Configure Endpoint
3 Search EDRS

Step 3 - Set EDRS search parameters. Parameters may be automatically populated if a case was selected in Step #1. The parameter list is variable, defined by the FHIR Server's operation definition for EDRS searching. Search results are shown below. If any results were found, users may select a case and load a summary of the data.

Selected MDI to EDRS Case
Endpoint Configuration

EDRS Search Parameters

Parameter Given Name

×

Parameter Family Name

×

Parameter EDRS File Number

×

+ Add Parameter

Clear Search

Search

Results

EDRS Results

EDRS Results

EDRS Results

If records are identified on the EDRS, the results are shown below the parameters. The results can be viewed either as a human readable table summarizing the matching records, or as a raw FHIR search set bundle. In addition, users can use the HTTP Request and Response tabs to better be able to identify the headers involved in the HTTP call to the EDRS. In the summary table under the default Results tab, a record may be selected to load further information.

Result	FHIR Result	HTTP Request	HTTP Response								
<table> <thead> <tr> <th>Name</th><th>Gender</th><th>Address</th><th>EDRS File Number</th></tr> </thead> <tbody> <tr> <td>Organa, Leia</td><td>Female</td><td>3432 Earth Street Alpharetta GA 12321</td><td>EDRS-04</td></tr> </tbody> </table>				Name	Gender	Address	EDRS File Number	Organa, Leia	Female	3432 Earth Street Alpharetta GA 12321	EDRS-04
Name	Gender	Address	EDRS File Number								
Organa, Leia	Female	3432 Earth Street Alpharetta GA 12321	EDRS-04								

Once selected in the Results table, the record is displayed below the table. As with the full search results, this can be viewed either as a human readable summary and as the underlying FHIR MDI to EDRS Document Bundle



Result	FHIR Result	HTTP Request	HTTP Response								
<table> <tr> <th>Name</th><th>Gender</th><th>Address</th><th>EDRS File Number</th></tr> <tr> <td>Organa, Leia</td><td>Female</td><td>3432 Earth Street Alpharetta GA 12321</td><td>EDRS-04</td></tr> </table>				Name	Gender	Address	EDRS File Number	Organa, Leia	Female	3432 Earth Street Alpharetta GA 12321	EDRS-04
Name	Gender	Address	EDRS File Number								
Organa, Leia	Female	3432 Earth Street Alpharetta GA 12321	EDRS-04								

## 1.2 Technical Manual

In the technical manual, we get into deeper and provide technical information for developers or users with a technical background. The contents of this section will be subject to change as the Raven evolves or is added with new features. As the changes can happen often, it's highly recommended to refresh each page so that the cached pages can be refreshed with new updates.

### 1.2.1 Standard MDI API (MAPI)

**Note:** Standard MDI API (MAPI) will be documented as a best practice in the MDI IG site in the future. Until then, the Raven documentation will temporarily house the standard MAPI specification.

#### Operation APIs for MDI-to-EDRS Workflow

MDI FHIR Implementation Guide (IG) is available in <http://hl7.org/fhir/us/mdi/> This IG should be used for the payload of MAPI.

FHIR defines base restful APIs for FHIR data transportation. Their documents are available from <https://hl7.org/FHIR/http.html>. And, the FHIR API Operations are documented in <https://hl7.org/FHIR/operationslist.html>. MAPI is extended the FHIR ASPI operations. Therefore, the basic rules of FHIR APIs and operations are also applied to MAPI. For example,

- Content-type for FHIR resources is application/fhir+xml or application/fhir+json. This needs to be specified in the HTTP header.
- application/x-www-form-urlencoded can be used for POST search requests if HTTP Form is used.

In FHIR, FHIR resources, interactions, and operations are published using CompatibilityStatement (GET [base]/metadata). Detailed information about the CompatibilityStatement is available in <https://hl7.org/FHIR/capabilitystatement.html>. It is recommended that EDRS FHIR servers publish their capability statement as defined in this link.

## Security Recommendations

This section covers a minimum level of security recommended by the MDI FHIR IG. There are more data exchange protocols and content models defined in the [FHIR Security document](#). MDI systems that require a higher level of security should refer to the FHIR Security document for the interoperability.

## Secure Data Transportation

In most modern systems, digital data are exchanged using web services. FHIR recommends a web service called RESTful Application Programming Interface (REST API) where REST stands for **RE**presentational **St**ate **T**ransfer. REST API uses Transport Layer Security (TLS) for the secure transportation. More accurately, TLS 1.2 or higher needs to be used. This is also known as HTTPS. All data exchanges in MDI FHIR IG must be done in HTTPS

## Standard Authorization Protocol

A standard authorization protocol that can be used for the data access is the OAuth 2.0 (OAuth2) Authorization Framework defined in [RFC 6749](#). There are many documents provided by OAuth2 service providers that are much easier and simpler to understand. Searching on Internet using “OAuth2” keyword will return several related documents.

## Roles in OAuth2

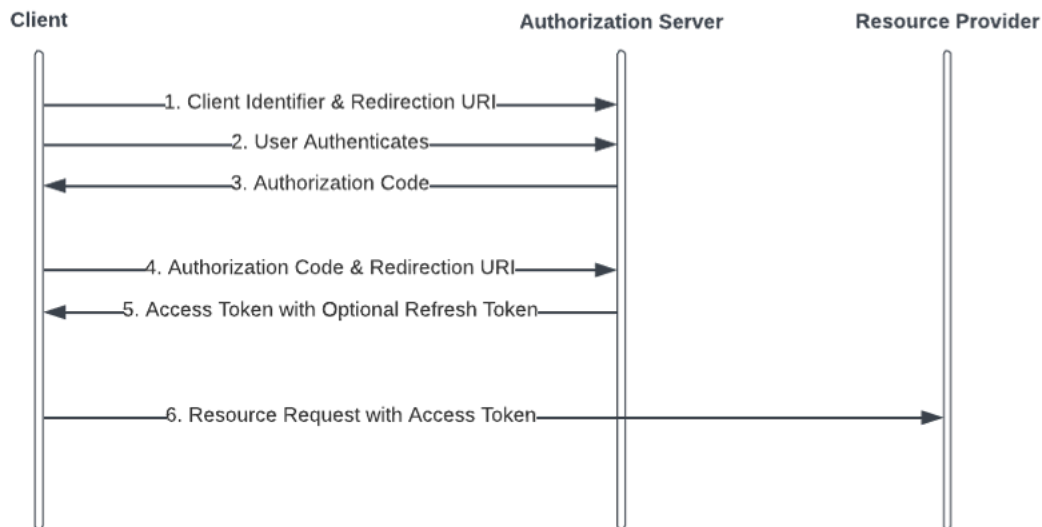
OAuth2 defines several components that play different roles. Systems in MDI IG should play the roles to support the OAuth2. The OAuth2 roles are changed depending on the roles in the MDI workflows. Table1 shows which OAuth2 roles the systems in MDI IG should play in the MDI-to-EDRS and Toxicology-to-MDI workflows. As more workflows are added to the MDI IG, additional roles may be added to the system, which may be ended up playing multiple roles.

Role	Responsibility	MDI- and- EDRS	Tox- and- MDI
Authoriza- tion Server	Server that authenticates the resource owner and issues access tokens to the client application. The authorization server can be the same as the authentication server or can be a separate server.	EDRS	CMS
Client	Application that wants to access the resource on behalf of the resource owner. The client can be a web application, a mobile application, or a desktop application.	CMS	LIMS
Resource Owner	User who owns the resource (such as a photo or a document) that a client application wants to access. The resource owner grants permission to the client application to access the resource.	CMS Users EDRS Users	LIMS Users
Resource Server (Provider)	Server that hosts the resource that the client application wants to access. The resource server verifies the access token and grants access to the resource if the token is valid.	EDRS	CMS

**Table1:** Roles in OAuth2 and MDI Systems

## OAuth2 Flows

OAuth2 defines different flows based on the client (or application) types. This document only discusses the flow(s) that might be applicable to the client types in MDI. Figure 1 depicts the authorization code flow that can provide authentication and authorization of clients in MDI workflows. Detail transactions for the authorization code flow are explained in section 4.1 of RFC 6749.



**Figure 1:** Authorization Code Flow in OAuth2

### Client Registration

For a client to be able to get authenticated and authorized, the client must be registered at the authorization server. When a client is registered, the client should provide *redirection\_uri*. *Client\_id* will then be issued to the client. The client will use the *client\_id* and *redirection\_uri* for its authentication and authorization.

### Authorization Request

Client first needs to get an authorization code. In Figure 1, 1, 2, and 3 are the authorization request steps. Client should provide client identifier with *client\_id* and *redirection\_uri* (optional). *Client\_id* and *redirection\_uri* will be matched with registered data at the authorization server (1). If the request is valid, then the client will be redirected to user authentication (2) where authentication and consent occur. Once client authenticated and authorized, authorization code is returned to client by being redirected to the *redirection\_uri* (3).

Parameters for the authorization request are as follows. They are included as URL parameters with HTTPS GET method. However, POST can also be used by having the parameters included in the payload with a content-type set to **application/x-www-form-urlencoded**.

#### Parameters

Request		
response_type	required	Fixed value: code
client_id	required	Client identifier issued at the registration
redirection_uri	optional	Full URL that authorization server will use to respond to request
scope	optional	
state	recommended	
Response		
code	required	Authorization Code to be used for the access token request
state	required	If client puts state in the request

Response to the request is sent to the *redirection\_uri* at the client using **application/x-www-form-urlencoded** content-type.

Example:

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=xyz&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
```

## Access Token Request

After authorization code is successfully received, access token request can be sent to authorization server (or token server) for an access token. Steps 4 and 5 in figure 1 are access token request flow. Parameters for the access token request are as follows.

### Parameters

Request		
grant_type	required	Fixed value: authorization_code
code	required	The authorization code received from the request.
redirection_uri	required	Full URL that authorization server will use to respond to request
client_id	required	If the client is not authenticating with authorization server
Response		
access_token	required	Access token issued by the authorization server
token_type	required	Type of the token issued
expires_in	recommended	The lifetime (in sec) of the access token
refresh_token	optional	Used to obtain a new access token
scope	optional	

Example

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=Sp1xl0BeZQQYbYS6WxSbIA&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
```

## Refresh Token Request

If refresh token is available, then a request can be sent to the authorization server (or token endpoint). If client authentication is included, the authentication needs to be performed.

### Parameters

Request		
grant_type	required	Fixed value: refresh_token
refresh_token	required	Refresh token issued to a client.
scope	optional	
Response		
access_token	required	Access token issued by the authorization server
token_type	required	Type of the token issued
expires_in	recommended	The lifetime (in sec) of the access token
refresh_token	optional	Used to obtain a new access token
scope	optional	

### Example

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token&refresh_token=tGzv3J0kF0XG5Qx2TlKWIA
```

## Accessing Resource Server

After authentication/authorization is (are) completed, client can put the access token in the header and submit the request to resource server for data. The access token is placed in the header as follows.

```
Authorization: Bearer <access token>
```

Client must check the *expires\_in* value. If token is expired, and refresh access token is supported, then client can submit the request to renew the access token (see sections above related to the requests).

## Error Handling

If error occurs during authorization, the server should respond as specified in 5.2 of [RFC 6749](#). In summary, the response should be 400 (Bad Request) status code (unless specified otherwise) with the following parameters.

### Error Parameters:

Key		
error	required	A single ASCII error code from the following values:
Values		
invalid_request	The request is missing a required parameter, includes an unsupported parameter value (other than grant type), repeats a parameter, includes multiple credentials, utilizes more than one mechanism for authenticating the client, or is otherwise malformed.	
invalid_client	Client authentication failed (e.g., unknown client, no client authentication included, or unsupported authentication method). The authorization server MAY return an HTTP 401 (Unauthorized) status code to indicate which HTTP authentication schemes are supported. If the client attempted to authenticate via the “Authorization” request header field, the authorization server MUST respond with an HTTP 401 (Unauthorized) status code and include the “WWW-Authenticate” response header field matching the authentication scheme used by the client.	
invalid_grant	The provided authorization grant (e.g., authorization code, resource owner credentials) or refresh token is invalid, expired, revoked, does not match the redirection URI used in the authorization request, or was issued to another client.	
unauthorized_client	The authenticated client is not authorized to use this authorization grant type.	
unsupported_grant_type	The authorization grant type is not supported by the authorization server.	
invalid_scope	The requested scope is invalid, unknown, malformed, or exceeds the scope granted by the resource owner.	
Values for the “error” parameter MUST NOT include characters outside the set %x20-21 / %x23-5B / %x5D-7E.		
Key		
error_description	optional	Human-readable ASCII text providing additional information, used to assist the client developer in understanding the error that occurred. Values for the “error_description” parameter MUST NOT include characters outside the set %x20-21 / %x23-5B / %x5D-7E.

18

Chapter 1. What is Raven?

## Example

```

HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "error": "invalid_request"
}

```

## Search API



The above diagram depicts the MDI to EDRS API workflow. MAPI design follows this workflow. We will start with the SEARCH operation. In most states, the case is created by funeral directors. For this document, we assume that the case has already been created at the EDRS with decedent's demographics.

The FHIR defines basic search API. However, the FHIR search parameters are specific to a resource. The extended search queries are complicated. So, MAPI extended the FHIR document generation operation (\$document) and defined search parameters that represent MDI data elements. Details about the base \$document operation is described in <https://www.hl7.org/fhir/composition-operation-document.html>

Let's first review how MAPI extended the \$document operation.

## Extended Operation for MDI-to-EDRS Document generation

This is a resource instance type extended operation. It means that the MDI document is generated from the Composition resource. And the extension is made to the extended search parameters.

This is an idempotent operation. Both POST and GET can be used with the following endpoint URL pattern.

```

POST [base FHIR Url]/Composition/$document with Parameters resource in the payload
GET  [base FHIR Url]/Composition/$document?name1=value1&name2=value2

```

## Search Parameters for the MDI Document Generation

Name	Cardinality	Type	Documentation
In Parameters			
id	0..1	uri	Composition.id of Composition - MDI to EDRS
tracking-number	0..1	token	Composition.extension:extension-tracking-number of Composition - MDI and EDRS
patient	0..*		One or more decedent related search parameters
patient.birthdate	0..1	date*	Decedent's date of birth
patient.family	0..1	string	Decedent's last name
patient.given	0..1	string	Decedent's first name
patient.gender	0..1	token	Decedent's gender
death-location	0..1	string	Location address in Location-death
death-date-pronounced	0..1	date*	Observation.component:datetimePronouncedDead in Observation - Death Date (either time or date-Time)
death-date	0..1	date*	Value[x] (actual or presumed date of death) in Observation - Death Date (either dateTime or Period)
Out Parameters			
return	0..1	resource	Bundle - Searchset or Bundle - Document MDI and EDRS. If [id] is supplied, then this should be Bundle - Document MDI and EDRS

\* [date parameter search in FHIR](#) uses first two characters for date range search (eg. “lt” for less than). To use the date range search, the type needs to be string.

Please note that the Search parameters related to patient are formatted with “.” (dot). In FHIR, this means that the search parameters after “.” are *part* of patient parameter in Parameters resource. See the example below.

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "patient",
      "part": [
        {
          "name": "family",
          "valueString": "Hans"
        }
      ]
    }
  ]
}
```

(continues on next page)



(continued from previous page)

```

    },
    {
      "name": "given",
      "valueString": "Kennoby"
    }
  ]
}

```

If *id* is provided within URL path (e.g., /Composition/*id*/\$document), then the output response should be an MDI document bundle as there will be only one or zero result.

If *id* or *search parameters* is provided in the URL parameter (e.g. [base]/Composition?name=value) or Parameters resource in the payload, then the output response should be a *searchset* Bundle resource with matching MDI document Bundle resources even if there is only one result. If “OR” search parameter is needed in the searching parameters, then as specified in the FHIR specification (<https://hl7.org/fhir/R4/search.html#escaping>), “,” should be used. For example, if we want to search records that has death-location equals to either a, b, or c, then its search parameter in Parameters resource will be like below.

```

"name": "death-location",
"valueString": "a,b,c"

```

Please see the examples of search Parameters resource and its response.

### Request

Listing 1: POST [FHIRbaseURL]/Composition/\$document

```

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "patient",
      "part": [
        {
          "name": "family",
          "valueString": "Hans"
        },
        {
          "name": "given",
          "valueString": "Kennoby"
        }
      ]
    }
  ]
}

```

### Response

```

{
  "resourceType": "Bundle",
  "id": "13ab1ecf-38ce-4f47-aebb-a38396a80775",

```

(continues on next page)

(continued from previous page)

```

    "type": "searchset",
    "total": 1,
    "entry": [
      {
        "resourceType": "Bundle",
        "id": "fd240814-5911-49bb-bb20-72066add4a18",
        "meta": {
          "profile": [
            "http://hl7.org/fhir/us/mdi/StructureDefinition/Bundle-document-mdi-to-
↪ edrs"
          ]
        },
        "type": "document",
        "entry": [
          {
            "fullUrl": "Composition/965a0688-e6f4-4bff-a96d-639cbd7ea295",
            "resource": {
              "resourceType": "Composition",
              "id": "965a0688-e6f4-4bff-a96d-639cbd7ea295"
            }
          }
        ]
      }
    ]
  }
}

```

## Error Handling

**API Level Errors** API itself can indicate errors. API errors are displayed in the HTTP code. 2xx are returned when API transactions are successfully processed. 4xx or 5xx are error codes. 3xx are not errors. These codes need to be supported at the client side if redirections are required by the server. More details can be found from [https://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_status\\_codes](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes).

CMS must check if the correct endpoint and search parameters are used if such errors are returned. Server also returns error code when there are document level errors. In this case *OperationOutcome* could be included in the payload. CMS would want to parse the payload as it contains the source of errors. For more information about the *OperationOutcome*, see the following section.

**MDI Document Level Errors with 2xx HTTP response** For all non 2xx status code, error(s) must be indicated in the response with a *OperationOutcome* resource.

In *OperationOutcome*, EDRS must include information what caused the error if the error needs to be fixed by CMS. If it's the EDRS that needs to fix the error, it must be indicated so that CMS user(s) can contact EDRS for the error. Below shows an example of *OperationOutcome*.

Listing 2: HTTP/1.1 500 Internal Server Error

```

{
  "resourceType": "OperationOutcome",
  "id": "searchfail",
  "text": {
    "status": "generated",

```

(continues on next page)

(continued from previous page)

```

    "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">\n
      <p>The "name" parameter has the modifier "exact" which is
↪not supported by
      this server</p>\n</div>"
  },
  "issue": [
    {
      "severity": "fatal",
      "code": "code-invalid",
      "details": {
        "text": "The \"name\" parameter has the modifier \"exact\" which is not
↪supported by this server"
      }
    }
  ]
}

```

## Read API

READ API uses the base FHIR operation \$document. The URL pattern is.

```
GET [base FHIR URL]/Composition/id/$document
```

id is a Composition resource Id, which is assigned by systems such as CMS and EDRS. If a server maintains the id for all generated FHIR Document Bundles, then this id should be used to get the document. The response for this API is a MDI document Bundle (not a *searchset* Bundle).

## Update API

During the death investigation, C/ME may need to update the case in the EDRS. This API allows CMS to update the active case. PUT should be used for the HTTP action method. And, Parameters resource is used to include the MDI document or profile(s) that C/MEs want to update. Since this API presumes that the case already exists in the EDRS, the case management system must either make sure identifier(s) is included in the MDI document or provide a parameter that EDRS can use to find the case to update.

FHIR endpoint for UPDATE API operations is as follow.

```
PUT [base url]/Composition/$update-mdi
```

The payload is Parameters resource as defined below.

Input/Output Parameters

Name	Cardinality	Type	Documentation
In Parameters			
Jurisdiction defined parameters	0..*	string	Any required parameters for a jurisdiction
tracking-number	1..1	token	EDRS case number if available
mdi-document	1..1	Bundle	MDI document bundle. The “mdi-document” is a reserved keyword. This should only be used for the MDI-and-EDRS profile bundle document.
warning	1..1	OperationOutcome	Informational OperationOutcome (For response ONLY)
Out Parameters			
return	0..1	OperationOutcome	If an error occurs, OO resource is returned. If response data need to be sent back, Parameters resource can be used.

Ex. **Request** in the payload

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "edrs-file-number",
      "valueString": "1234"
    },
    {
      "name": "jurisdiction defined key",
      "valueString": "value"
    },
    {
      "name": "mdi-document",
      "resource": {
        [Your MDI document bundle goes here in JSON or XML.]
      }
    }
  ]
}
```

*In Parameters* includes parameters that can be used for the update operation.

UPDATE API allows custom parameters (labeled as *Jurisdiction defined parameters*). They are locally defined parameters. It can be used in any ways by the systems that defined the parameters. If *Jurisdiction defined parameters* exist but cannot be understood, they should be ignored and should NOT cause any error.

The *mdi-document*, is a death certificate document in MDI FHIR IG. If CMS is updating the complete death certificate, then all the required data elements should exist in the document.

Partial document is allowed if CMS needs to update only portion of death certificate document. However, to conform to MDI FHIR IG, any empty required fields must be extended to include data-absent-reason extension.

The response for a successful UPDATE API should be 200 OK. The payload is not required in the response. If EDRS or CMS needs to respond with some data in the response, the Parameters resource can be used. EDRS and CMS can use the same parameters as *In Parameters* parameters. If the submitted document will be included in the response body, then “mdi-document” parameter key should be used.

If the API operation was successful, but there were some warnings that EDRS wants to send back to CMS, then parameter key, “warning”, should be used. And, “resource” should be used to include OperationOutcome resource. If the API operations were failed, then the response should be OperationOutcome resource with a HTTP error status code. Please see the example of response below.

Ex. **Response** if the operation was successful, and EDRS wanted to respond with updated data.

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "jurisdiction defined key1",
      "valueString": "value1"
    },
    {
      "name": "jurisdiction defined key2",
      "valueString": "value2"
    },
    {
      "name": "mdi-document",
      "resource": {
        "MDI document bundle"
      }
    },
    {
      "name": "warning",
      "resource": {
        "OperationOutcome resource"
      }
    }
  ]
}
```

**Response** if error occurred.

```
{
  "resourceType": "OperationOutcome",
  "id": "searchfail",
  "text": {
    "status": "generated",
    "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">\n      <p>The &quot;case_↵
↵number&quot; 1234 does not exist</p>\n    </div>"
  },
  "issue": [
    {
      "severity": "fatal",
      "code": "case-invalid",
      "details": {
        "text": "The \"case number\" 1234 does not exist."
      }
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

    }
  }
]
}

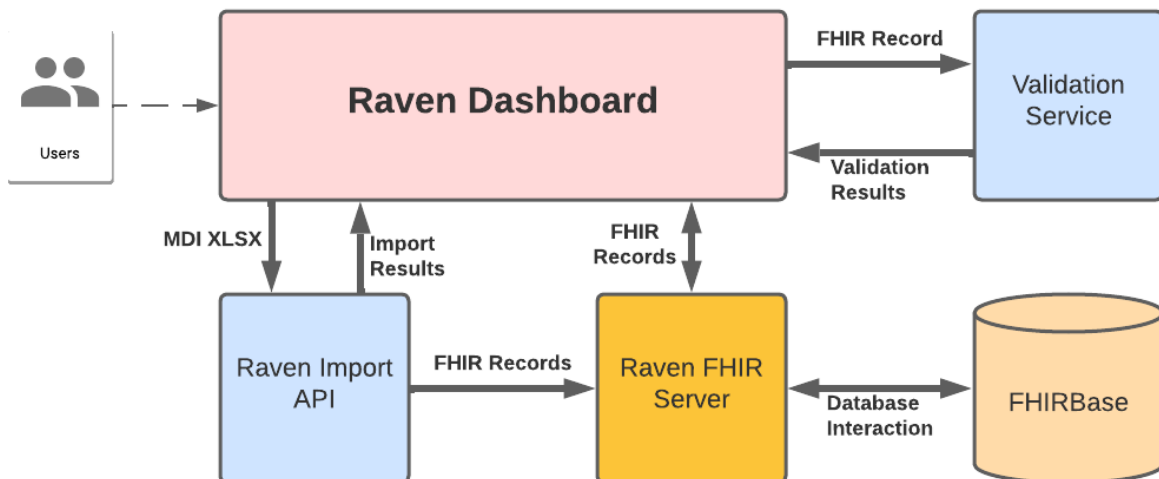
```

## Update using FHIR Messaging

If a messaging infrastructure is already in place, or if the content needs to be forwarded to another endpoint, it may be necessary to handle the target endpoint differently, given that the FHIR receiving endpoint is not the actual target. If this direction is deemed appropriate, the FHIR *process-message* operation (<https://hl7.org/fhir/R4/messageheader-operation-process-message.html>) can be employed.

If the decision is to utilize the *process-message* operation, the payload should take the form of a bundle, with the initial entry being a *MessageHeader* resource. Subsequent to this entry, parameters must be present, adhering to the specifications outlined in the Update API.

## 1.2.2 Component Overview



## Raven FHIR Server

The Raven platform stores MDI case data in the Raven FHIR server. The Raven FHIR server is developed using [HAPI FHIR Java library](#) with [fhirbase](#) as the backend database. Basic instance level of the FHIR APIs are implemented and available as,

```

GET [base FHIR Url]/Patient/[id] or [search parameters for SEARCH]
POST [base FHIR Url]/Patient with Patient Resource in the payload
DELETE [base FHIR Url]/Patient/[id]

```

In addition to the basic FHIR API, FHIR operation APIs are also implemented for transaction, batch, \$document, and \$process-message operations. \$process-message is the operation that Toxicology-to-CMS workflow is using.

## Bluejay FHIR Server

The Bluejay FHIR server is an instance that is configured to simulate EDRS that supports *Standard MDI API (MAPI)* (MDI-API). The Bluejay FHIR server is also based on the same code stack as Raven FHIR Server. Thus, the Bluejay FHIR server also provides the basic instance level of the FHIR APIs.

MDI-API that the Bluejay FHIR server currently supports is search API. Case Management Systems can test their MDI-API's search API feature with the Bluejay FHIR server. Please contact *our team* to arrange the testing.

## Raven Dashboard

The Raven Dashboard is the user interface for the Raven Platform. It consists of multiple core modules and features.

- **Record Importing and Viewing**
  - Record Viewer - View MDI case records currently stored on the Raven FHIR Server, with the ability to view the underlying FHIR structures in a human readable narrative, XML, or JSON.
  - Import Records - Import records to the Raven FHIR Server. Records can be submitted directly as a FHIR MDI-to-EDRS Document Bundle or from the MDI test case spreadsheet (XLSX file).
- **Validate and Compare**
  - FHIR Validator - UI wrapper for the official HL7 FHIR Validator command line tool.
  - Record Comparison (In Development) - Compare a user-generated FHIR MDI Document bundle created from a test case against a known valid rendering of the same test case.
- **Workflow Simulator (In Development)** - Move through steps of one of several test scenarios for various MDI related workflows, such as CMS to EDRS or a Toxicology Lab to CMS. The workflow simulator integrates other features.

The Raven Dashboard is a frontend TypeScript project developed using the Angular framework, leveraging major libraries such as Angular Material Design components.

## Raven Import API

The Raven Import API provides a backend service to import test cases from XLSX spreadsheets into the Raven FHIR Server as a FHIR MDI-to-EDRS Document Bundle. The API returns the results of the process to the Dashboard for rendering to users.

## Validation Service

The Validation Service is a web API which wraps the HL7 command line FHIR validation tool. The Raven Dashboard allows users to post a FHIR resource to the validation service, which returns the results of the validation.

### 1.2.3 Libraries

Raven provides helper libraries for developers who develop MDI IG functionality in their systems. In fact, Raven itself is using this library in order to produce and consume the MDI IG data. The libraries are available in Java and .NET and are also available as an open source. Details for each library are provided below.

#### MDI JavaLib

This Java Library is for the following FHIR Implementation Guides (IG)

- Medicolegal Death Investigation (MDI) FHIR IG | <http://hl7.org/fhir/us/mdi/>
- Occupational Data for Health (ODH) FHIR IG | <http://hl7.org/fhir/us/odh/STU1.1/>
- US Core FHIR IG | <https://www.hl7.org/fhir/us/core/>

The model profiles are built with the annotation package and base model definitions from HAPI-FHIR (<https://hapifhir.io/>)

MDI Javalib is available as a buildable source package in the MortalityReporting github organization ([https://github.com/MortalityReporting/MDI\\_javalib](https://github.com/MortalityReporting/MDI_javalib))

The library is built as a maven project and can be added as a dependency to existing projects (<https://maven.apache.org/>)

#### User can use the MDI Javalib to

- Deserialize JSON or XML into java objects
- Create new resources from an internal data source
- Serialize java objects into JSON or XML for transmission

#### MDI .NET

This .NET Library is for the following FHIR Implementation Guides (IG)

- Medicolegal Death Investigation (MDI) FHIR IG | <http://hl7.org/fhir/us/mdi/>
- US Public Health (US PH) FHIR IG | <https://build.fhir.org/ig/HL7/fhir-us-ph-common-library-ig/>
- US Core FHIR IG | <https://www.hl7.org/fhir/us/core/>

All profiles are built on top of standard .NET FHIR classes (<https://github.com/FirelyTeam/firely-net-sdk>).

ODH, US PH, and US Core IGs are base IGs that MDI and CBS IGs are built on. Thus, only referenced profiles in US PH, US Core, and ODH are implemented. The rest of the profiles will be added based on the needs.




MDI .NET libraries are available for download from nuget.org. Simply search by “MDI FHIR” at the nuget manager in Visual Studio. If you want to download from nuget.org, then the link will be <https://www.nuget.org/packages?q=MDI+FHIR>. The result will show up as follow, and C# developers need to install all three of libraries,





3 packages returned for MDI FHIR

Filter

- 
**fhir-ig-mdi-dotnet** by: [myung](#)  
 ↓ 88 total downloads ⌚ last updated 12 minutes ago 📦 Latest version: 1.0.0 🔗 [FHIR MDI IG](#)  
 .NET Library for HL7 FHIR® Implementation Guide: Medicolegal Death Investigation (MDI), Release 1 - US Realm (1.0.0 - STU 1 US)
- 
**fhir-ig-share-dotnet** by: [myung](#)  
 ↓ 90 total downloads ⌚ last updated 15 minutes ago 📦 Latest version: 0.2.2 🔗 [FHIR Share IG MDI CBS](#)  
 .NET Library for Commonly Shared Definitions in FHIR MDI and CBS IG
- 
**fhir-ig-uscore-dotnet** by: [myung](#)  
 ↓ 102 total downloads ⌚ last updated 13 minutes ago 📦 Latest version: 0.2.1 🔗 [FHIR US Core USCore IG](#)  
 .NET Library for US Core FHIR IG v5.0.1 - not all resource(s) are implemented. Only refereced ones from CBS and MDI are implemented

Source codes are also available for developers who are willing to contribute to the IG library developement in .net - <https://github.com/MortalityReporting/fhir-ig-dotnet>

## 1.3 About Raven Team

Jon Duke	Project Director
Alexandra Ramirez, Julie Mittelstedt, Marla Gorges	Project Manager
Myung Choi	Project Lead, MDI FHIR Server, MDI-API, MDI .NET Library
Michael Riley	MDI FHIR Import/Mapper, Validator, MDI Java Library, Community Engagement
Elizabeth Shivers	Documentation Lead, DevOps/CI, User Experience and Interface (Dashboard), MDI-API IG
Plamen Tassev	User Interface (Dashboard) Lead, Case Viewer, Validator
Andrew Stevens	MDI-API IG Lead
Russell Mitchell	User Interface (Dashboard)

For any questions or requests, please use [Zulip #Medicolegal Death Investigation Stream](#).

Raven is released under the [Apache License 2.0](#).

Public Raven URL: <https://apps.hdap.gatech.edu/raven/>

Source repositories for Raven can be found in the [GitHub Mortality Reporting Organization](#).

---

**Note:**

- All data shown is synthetic for demonstration purposes only and does not represent actual cases or decedents.\*
- Screenshots may be taken from earlier internal versions of Raven and may not be 100% accurate to the final release.\*

If you find an error in the Raven platform and documentation, please go to the “[About Raven Team](#)” page and let us know!

---