Measure Authoring Development Integrated Environment (MADiE) Test Case JSON Guide

Version 2.0

Revision History

Version	Date	Author	Revisions
1.0	11/9/2022	ICF	Initial MADiE Test Case JSON Guide
1.1	2/13/2023	ICF	Added description and instruction for use of the QI-Corev4.1.1 Test Case Template and test case QI-Corev4.1.1 Example Bundle version 2
1.2	3/23/2023	ICF	Added instructions for inclusion of meta.profile in bundle resources using QI-Core 4.1.1 profiles
2.0	3/11/2025	ICF	Removed "Exporting Test Cases from Bonnie FHIR" section. Add references for QI-Core version 6.0.0 Added description and instruction for use of the QI-Corev6.0.0 Test Case Template and test case QI-Corev6Example Bundle

Note: Please refer to the MADiE User Guide for more information regarding MADiE functionality. The focus of this document is providing guidance on MADiE test case JSON. The MADiE User Guide is located on the MADiE public site, Training & Resources tab.

TABLE OF CONTENTS

1 Introduction to Javascript object notation (JSON) 5

1.1 About MADiE Test Cases 5

2 JSON Syntax 6

2.1 JSON Datatypes and Rules 6

2.2 Example -The FHIR Patient Resource in JSON 6

2.3 Complex Datatypes and Arrays 7

3 CONSTRUCTING FHIR BUNDLES IN JSON 8

3.1 Structure 8

3.2 Collection Bundles 8

3.3 Example of a Measure Denominator Test Case Bundle 9

3.4 Example of a Measure Denominator Exclusion Test Case Bundle 12

3.5 Transaction Bundles 14

4 STEPS FOR CREATING A NEW TEST CASE IN MADIE 14

6 TIPS FOR USING QI-CORE PROFILE STRUCTURE DEFINITIONS 19

6.1 Identifying the QI-Core v4.1.1 Profile Official URL for use in the resource meta 19

6.2 Profile Content 21

6.3 Cardinality 22

6.4 DATA Type 22

7 EXAMPLE TEST CASE BUNDLES 22

7.1 QI-Corev4.1.1 Example Bundle version 2 23

7.1.1 Instructions for Use 24

7.2 QI-Corev6Example Bundle 25

7.2.1 Instructions for Use 26

1 INTRODUCTION TO JAVASCRIPT OBJECT NOTATION (JSON)

JavaScript Object Notation (JSON) is a lightweight, text-based language for the purpose of storage, representation and exchange of structured data within and between applications. Because JSON is simple to read, compact and is supportive of commonly used data types and can support hierarchical structures, JSON has become a standard for representing and exchanging data in web-based applications and APIs. It is also a supported standard for representing FHIR data and metadata.

1.1 ABOUT MADIE TEST CASES

Measure Authoring Development Integrated Environment (MADiE) test cases are constructed using JSON for representing measures and test case bundles. In MADiE, bundles are JSON files, each representing a measure test case, typically for a population used in the measure such as Initial Population, Denominator and Numerator with criteria for meeting success or failure. For example, an Initial Patient resource may contain patient and encounter data or a procedure. Test case bundles can be created directly within the MADiE tooling environment which offers a window to paste JSON and validate and includes validation errors and highlighting of the code segment that fails validation.

At this time, all synthetic test cases must be constructed in JSON. The JSON test cases may be constructed within the MADiE application, or they may be constructed in an external editor and pasted into the MADiE JSON panel. When test cases are developed externally, it is recommended that use of an editing tool supporting JSON validation is employed. JSON test cases in MADiE must be syntactically correct and must be mapped correctly to QI-Core profiles to support the data model and correctly meet the intent of the measure. MADiE offers functionality to assist the developer in identifying JSON errors with additional features to help evaluate test case alignment with measure intent.

2 JSON SYNTAX

JSON structure at simplest form includes paired data identifiers and values separated by tokens in the form of a name followed by a colon literal and a value with comma separation within an object. For example, “first name”: “Anna”. The following rules apply to JSON syntax.

2.1 JSON DATATYPES AND RULES

- JSON is case sensitive.
- Supported data types include string, number, Boolean, null, object and array.
- Numbers can be integers or floating point.
- At the lowest level of granularity, data is specified as name and value pairs separated by a colon “:” symbol. For example: “first name”: “Jonathan”
- Data elements are separated by commas.
- Objects or complex data types are enclosed within curly braces {}.
- Array data is enclosed within square brackets [].
- Quotation marks are used to surround names, data when not numeric or Boolean.

2.2 EXAMPLE -THE FHIR PATIENT RESOURCE IN JSON

Below is a snippet from the FHIR v4.0.1 Patient Resource in JSON format.

```
{
    "resource": {
        "resourceType": "Patient",
        "id": "ip1-pass",
        "identifier": [
            {
                "use": "usual",
                "type": {
                    "coding": [
                        {
                            "system": "http://terminology.hl7.org/CodeSystem/v2-0203",
                            "code": "MR",
                            "display": "Medical Record Number"
                        }
                    ]
                },
                "system": "http://hospital.smarthealthit.org",
                "value": "9999999911"
            }
        ],
        "name": [
            {
                "family": "Pass",
                "given": [
                    "John"
                ]
            }
        ],
        "gender": "male",
        "birthDate": "1964-06-30"
    }
}
```


Figure 1: FHIR Patient Resource

The JSON file contains a single patient.

Pairing names are always quoted and commas are used to delineate data elements. Unless the value is numeric or Boolean, the pairing value is also quoted. The cardinality of the element will determine if it needs to be enclosed in square brackets indicative of array values, or elements which can have more than one value. Complex datatype elements will be enclosed in curly braces.

2.3 COMPLEX DATATYPES AND ARRAYS

Using the HL7 FHIR Resource structure definition for the Patient resource, the cardinality and complexity of each object can be obtained. The datatype is hyperlinked In the HL7 FHIR Resource or Profile structure definition. Name and Coding are complex datatypes in this example. Additionally, a given name may have more than one value and is also an array as denoted below with square brackets.

3 CONSTRUCTING FHIR BUNDLES IN JSON

MADiE test cases in JSON follow the structure of a FHIR Bundle. A bundle is another resource in FHIR. For simplicity, JSON bundles may be thought of as means of combining a group of resources for the server in a transaction which reduces roundtrips to the server and mitigates loss of referential integrity. MADiE test file bundles can be thought of as the resources necessary to execute the test case and meet its intended results. In most cases it will include a patient resource and may include encounters, conditions, procedures, observations, and other resources.

MADiE test case bundles may be of type collection (value='collection') or transaction (value='transaction'). Special rules in FHIR apply to different kinds of bundles. For example, the resources in bundles of type transaction will also contain a FHIR operation such as POST or PUT. For more information on Bundle resources and what rules are applicable to each type of bundle, refer to the HL7 Bundle resource. resource.

3.1 STRUCTURE

Test case bundles must contain a resource type of 'Bundle', a string identifier which may be generated or named for the test case. The Bundle type may be a collection or a transaction. This is followed by an array of resource objects needed for the test case.

3.2 COLLECTION BUNDLES

The bundle begins with a Resource type of 'Bundle' and a bundle id and type of value collection. This is followed by an entry array consisting of the test case resources. In this case, there are two resources included for patient and procedure. Each resource must include a unique identifier pairing and a fullUrl pairing. The id value must be unique for each resource in the JSON bundle however it may also be used in reference elements referring to another resource of a resource may be used in subsequent resource references such as an encounter reference to a patient within the bundle.

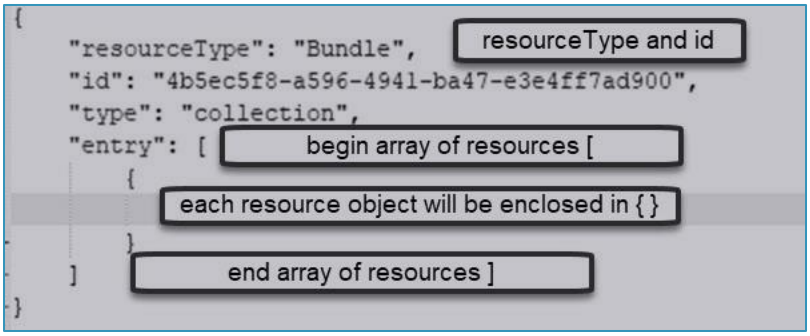

Figure 2: Bundle Structure

When the measure model is QI-Core, the best practice is to use QI-CORE profiles for bundle resources to avoid unexpected results. The HL7 FHIR QI-Core Implementation Guide can be followed for guidance in populating test case resource entries in JSON bundles.

Note: It may be helpful to use a UUID generator for assignment of id elements within bundles.

Steps to populate resource entries which may be helpful are:

1. Identify data elements needed for your measure
2. Map additional data elements needed for the measure to the corresponding profile requirement.
3. Reference the QI-Core Profiles to determine additional data elements necessary as determined by the element cardinality
4. Note the datatype of the data element on the QI-Core profile structure for non-primitive data types.

3.3 EXAMPLE OF A MEASURE DENOMINATOR TEST CASE BUNDLE

Consider a JSON bundle for a test case meeting a denominator population for a measure requiring a patient aged between 18 and 65 and a cataract surgical procedure during the measurement year. The JSON bundle will require a patient and procedure resource and will reference the QI-Core version 4.1.1 Profiles for QICorePatient and QICoreProcedure.

The JSON can be started with the collection bundle structure as illustrated in Figure 1 and replacing the bundle id with another UUID.

Profile	Data elements needed for measure	Data Elements required by Profile
QICorePatient	birthDate	identifier, name, sex

It is important to note that each profile may require data elements that the measure is not necessarily using. For example, when examining the QICorePatient profile structure definition you will see the profile requires that at least one identifier data element of data type Identifier, a complex datatype. It also requires at least one name of datatype HumanName, also a complex datatype. Cardinality mandates a sex is also entered. Although it is not required by the profile, the measure requirement needs a birthdate data element.

Enter the patient entry as an object within the collection array. The fullUrl and id are each unique for each entry. Notably this includes the ids of resources which have more than one instance in the bundle such as two encounters. Meta is populated to reflect the correct QI-Core profile being used. A medical record number is included to fulfill the requirement for an identifier of data type Identifier in the profile structure definition. Additionally, Name is included conforming to FHIR data type HumanName and gender is added conforming to the profile definition (datatype code) and binding to a code in VSAC value set ONC "ONC Administrative Sex". Last a birthdate element is added to the entry object. The profile for QI-Core Patient also indicates the datatype for birthdate.

The object is closed with a curly brace} and an added comma to indicate another entry follows.

Figure 3: JSON Bundle with QICorePatient Entry for Denominator Test Case

The same steps can be followed to add the procedure with reference to the QICoreProcedure profile structure definition. The denominator requires that the patient have a cataract procedure in a VSAC value set which is completed and is performed during the measurement period. The procedure will also need to reference the patient in the first entry. Note that for QI-Core, the resource element includes a meta entry identifying the QI-Core profile.

Referencing the QICoreProcedure profile it can be determined that the patient is mapped to the subject data element referencing the QICorePatient id. The status data element has a required binding for value. Performed is a choice data element which can be populated either performedDateTime or performedPeriod which are FHIR datatypes dateTime and Period, respectively. Code for the procedure is a complex datatype of type CodeableConcept and this datatype must be referenced to construct it correctly. The completed entry for the cataract procedure.

Figure 4: Completed Procedure Object

Since this test case bundle needs only the patient and procedure the JSON object array can be closed with a square bracket and the bundle object is closed with a curly bracket.

3.4 EXAMPLE OF A MEASURE DENOMINATOR EXCLUSION TEST CASE BUNDLE

The measure may include denominator exclusions. In this example, exclusions include value sets of conditions that direct the test case into the denominator exclusion population so long as the condition is active, the code is a member of an exclusionary value set for the member and the onset of the condition must predates the denominator procedure. Again, reference to the QICore Condition profile is helpful to determine the mapping, data types and cardinality.

```
{
    "fullUrl": "33e3e44c-9ace-11ed-9919-f340a01f594c",
    "resource": {
        "resourceType": "Condition",
        "id": "Condition-2",
        "meta": {
            "profile": [
                "http://hl7.org/fhir/us/qicore/StructureDefinition/qicore-condition"
            ]
        },
        "clinicalStatus": {
            "coding": [
                {
                    "system": "http://terminology.hl7.org/CodeSystem/condition-clinical",
                    "code": "active"
                }
            ]
        },
        "verificationStatus": {
            "coding": [
                {
                    "system": "http://terminology.hl7.org/CodeSystem/condition-ver-status",
                    "code": "confirmed"
                }
            ]
        },
        "category": [
            {
                "coding": [
                    {
                        "system": "http://terminology.hl7.org/CodeSystem/condition-category",
                        "code": "encounter-diagnosis",
                        "display": "Encounter Diagnosis"
                    }
                ]
            }
        ],
        "severity": {
            "coding": [
                {
                    "system": "http://snomed.info/sct",
                    "code": "24484000",
                    "display": "Severe (severity modifier)"
                }
            ]
        },
        "code": {
            "coding": [
                {
                    "system": "http://hl7.org/fhir/sid/icd-10-cm",
                    "code": "I25.5",
                    "display": "Ischemic cardiomyopathy"
                }
            ]
        },
        "subject": {
            "reference": "Patient/Patient-1"
        },
        "onsetDateTime": "2022-01-02"
    }
},
```


Figure 5: Adding a QICore Condition Entry to the Bundle

3.5 TRANSACTION BUNDLES

Transaction bundles in FHIR are most useful when specific supporting operations are desired. For transaction bundles the type element is set to 'transaction'. An additional element must follow each resource with instruction to POST/PUT the resource.

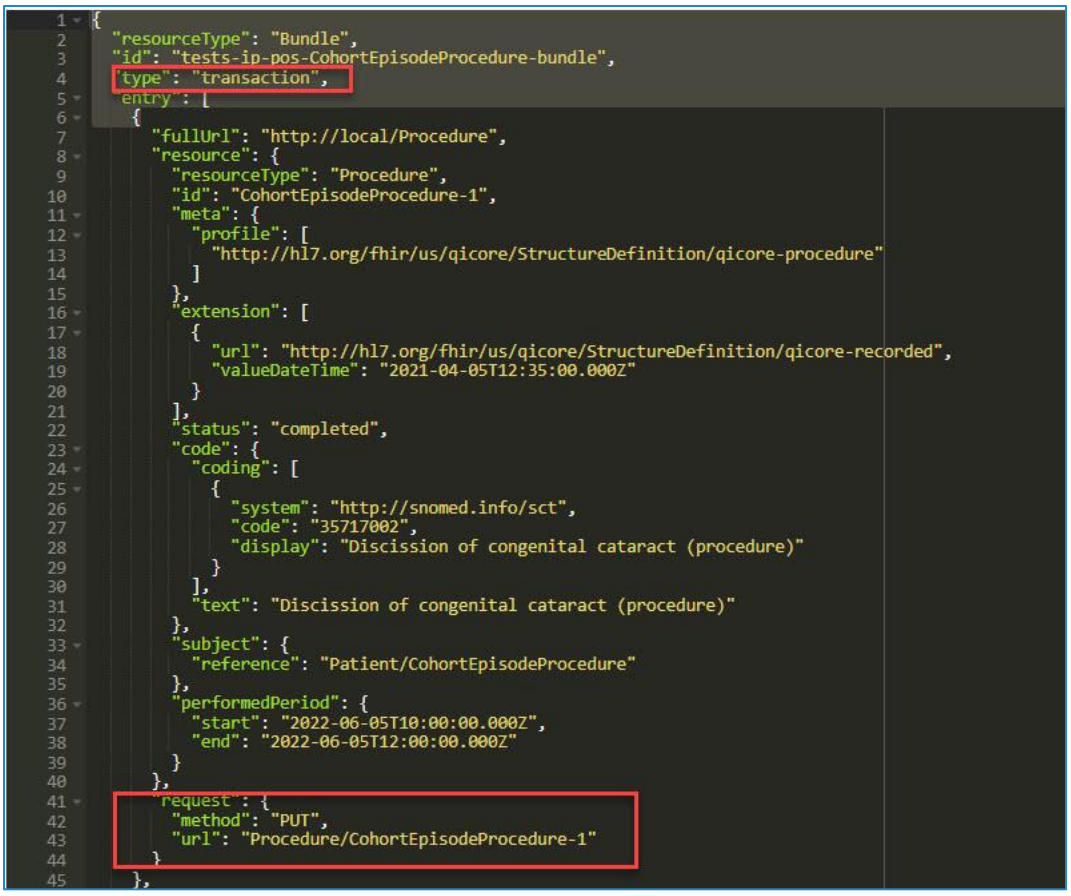

Figure 6: Snippet of a Transaction Bundle

4 STEPS FOR CREATING A NEW TEST CASE IN MADIE

The following example details the construction of a FHIR bundle of type collection to be used in a MADIE test case for a Measure Denominator.

1. Sign into MADiE and select the intended measure to add test cases to with the View/Edit button.

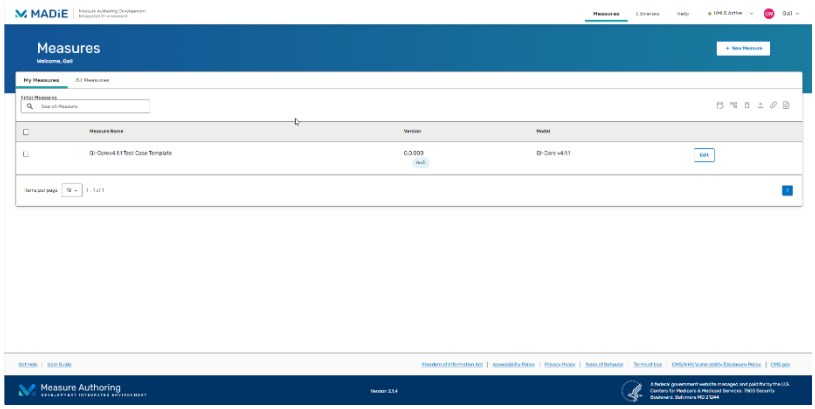

Figure 7: MADiE Measures Panel

2. With the intended measure open for edit, navigate to the Test Cases tab.

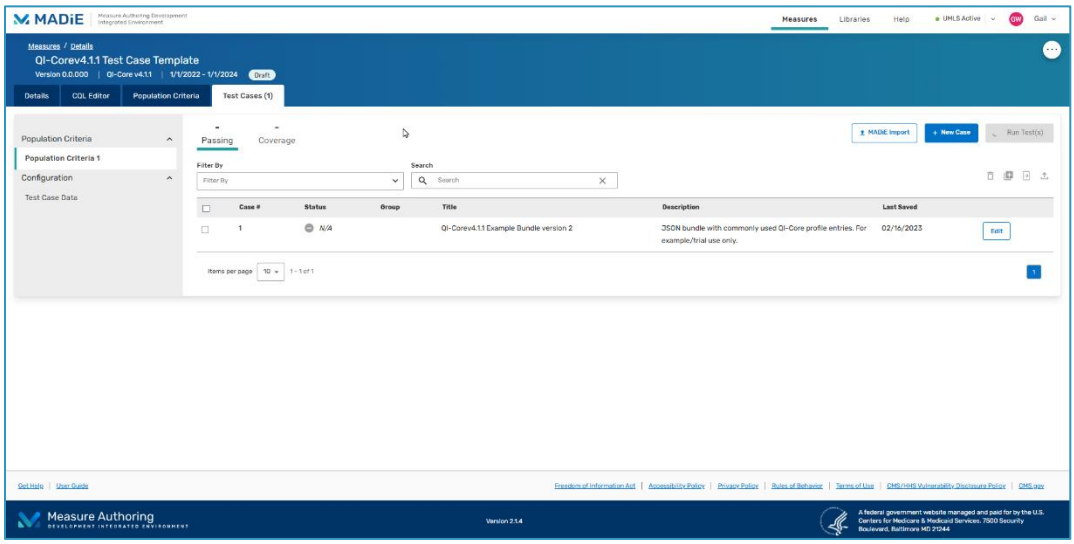

Figure 8 Measure Test Cases Tab

3. The Test Case tab will open. Click the "+ New Case" button to open the window. Enter the name and a description of the test case and click the Save button.

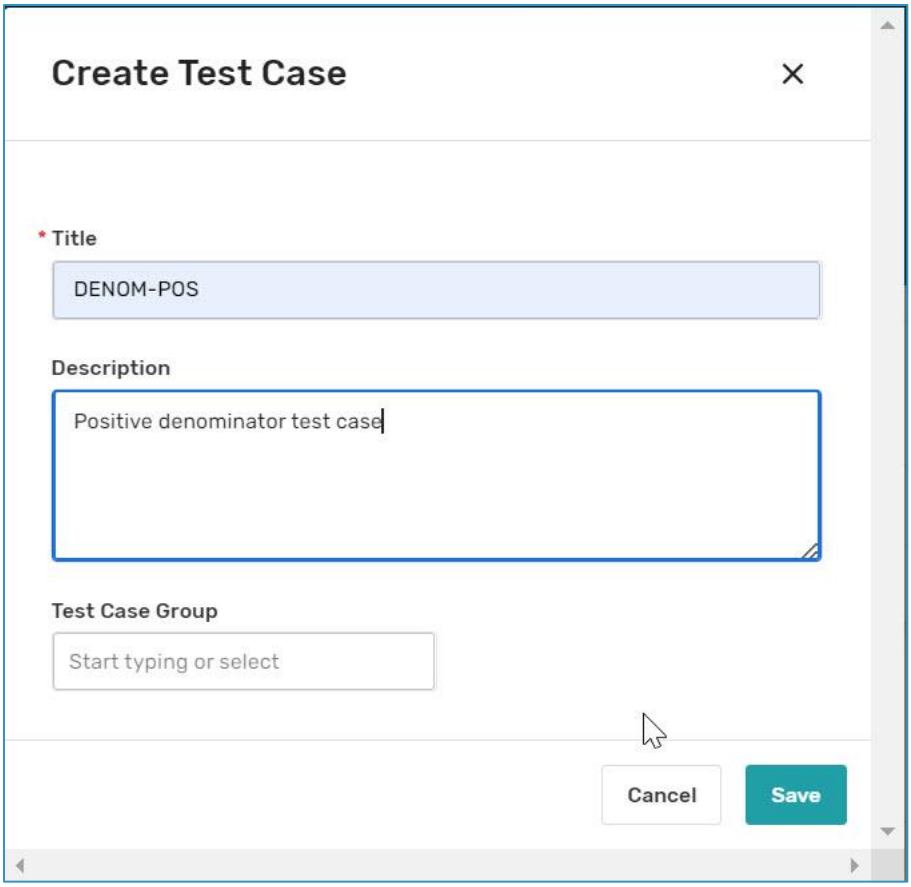

Figure 9: Create Test Case Modal

4. The test case Status will display “Invalid” until JSON is entered and passes validation.

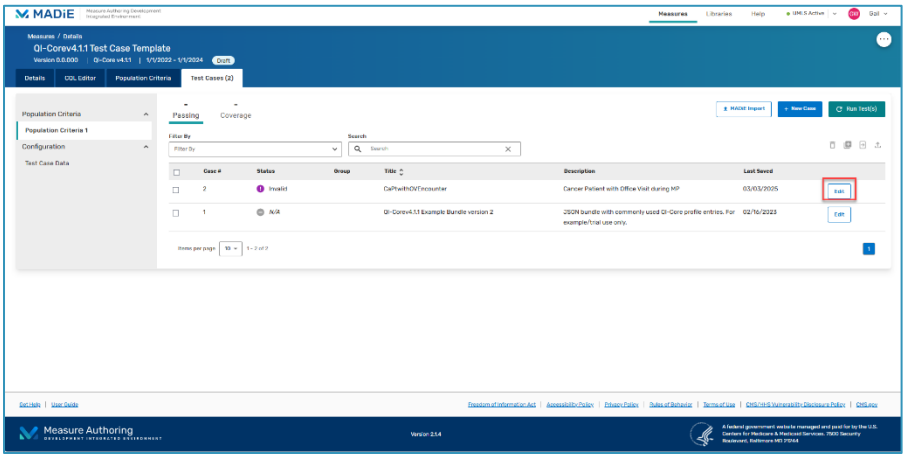

Figure 10: Invalid Test Case

5. Enter the test case editor by clicking on the Edit button. The measure CQL appears in the right window on the screen which may be a useful reference while working on the test case.

The left pane is a blank editor where you can enter or paste your test case JSON. Users may also import JSON from a file using the Import button.

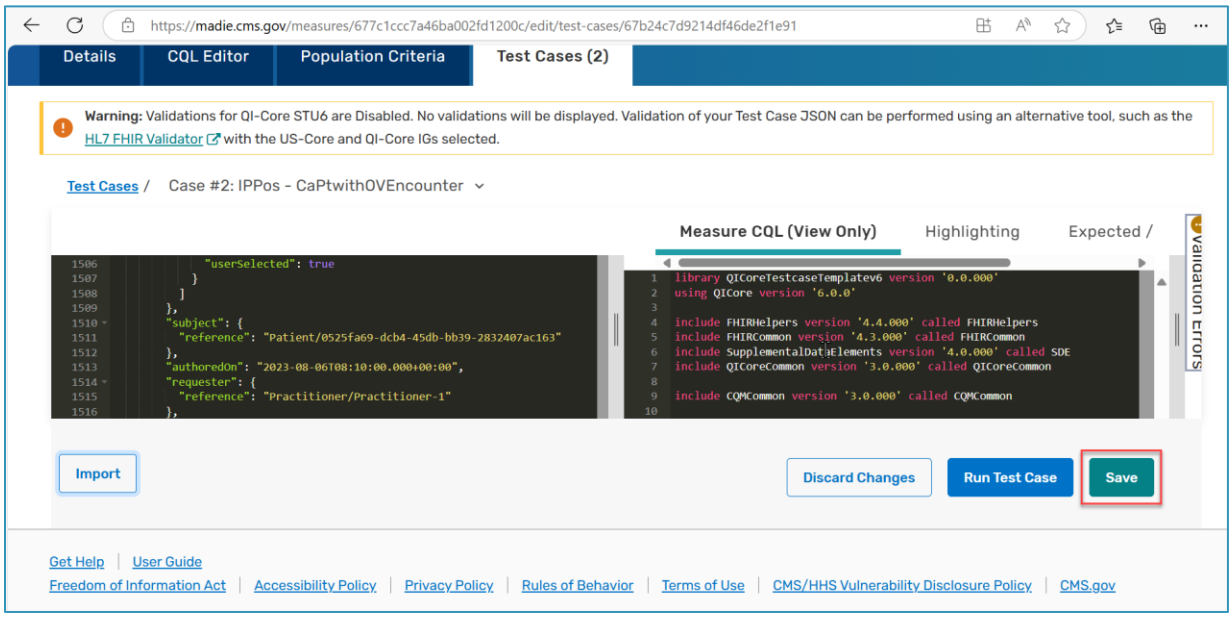

Figure 11: MADiE Test Case Editor

6. When the test case JSON is completed use the Save button on the bottom right of the window to save the changes. The Validation Errors icon will be red if the test case contains errors. When all validations errors are resolved the Status of the test case will no longer be Invalid. **Note:** Validations for QI-Core v6.0.0 test cases are currently disabled in MADiE. Validation of your Test Case JSON can be performed using an alternative tool, such as the HL7 FHIR Validator with the US-Core and QI-Core IGs selected.

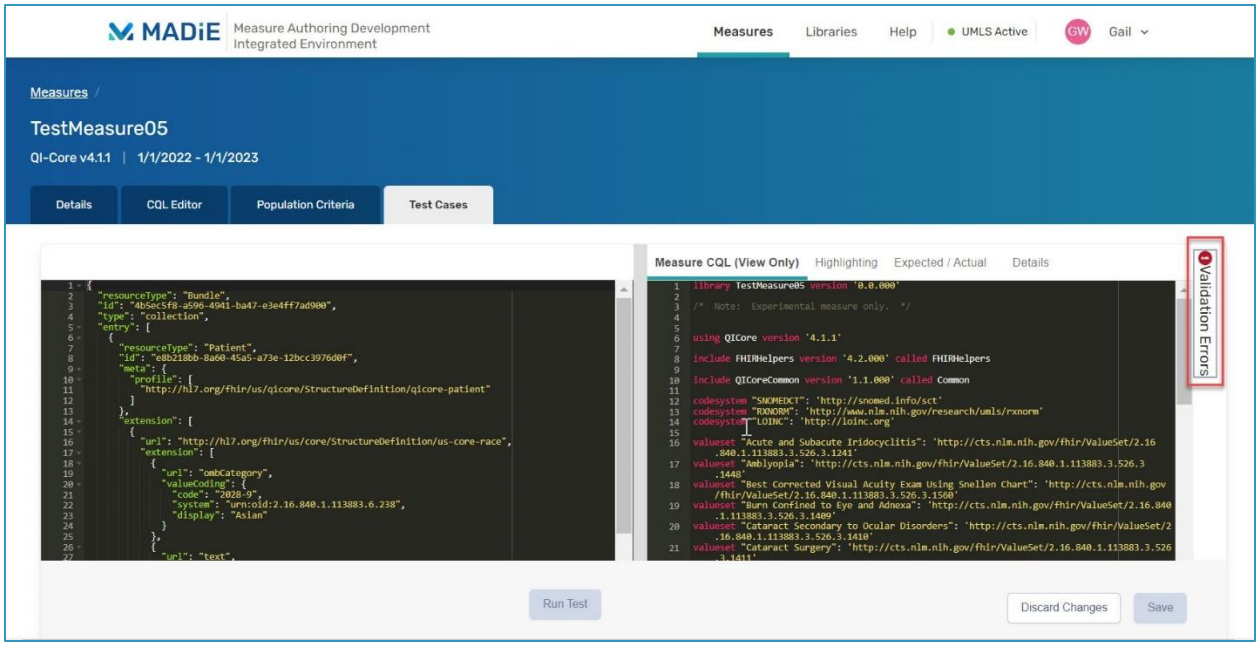

Figure 12: JSON Contains Errors and Does Not Pass Validation

7. Clicking on the Validation Errors icon will show the JSON errors which are preventing the test case from passing validation and need to be corrected. Each error must be corrected. Once they have click save and if there are no more errors the Validation icon will revert to black and white.

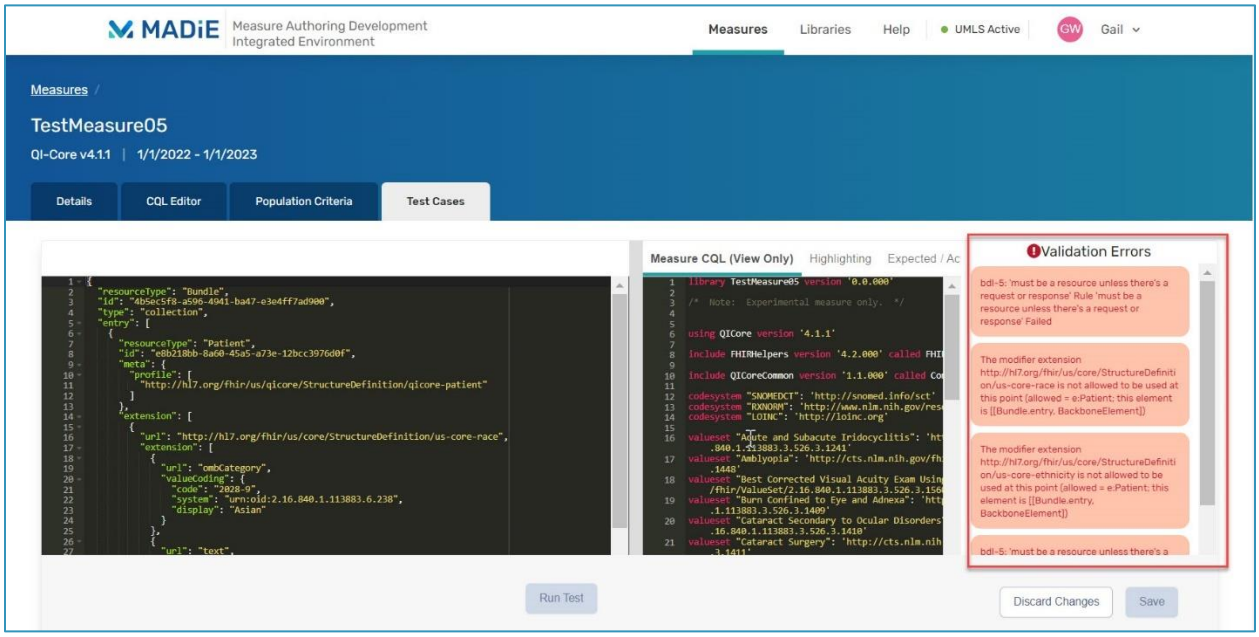

Figure 13: Validation Error Details Display by Clicking the Validation Icon

8. When the JSON passes validation, click on the Expected/Actual tab and enter the Expected results for the measure. Then click on the 'Run Test' button in the bottom center of the window.

   Note: Population values for the measure may vary depending upon the measure and its population definition and measure groups. The Actual results will be populated after running the Test Case.

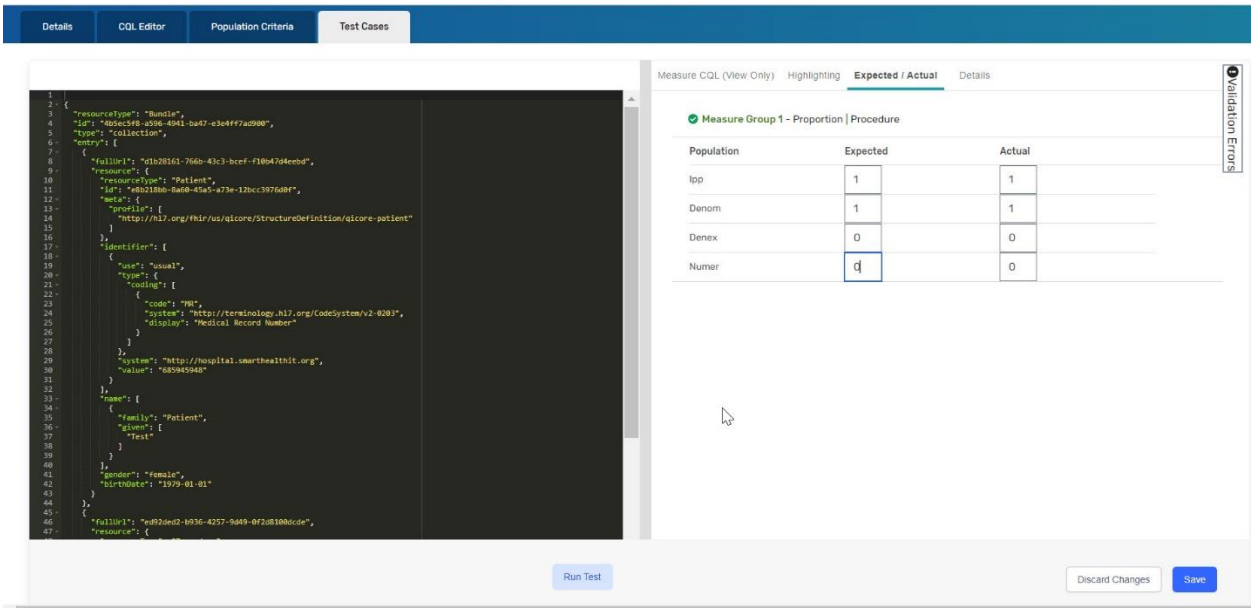

Figure 14: MADiE Test Case Expected and Actual Results

6 TIPS FOR USING QI-CORE PROFILE STRUCTURE DEFINITIONS

This section will reference the QI Core Implementation Guide Version 4.1.1 with focus on the QICorePatient profile.

Creating test tases in JSON that meet the profile requirements require an understanding of profile structure definitions.

6.1 IDENTIFYING THE QI-CORE V4.1.1 PROFILE OFFICIAL URL FOR USE IN THE RESOURCE META

The Official URL for the QI-Core profile is available here:

https://hl7.org/fhir/us/qicore/STU4.1.1/StructureDefinition-qicore-patient.html

The URL for the QI-Core profile is needed in the bundle resource meta.profile element. Each resource entry will need this information added if it is not already included as with the patient resource. The official URL for each profile is located on the QI-Core Implementation Guide in the Profile Section. Click on the QI-Core Profile of interest to open the profile details.

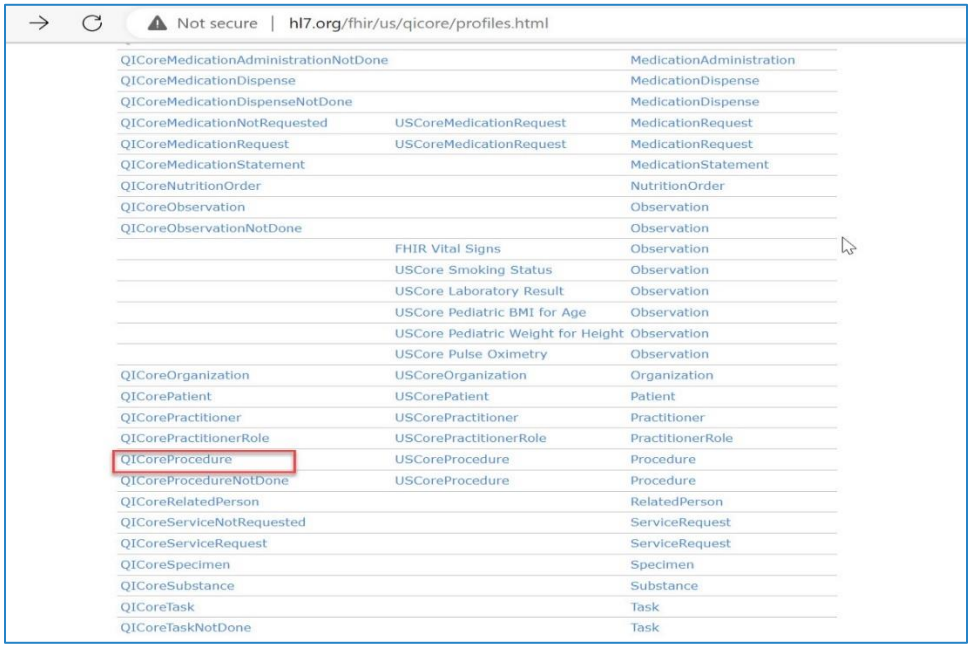

Figure 15: QI-Core v4.1.1 Profile List

Note the official URL on the profile details page. This will be used in the resource meta.profile entry in the bundle. Note that each resource will need the appropriate QI-Core profile official URL.

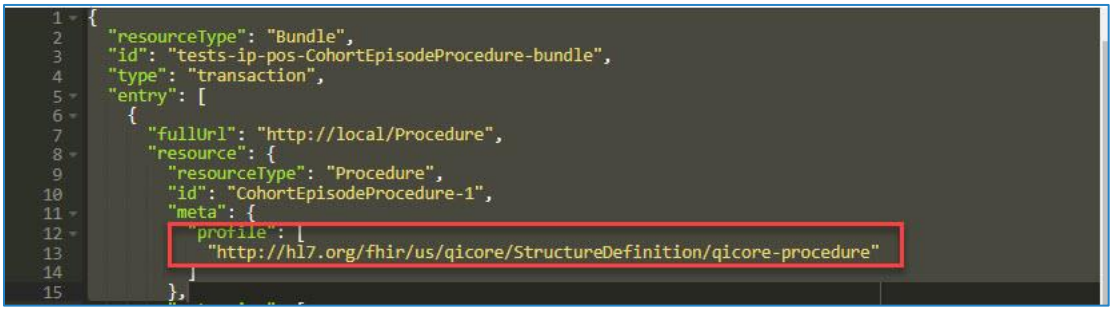

Figure 16: Profile URL Usage in Test Case Bundle

6.2 PROFILE CONTENT

On a profile structure definition, the snapshot view includes all properties which are in a base resource and added profile elements and may be preferable to the implementer creating the bundle resource (test case).

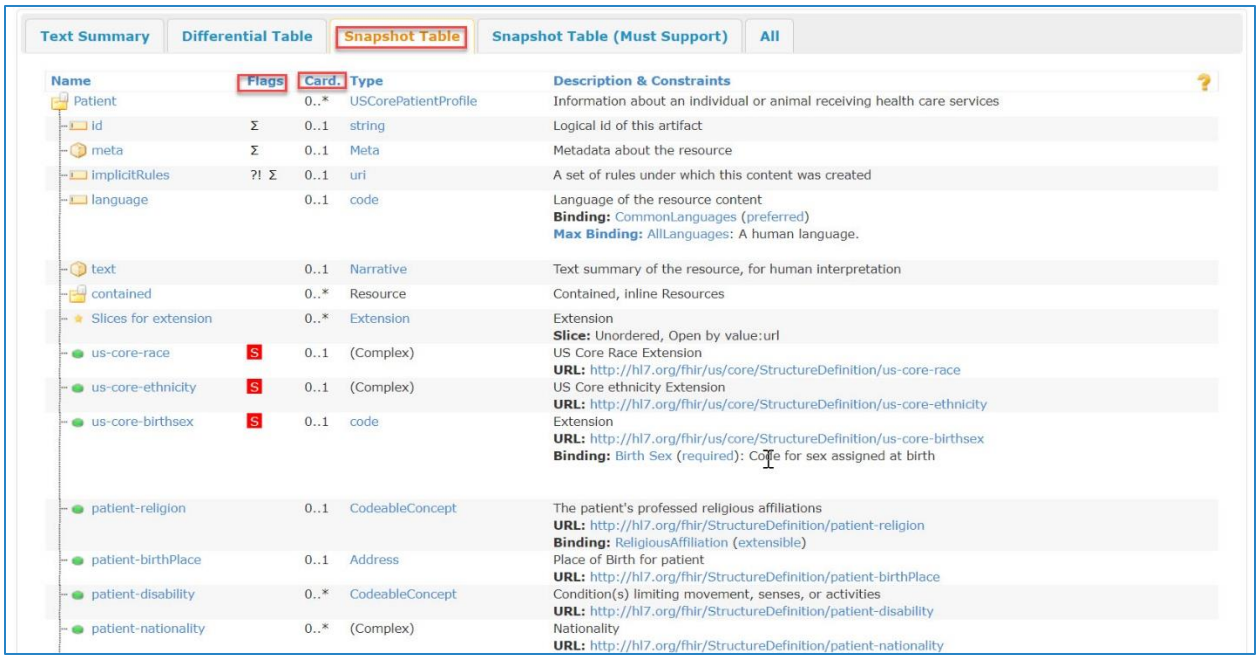

Figure 17: QI-Core Patient Snippet

Profile content will include information pertaining to property names and description, cardinality, and flags. The following definitions may be helpful.

Column	Content
Name	The name of the element in the resource (manifests as XML element name or JSON or RDF property name). Some names finish with `[x]` - the meaning of this is discussed below. In addition, this column contains an icon that denotes the underlying type of the content. The icons are described below
Flags	A set of information about the element that impacts how implementers handle them. The flags are described below
Card.	Cardinality: the lower and upper bounds on how many times this element is allowed to appear in the resource
Type	The type of the element (hyperlinked to the definition of the type). Note that the type of the element has one of two meanings, depending on whether the element has defined children. If the element has children, then the element has an anonymous type that specializes the given type. If the element has no children, then the element has properties and children as specified by the nominated type
Description & Constraints	A description of the element, and details about constraints that are applied to it. Particularly, for coded elements, information about which codes can be used. The description comes from ElementDefinition.short

Figure 18: Content of a Profile Structure Definition

Flags contain information about the element which impacts how implementers handle them. Important flags include:

?! : Modifier elements – a modifier element can change the interpretation of the resource. For example, a verification status.

I : Element is affected by Constraints. Also known as invariants. In FHIR this can change the meaning of the resource. For example, to indicate that a medication was not administered.

S: Must Support : implementations that produce or consume resources SHALL provide support for the element in some meaningful way.

6.3 CARDINALITY

Cardinality can be used to determine if an element is mandatory and needs to be populated in a resource within the test case bundle. Use the profile structure definition to determine if the element is required.

FHIR specification only defines four cardinalities: 0..1, 0..*, 1..1, 1..*.

0..1 and 0..* indicate the element is not required. It is left to the implementer to determine what is useful information. The element may have no or 1 value (0..1) or no and more than 1 values (0..*)

1..1 indicates the element is required and may only have one value

1..* indicates element is required and may have more than one value

6.4 DATA TYPE

Resource element datatype can be either complex or primitive. Complex data types are supertypes with additional elements within them. For example, an Address or a Codeable Concept. Primitive data types do not have additional elements within them.

This indicates to author the JSON it may require additional understanding of complex data types.

To do this, click the data type hyperlink on the profile structure definition. A deeper understanding of primitive and complex data types in FHIR can be found Foundation data types.

7 EXAMPLE TEST CASE BUNDLES

For each QI-Core version supported by MADiE example measures are available for users to view in MADiE with an included test case JSON bundle. This test case contains entries for many

QI-Core (v4.1.1 or v6.0.0 profiles respectively.) The purpose of the test case JSON bundles are to provide users with the framework for commonly used QI-Core profiles that can be leveraged with their own test case development. The purpose of the simple measures is to demonstrate how to construct JSON resources for profiles. Each sample measure has one test case containing commonly used profiles to include use of negation profiles. QI-Core 4.1.1 Example Profiles are discussed in QI-Corev4.1.1 Example Bundle Version 2. QI-Core 6.0.0 Example Profiles are discussed in QI-Corev6Example Bundle.

7.1 QI-COREV4.1.1 EXAMPLE BUNDLE VERSION 2

The following profiles are included:

- Adverse Event
- Allergy Intolerance (medication)
- Condition (2: Problem List, Diagnosis)
- Coverage
- DiagnosticReportLab
- Encounter (2: Inpatient, Ambulatory)
- Immunization
- ImmunizationNotDone
- Location
- Medication
- MedicationAdministration
- MedicationRequest
- Observation
- Organization (2: Health Provider, Payer)
- Patient
- Practitioner
- PractitionerRole
- Procedure

- ServiceNotRequested
- ServiceRequest
- Task

The QI-Core profiles examples in the JSON bundle are for guidance only and are not intended to fulfill any measure or testing requirement. Updates to reflect the specific details of your test case will be necessary. For example, test case(s) may require a different medication than what is provided in the profile example for MedicationAdministration. Measure developers may copy any of the profiles from the 'QI-Corev4.1.1 Example Bundle version 2' test case directly to test case(s) or use a source code editor (e.g., Notepad++ and Visual Studio Code) and make any updates needed to meet testing needs.

7.1.1 INSTRUCTIONS FOR USE

1. Select the All Measures table list and search for the measure 'QI-Corev4.1.1 Test Case Template' and open the measure by clicking "View" in the "Action" column. Once opened, click the Test Cases tab. Open the test case 'QI-Corev4.1.1 Example Bundle version 2' by clicking "View" in the "Action" column.
2. While viewing the test case, it may be helpful to copy the test case bundle and save as a JSON file for use with Visual Studio Code, Notepad++ or another editor that supports JSON.
3. Determine the QI-Core profile(s) available in the example test case that are needed for the test cases you are creating or editing. For example, the test cases may need the QI-Core Patient, Encounter, Condition and Procedure profiles.
4. Search the test case editor in MADiE or the external editor in use to locate the Resource example.
5. Copy the resource block and paste it into the measure test case bundle for your measure, beginning with the "{" above the "fullUrl" pairing above the resource entry until the closing "}" before the beginning of the next resource.

   Note: Test cases with multiple resources must separate each resource with a comma after the closing "}".
6. Edit the resource block to update the test case as needed for the measure.

Note: Edited values need to be QI-Core v4.1.1 compliant. For example, codes may need to be from a particular value set indicated by binding requirements.

7. After making changes ensure the bundle is valid JSON by using the editor feature or by pasting and saving in MADiE as a test case. If working in Visual Studio Code or a local editor, it may be helpful to format the JSON file and ensure it is syntactically correct before adding it to the MADiE test case bundle.

7.2 QI-COREV6EXAMPLE BUNDLE

The following profiles are included:

- Adverse Event
- Allergy Intolerance
- Condition Encounter Diagnosis
- Condition Problems Health Concerns
- Encounter (Inpatient, Ambulatory)
- Immunization
- Immunization Not Done
- Location
- Medication
- Medication Administration
- Medication Request
- Medication Not Requested
- Observation Lab
- Observation Screening Assessment
- Patient
- Practitioner
- Practitioner Role
- Procedure
- Service Request

- Service Not Requested
- Simple Observation
- Task

7.2.1 INSTRUCTIONS FOR USE

1. Select the All Measures table list and search for the measure 'QI-Corev6.0.0 Test Case Template' and open the measure by clicking "View" in the "Action" column. Once opened, click the Test Cases tab. Open the test case 'QI-Corev6Example Bundle' by clicking on "View" in the "Action" column.