Measure Authoring Development Integrated Environment (MADiE) QI-Core Test Case User Guide

Version 1.0.0

March 17th, 2026

Record of Changes

Version	Date	Author / Owner	Description of Change
1.0.0	March 17, 2026	Nikki Hunter / ICF	Updates for MADiE 3.0.0

Table of Contents

1 Introduction

This QI-Core Test Case User Guide is designed to help measure developers understand how to create and manage QI-Core–based test cases within the MADiE Test Cases tab. In this guide, you will learn how to create, maintain, and validate QI-Core test cases.

This guide focuses solely on the features and processes unique to QI-Core test cases. It assumes users have an active MADiE account and already created a QI-Core measure. For information on other aspects of MADiE, such as creating measures, authoring CQL logic, managing populations or QDM test cases, please refer to the main MADiE User Guide. That resource provides comprehensive instructions for all other components of the measure creation and management process and can be found on the Training & Resources tab on the MADiE public website.

2 Test Cases List Page

The MADiE Test Cases list page is the same across all model types. For detailed information please see the MADiE User Guide found on the MADiE public website under the Training & Resources tab.

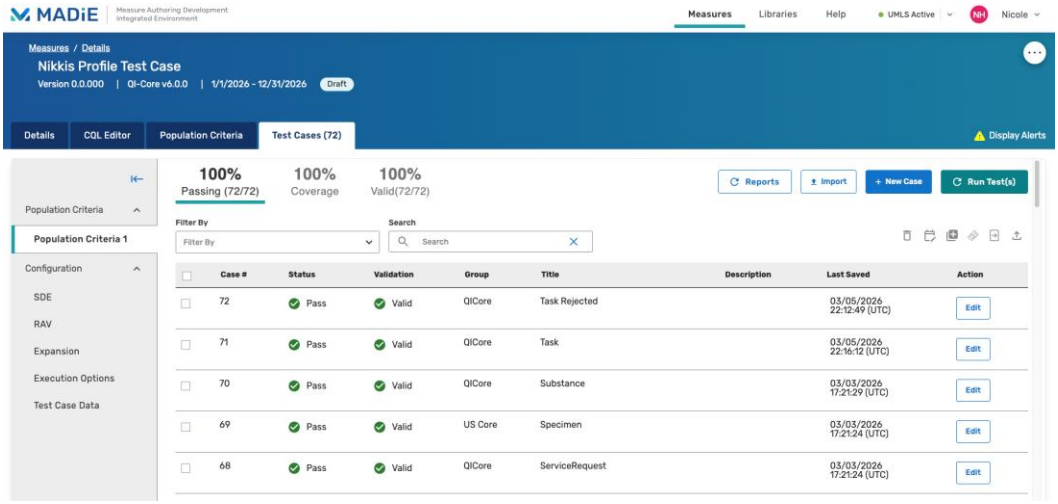

Image: QI-Core Test Case List Page

3 Create Test Case

Creating tests cases is the same across all model types. For detailed information please see the MADiE User Guide found on the MADiE public website under the Training & Resources tab.

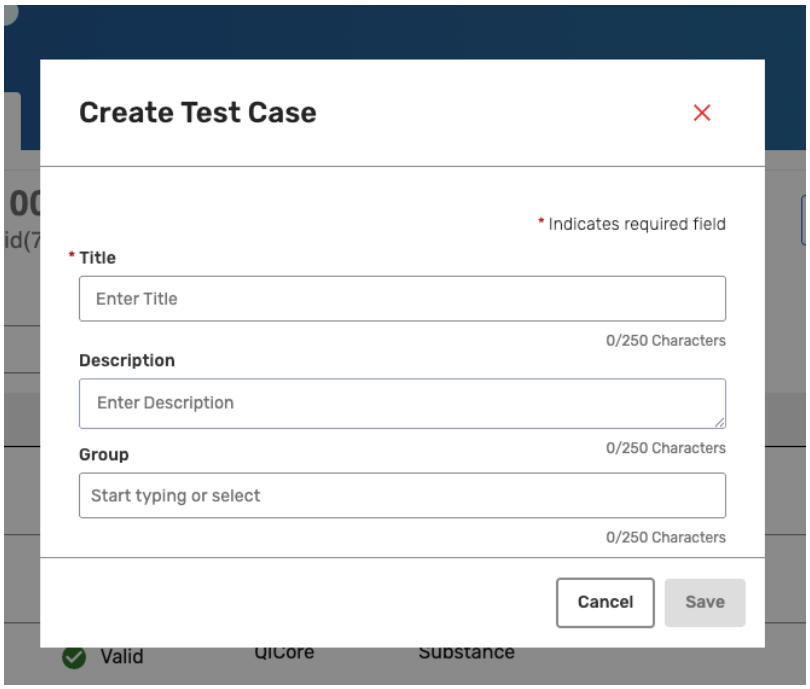

Image: Create Test Case Modal

4 Edit Test Cases

MADiE allows users with edit access to the measure to review, update, and refine QI-Core based test cases to ensure they accurately represent expected clinical scenarios and align with measure logic. Within this workspace, you can adjust key elements of each test case, including patient attributes, clinical data elements, QI-Core profiles, and expected population results, to validate that the measure behaves as intended across a variety of inputs.

Test cases can be edited in two ways: through the visual UI Builder or by working directly in the JSON. The Builder provides a guided, user-friendly interface for adding and managing elements, while the JSON view allows users to work hands-on with the test case structure itself. Both views are kept in sync, and users may switch between them at any time, choosing whichever editing method best fits their needs.

The UI Builder relies on syntactically valid JSON and cannot operate when the JSON contains syntax errors or cannot be parsed. If this occurs, the Available and Added tabs will be locked, and the system will display a message indicating that all JSON errors must be resolved before the UI Builder can be used. Once the JSON is corrected, the UI Builder automatically becomes available again, allowing users to continue working in the UI Builder or in JSON.

When a test case is opened for editing, the user will see two distinct sections. The first, on the left-hand side, is where the test case will be constructed, with tabs for Available, Added, and JSON. The second section is on the right-hand side, where the users will see information about

the measure, set the expected values, see results from test case execution, and update metadata. The tabs on the right-hand side are CQL, Highlighting, Expected/Actual, and Details. Between the sections is a draggable bar that allows the user to expand either section to provide screen real estate where the user most needs it. Navigation between tabs on the two sections can happen independently to allow users to customize the screen to meet their needs.

1. Available Tab
2. Added Tab
3. JSON Tab
4. Sliding Bar
5. CQL Tab
6. Highlighting Tab
7. Expected/Actual Tab
8. Details Tab
9. Validation Side Bar

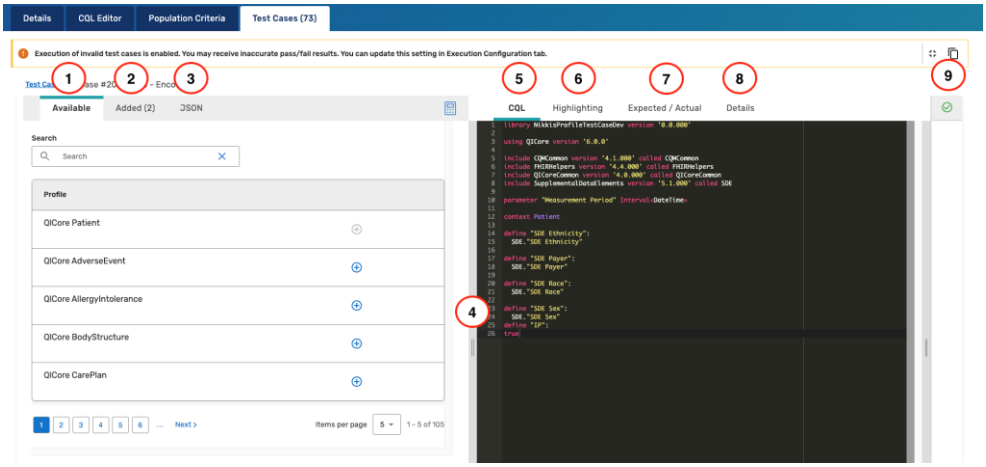

Image: Edit Test Case Page

See below to explore each tab on the edit test case page.

4.1 Available Tab

When a user has access to edit a measure, the Available tab is the starting position when opening a test case for editing. It allows users to browse the full set of QI-Core and US Core Profiles that can be added to a test case in the builder. Users will select the profiles needed t for constructing the test case.

The Available tab contains an alphabetically sorted table displaying all available profiles to be included in the test case. A profile is considered available if it is referenced in the measure's CQL. If the measure's CQL does not contain any profile reference, then all profiles will be available to select. In the table, each row represents a single profile, showing the profile name

and an Add button. Standard FHIR base resources do not appear in the Available list, regardless of whether they are referenced in the measure CQL.

The Available Profiles table is fully paginated. It initially displays five items per page, but users can change this setting to show a larger or smaller number of profiles at once. Page arrows allow users to navigate between pages easily. The table also remembers the user's pagination preferences — including the selected items-per-page value and the current page number — even if the user switches tabs, navigates away, refreshes the page, or closes and reopens the test case. This ensures a consistent experience across sessions.

1. **Profile Search**: Allows users to search for a specific profile. This is a contains search and case insensitive.
2. **Profile Name**: This is the list of profiles that are available to the users to add to the test case.
3. **Plus Icon**: Clicking this icon will add the profile to the test case.
4. **Pagination Controls**: These controls limit the number of profiles shown to the users at a time and allow users to navigate to other pages.

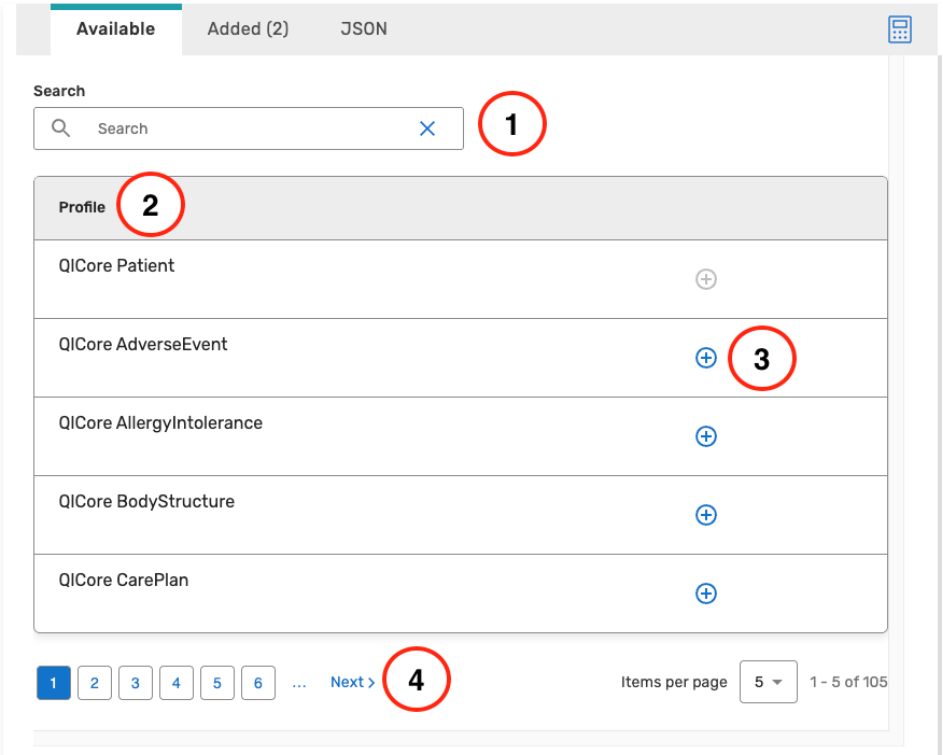

Image: Available Profiles Table

To begin constructing a test case, users should decide on the profiles needed to construct their test cases, locate them in the available profile table, and click the Plus icon. This action will make a working copy of the profile available for use in the test case. The user can navigate to the

Added tab and fill in information about the profile instance. The number in the Added tab will increment each time a profile instance is added, so the users can visually see that the profile was added. Users can add as many profile instances from the Available tab as needed prior to switching to the Added tab. To create a successful test case a Patient profile instance will need to be added. Due to this dependency, QICore Patient is the first profile in the list to select. MADiE supports only one Patient profile per test case. After the Patient profile instance is added, the Add button will be greyed out.

Once a profile instance is added, the JSON will be updated with the new entry and resource value. The following will be added by default:

1. **fullURL**: A valid fullURL with the profile and the profile id. The added profile is not saved to the database until the user clicks Save on the test case
2. **id**: MADiE will automatically assign an id for each profile. On addition of the profile, it will be set in the JSON automatically for the user to reference in other profiles.
3. **ResourceType**: Indicates the resource type of the profile the user entered.
4. **Meta**: Describes how the resource should be interpreted.

Example of JSON that would be added for a QI-Core patient:

```
{
  "fullUrl": "https://madie.cms.gov/Patient/740d65bf-5ab6-48c9-adf7-23b00e8d7d97",
  "resource": {
    "id": "740d65bf-5ab6-48c9-adf7-23b00e8d7d97",
    "resourceType": "Patient",
    "meta": {
      "profile": [
        "http://hl7.org/fhir/us/qicore/StructureDefinition/qicore-patient"
      ]
    }
  }
}
```

4.2 Added Tab

The Added tab consists of two main components: the Added Profile Table and the Edit Profile Card. When the tab first loads, only the Added Profile Table is displayed, giving users a clean overview of all profiles that have already been included in the test case. From there, users can select any profile in the table to open its corresponding Edit Profile Card. The Edit Profile Card allows users to add attributes, enter data, and refine the details for each profile. This layout keeps the interface simple at first glance, while still offering full editing capabilities as needed.

4.2.1 Added Profile Table

The Added Profile Table lists all profiles currently included in the test case. Each row shows the Profile Name and its system-generated ID, making it easy to identify and navigate to individual

profiles. Action icons on the right side of each row allow users to manage the selected profile quickly and efficiently.

Table Columns:

1. **Profile** – Displays the name of the profile added to the test case.
2. **ID** – Shows the unique ID generated by the system for that profile.
3. **Action Center** – Provides actions the user can take:
    a. **Delete** – Removes the profile from the test case. This delete is not saved to the database until the user clicks Save on the test case.)
    b. **Edit** – Opens the profile's Edit Profile Card (see section 4.2.2 for details.)

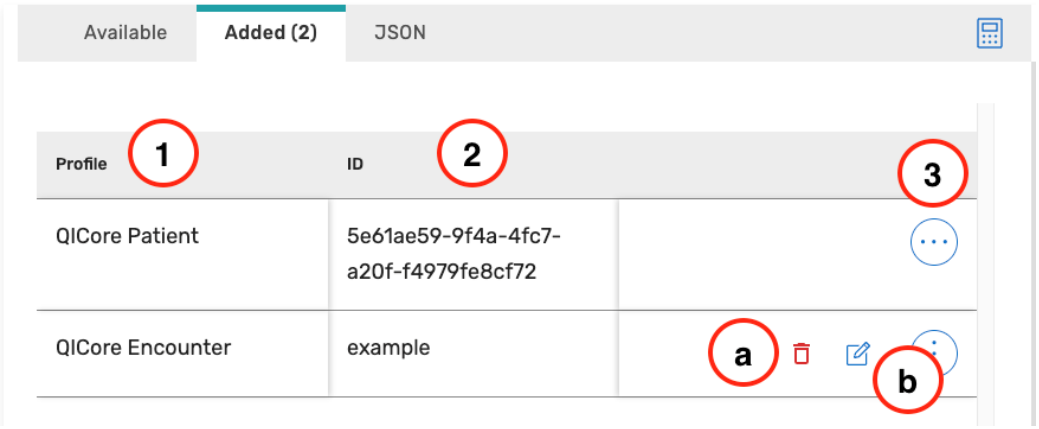

Image: Added Profile Table

The Added Profile Table is the main entry point for reviewing and managing all profiles included in the test case.

4.2.2 Edit Profile Card

The Edit Profile Card is where most of the Builder-based editing takes place. When a profile is opened for editing, its card appears above the Added Profile Table.

Card Elements**:**

1. **Profile Name** – Indicates which profile is currently being edited.
2. **ID** – Displays the ID of the profile being edited.
3. **Profile Type** – Indicates if the profile is QICore, USCore, or Base FHIR.
4. **Add Attribute(s)** – Opens the Add Attribute modal, allowing users to include additional attributes.

5. **Required Attributes** – These appear automatically based on the FHIR profile definition and are marked with an asterisk (*).
6. **Additionally Added Attributes** – Shows optional attributes that have been added by the user through the modal or manually via JSON. These do not contain an asterisk.
7. **Edit Attribute Section** – Displays the input fields for the selected attribute. Users must select Apply to add their entries to the UI representation. (These updates are not saved to the database until the user clicks Save on the test case.)

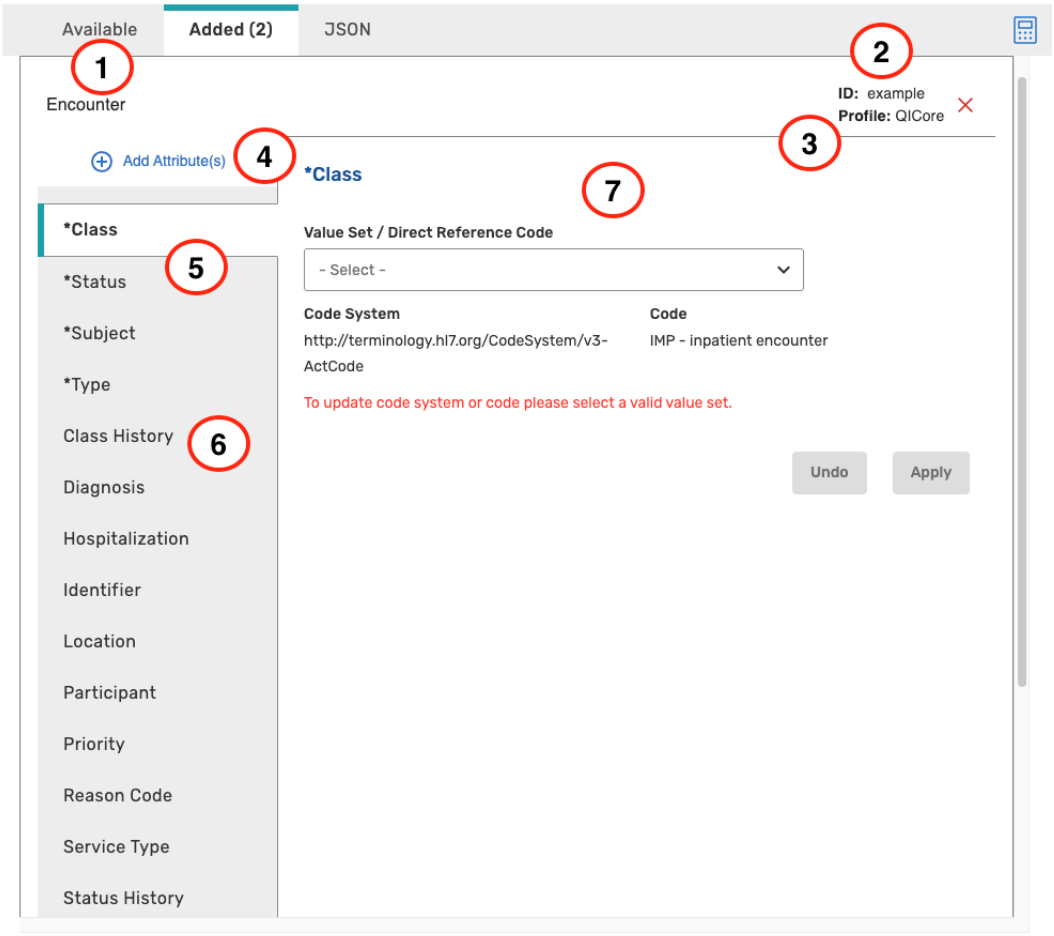

Image: Edit Profile Card

4.2.2.1 Required Attributes

In accordance with FHIR, some first-level attributes are required and are included automatically. Users must navigate to each required attribute tab to provide the necessary data. Again, data added here is not reflected in the JSON until the user selects Apply for that attribute. Also note, until these fields are filled in, it is expected you will receive validation errors because required attributes are missing from the profile instance.

4.2.2.2 Adding Optional Attributes

To add optional attributes, select Add Attribute(s).
The modal includes:

- An Attribute Selector showing all attributes currently added (displayed as chips).
- A scrollable list of all available attributes for the profile.

Users select additional attributes by checking their corresponding boxes. The selected attributes appear in the Attribute Selector for confirmation. Nested or extension attributes are displayed using their full hierarchical names.

After selecting attributes, users may select “Apply” which adds selected attributes to the Edit Profile Card or select “Discard Changes” which closes the modal without applying changes.
A close icon is also available in the upper right.

Like required attributes, users must navigate to each newly added attribute’s tab to enter its data.

4.2.2.3 Entering Attribute Data

Different attribute types (e.g., string, CodeableConcept, dateTime, Reference, etc.) require different inputs. Some attributes support multiple data types (choice types), allowing users to select the type that best fits their data.

Each attribute tab includes the necessary fields for its type. Users should enter data and then select “Apply”. These updates exist in the Builder UI until the test case is saved.

4.2.2.4 Reference Attributes

Reference attributes are more complex. Users will first choose the reference type, then select the profile instance to reference. The dropdown lists the IDs of applicable profiles already added to the test case, as well as an option labeled “ID Not Present (Add New)”.

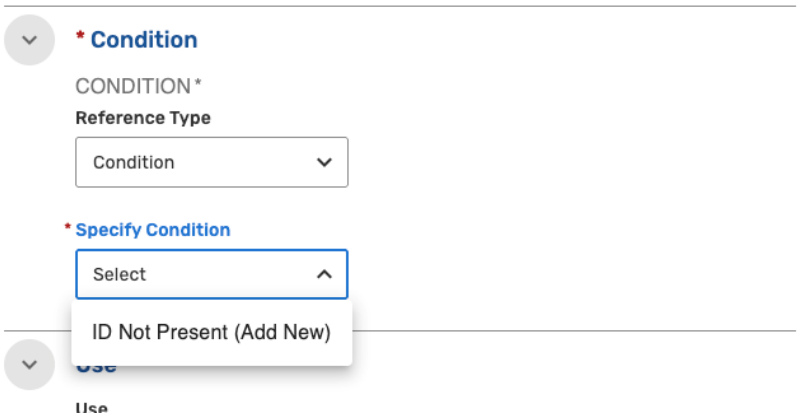

Image: Reference Type – ID Not Present (Add New)

If "Add New" is selected and multiple valid profiles exist based on that item (e.g., Condition), a prompt will appear asking the user to choose which profile type to create. After selection, a new profile is generated and added to the test case, and users must return to complete its required fields. Note: The profile types in the Choose Profile dialog are based on the narrowest implementation principle. Specific guidance is given in QI-Core STU 6 regarding references as well as US Core STU 6.1.0. Preference is given first to use of QI-Core profiles, then US Core profiles, when QI-Core profiles are not available, and finally core FHIR Resources, when neither QI-Core nor US Core profiles exist.

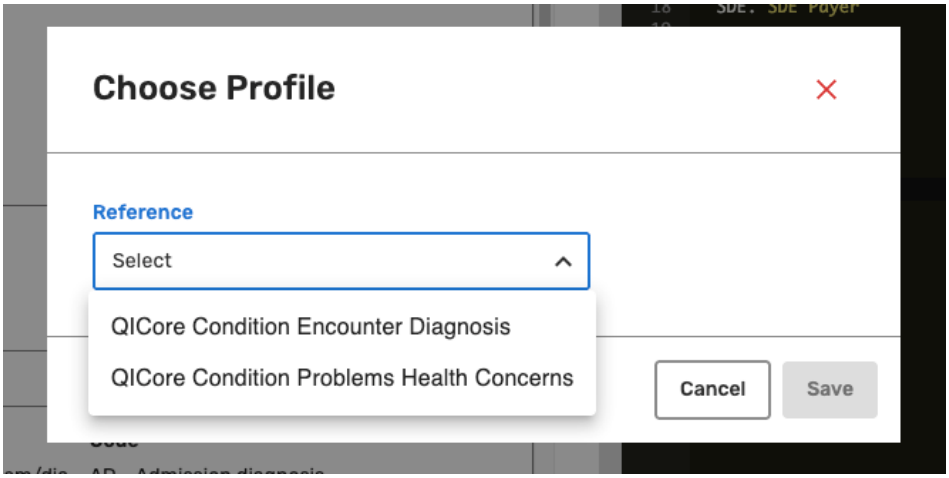

Image: Choose Profile Modal

4.2.2.5 Quantity Types

In the FHIR specification, Quantity and SimpleQuantity are closely related but serve different purposes. A Quantity is the more flexible and expressive type, and may include a value, comparator, unit text, system, code, and other optional fields. This makes it suitable for complex data elements that require richer detail or comparisons (such as <, ≤, or ≥).

A SimpleQuantity is a constrained version of Quantity. It includes the essential components needed to represent a measurement, typically the value, unit, system, and code, but does not allow comparators or some of the advanced extensions. Because it is streamlined and more predictable, SimpleQuantity is often used in profiles where a standardized format is preferred.

MADiE supports both Quantity and SimpleQuantity. However, when a QICore profile defines an element that may involve multiple quantity-related fields or allows more than one quantity type, the builder uses SimpleQuantity to provide a consistent and user-friendly interface. This simplifies data entry in the UI and ensures that the structure remains valid and aligned with the profile expectations.

When entering units in the Builder, users should provide the UCUM code, not a plain-text unit label. MADiE validates the entered unit to confirm it is a valid UCUM value, helping prevent errors and ensuring the test case conforms to FHIR expectations.

If users need to supply more detailed information than SimpleQuantity supports, or if their scenario requires the full flexibility of a Quantity, they can always switch to the JSON view to enter or edit the data directly. Any valid Quantity structure added through JSON will be preserved by MADiE, even if the Builder does not display every advanced field.

This approach allows MADiE to offer a streamlined Builder experience while still giving users full control through JSON when more complex modeling is required.

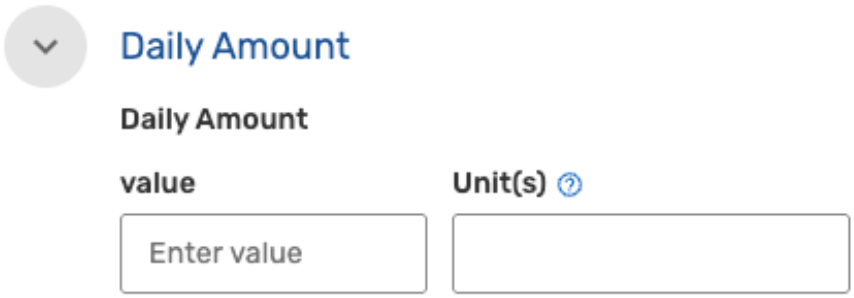

Image: Simple Quantity Input Fields

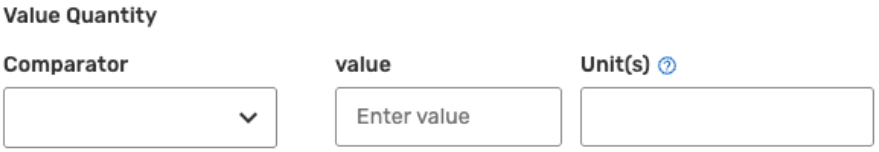

Image: Quantity Input Fields

4.2.2.6 Code Types

For attributes that use code data types, MADiE displays the code's human readable display value rather than the raw coded value itself. This approach makes it easier for users to understand what the selected code represents without needing to interpret terminology codes directly. For example, instead of showing the code IMP, MADiE displays the associated description of "Inpatient Encounter."

This does not change the underlying data. MADiE still stores and exports the full code, system, and display according to the FHIR specification. The Builder simply presents the display so users can confidently choose the correct concept.

If users need to inspect or modify the raw coded value, system URL, or other terminology details, they can switch to the JSON view, where the full code structure is visible and editable. This dual-view approach keeps the Builder intuitive while still giving users full control when they need it.

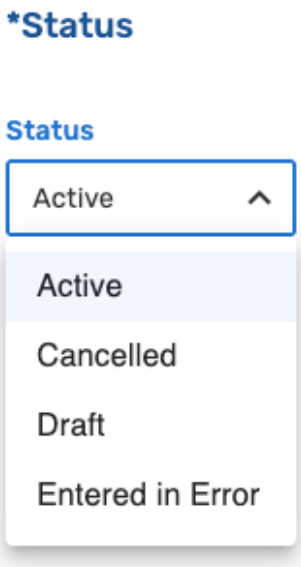

Image: Code Example – Claim.Status

4.2.2.7 CodeableConcept Types

CodeableConcept attributes represent coded clinical concepts drawn from value sets or terminology systems. In MADiE, these attributes are displayed in a structured format that helps users select, add, and manage codings according to the requirements of the underlying FHIR or QI-Core profile.

At the top of the CodeableConcept section, MADiE provides a field for selecting a Value Set or Direct Reference Code. When a profile specifies a required binding, the dropdown will display only the bound value set, and users must select a code from that specific value set. This ensures the selected coding aligns with the required terminology standards defined by the profile.

When the binding is not required, such as extensible, preferred, or example bindings, the value set associated with the attribute is still offered as an option, but it is not the only one. In these cases, the dropdown will also include any value sets referenced in the measure's CQL, as well as

custom codes. This gives users flexibility to choose the coding that best meets the needs of the test case while still supporting the preferred terminology guidance.

Below the value set/direct reference code selection, users can define the specific coding by choosing a Code System and Code.

If no applicable value set appears in the dropdown, or if the concept needed is not contained in any provided value set, MADiE offers a Custom Code option. Selecting this option provides free-text fields for both the Code System and the Code, allowing users to manually enter the terminology values needed for the scenario. This ensures users retain full flexibility, even in cases where a required value is not represented by any bound or available value set.

This structure allows MADiE to support both strict terminology requirements and flexible binding scenarios, helping users choose valid codes while still allowing room for measure-specific or custom terminology when appropriate.

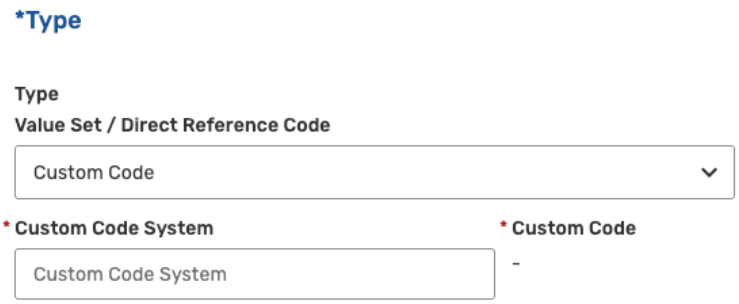

Image: CodeableConcept Custom Code

4.2.2.8 Unsupported Types

MADiE supports a wide range of QI-Core and FHIR attribute types within the Test Case Builder. However, there are a few attribute types that the Builder cannot interpret or render at this time. When an attribute's type is not supported, it will not appear as an editable option in the Builder interface. These attributes can still be included or maintained in the test case JSON, but they must be entered and updated manually in the JSON view.

If users encounter an attribute that does not display in the Builder or expected input fields are missing, they may safely continue working in the JSON view to supply the necessary data. If a user finds a needed QI-Core element that appears unsupported, we welcome that feedback to help prioritize future enhancements.

Unsupported attributes will never remove or alter data already present in the test case JSON; they simply cannot be edited visually. The Builder will preserve the existing JSON structure and will incorporate any valid data added manually once the JSON is saved and formatted correctly.

The list of unsupported types is as follows:

- Base64Binary
- Markdown
- Expression
- ParameterDefinition
- Annotation
- Attachment
- Contributor
- SampledData
- RelatedArtifact
- TriggerDefinition
- UsageContext
- Meta
- Address
- ContactPoint
- ContactDetail
- DataRequirement

4.2.2.9 Complex and Multi-Cardinality Attributes

Some attributes contain sub-attributes. These appear within expandable/collapsible headers, which are open by default but can be toggled for easier navigation.

MADiE supports multi-cardinality attributes.

- If an attribute supports multiple instances, an **Add** button appears next to it.
- Clicking **Add** creates another copy of the attribute fields.
- A **Delete** icon appears next to newly added instances, allowing users to remove them.
- For single (non-multi-cardinality) attributes, the delete icon clears the field but does not remove the attribute tab.

4.2.2.10 Deleting Attributes

To delete an added attribute:

1. Open its attribute tab.
2. Open the **Attribute Action Center** (upper-right corner of the attribute section).
3. Select the **Delete** icon.

If the Delete icon does not appear, the attribute is required and cannot be removed, though users can update its value.

4.3 JSON Tab

The JSON Editor is where the JSON logic of a test case can be viewed and edited. All information can be added, edited, deleted, and must be in correct JSON syntax. When saving, any errors in the JSON will be displayed. JSON files may contain errors and warnings. JSON containing errors can be saved but may prevent dependent functionality from working properly (e.g., test case calculation). JSON files containing just warnings can be saved and calculated. Verify the JSON is complete, and errors are resolved to ensure proper functionality of areas dependent on it.

MADiE will validate the PatientID used for the patient throughout the test case JSON. When the test case JSON is saved MADiE will automatically add a PatientID for the patient if it was not included and will overwrite the PatientID if the PatientID used was inconsistent. This will be in UUID format. At this time, only the PatientID will be overwritten or added. No reference to the patient will be updated with the new PatientID.

When editing dates with timezones in MADiE, MADiE will force the timezone offset to be zero when users click “Save.” This will ensure consistency across all timezones and allow users to get consistent results regardless of the timezone the user is in when running test cases.

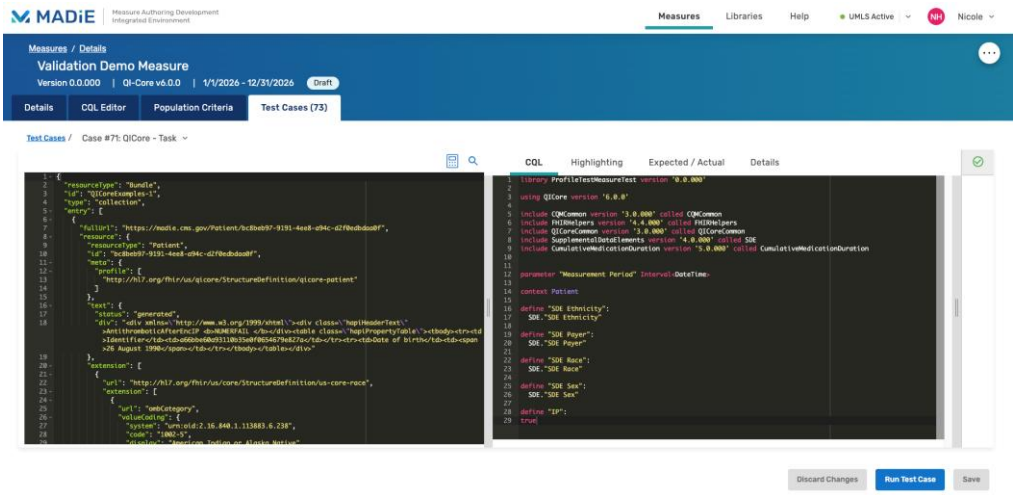

Image: QI-Core Test Case Editing

The JSON Editor allows users to find specific phrases in the JSON. User can perform a search by clicking the search icon above the Editor or, with your curser in the Editor, press Ctrl+F or Cmd+F on the keyboard to open the Find & Replace window. Users can type a string in the Find text area and the editor will find all locations of the test. Use the arrow keys to navigate forward and backward between the elements. Clicking “All” will highlight all instances of the text.

If users with edit access wish to replace text, use the find function to find the text to replace. Then type in the replacement text in the "Replace with" text area. Clicking "Replace" will replace the current highlighted text and highlight the next instance of the word. Selecting "All" will replace every instance of the find text in the JSON. If the replace field is not shown in the pop-up, simply click the Plus button below the find text area and it will appear.

4.4 CQL Tab (View-only)

The right panel initially displays the measure CQL to aid in building the test case JSON. However, the measure CQL is view-only and cannot be edited in this area. CQL edits can only be made in the CQL Editor tab.

4.5 Highlighting Tab

The Highlighting subarea (second tab from the left in the right panel, next to the CQL tab) displays a test case's highlighting results calculated against the measure CQL. The highlighting subarea is view-only and cannot be edited. No highlighting will display until after clicking the "Run Test" button. Green and/or red highlighting is generated. Green highlighting indicates a passing result for any applicable lines of measure CQL, while red highlighting indicates a failing result for any applicable lines of measure CQL. Highlighting is broken down and displayed on separate tabs, one each for:

- Measure Populations (a list of the populations applicable to the measure)
- SDE (Only appears if the user has specified, they want to include SDEs in the test case configuration)
- Definitions
- Functions
- Unused

Users can click through the tabs to get targeted highlighting information. If users would like to look at highlighting for a different Population Criteria, they will click the dropdown at the top of the section. It will open a list of Population Criteria, and users can select the population criteria they want to view. The "Run Test" button must be clicked each time a user would like to calculate new results.

The highlighting tab also contains a Results section for each definition. The Results section can be found directly below each definition and can be expanded and collapsed. It displays the results of the logic evaluated on the test case.

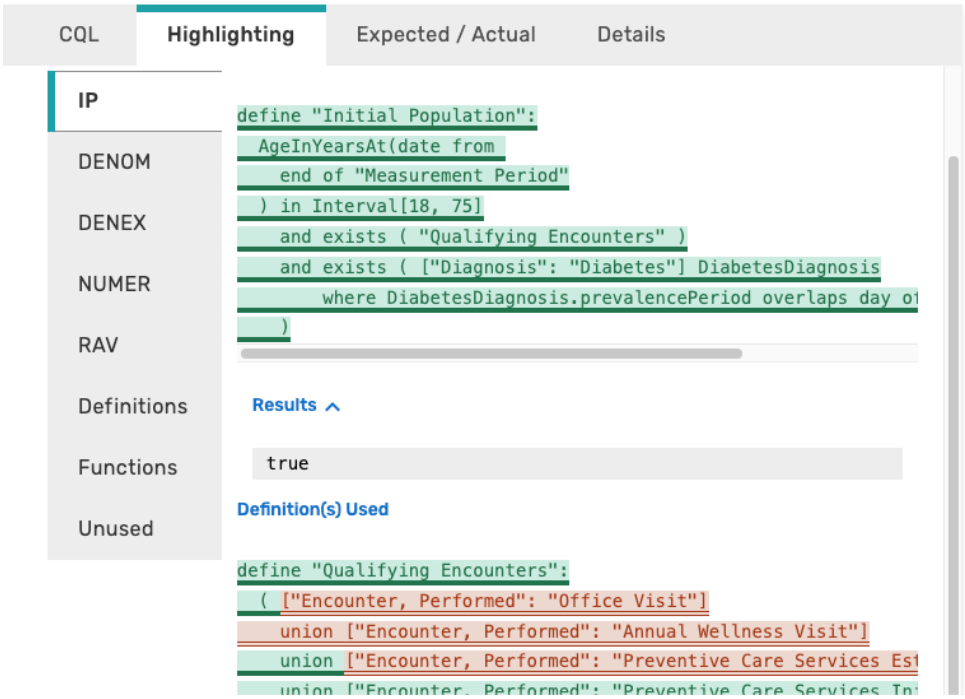

Image: Test Case Pass/Fail Highlighting

4.6 Expected/Actual Tab

The Expected/Actual subarea (third tab from the left, next to the Highlighting tab, in the right panel) displays population criteria tables, where expected values for populations, stratifications, and observations within each set of population criteria can be configured. Upon clicking the "Run Test" button, Actual values are displayed along with pass/fail results for the test case across each set of population criteria. Updating the expected values will clear out any actual values that are displayed. Then the "Run Test" button must be clicked each time a user would like to calculate new results.

4.7 Details Tab

The Details subarea (last tab in the right panel, next to the Expected/Actual tab) displays:

1. **Title:** The Title field is the current name of a test case.
2. **Description:** The Description field communicates what a test case is testing and is auto populated with any Description field content entered during the creation of the test case. **Note:** Description can only contain 250 characters.
3. **Test Case Group:** The Test Case Group field assigns a test case to a group that can be used to sort and organize the test case table according to a user's preferences. It is auto populated with any Test Case Group assignments entered during the creation of the test case.

5 Test Case Validation

5.1 QI-Core v4.1.1

QI-Core v 4.1.1 test case validations will display in the Validations tab. Error and warning messages are displayed in this section. The section is expandable or collapsible by clicking on the text “Validations.” To increase the size of your working area, select the divider and drag it to the left to make the section bigger or smaller.

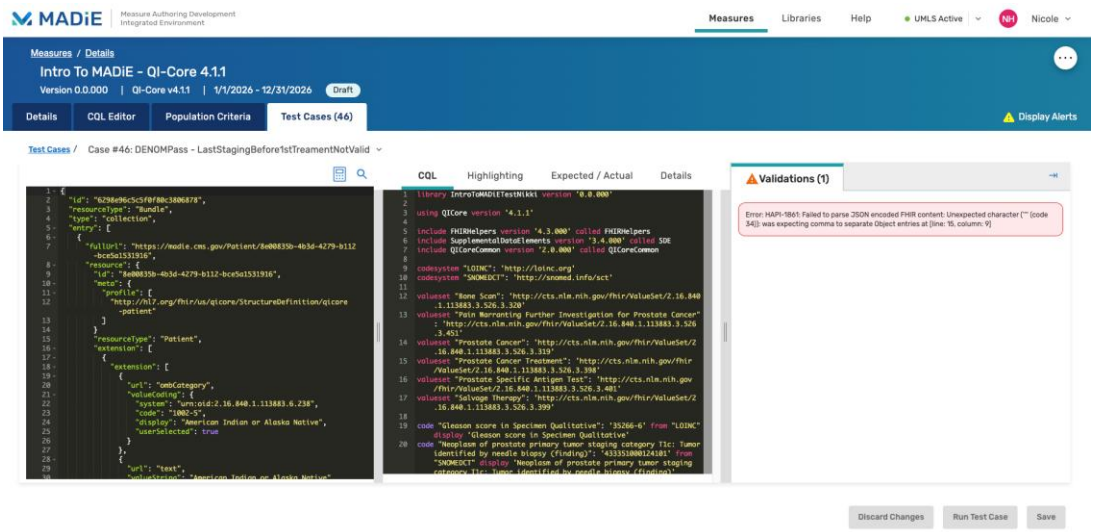

Image: QI-Core v4.1.1 Validations

5.2 QI-Core v6.0.0

In MADiE, validation for QI-Core STU6 test cases runs automatically and asynchronously after a test case is saved. Once saved, the test case enters a validation queue, and its status is set to Pending. After a short delay, the status will update to either Valid or Invalid, depending on the outcome. Because validation happens in the background, users can continue working, whether in the same test case, a different one, or even after logging out. The results will be available when the user returns.

A test case is considered valid if the JSON is properly formatted, all required data elements are present, codes align with the appropriate value set bindings, and no validation errors are returned. Any error will mark the test case as invalid. Warnings or informational messages do not affect validity.

Validation status is visible on the test case list page, but to view detailed messages, users must open the test case for editing. The validation panel on the right-hand side displays the current status icon. Clicking the icon expands the panel to show messages from the validation service with errors appearing first, followed by warnings, then informational notes.

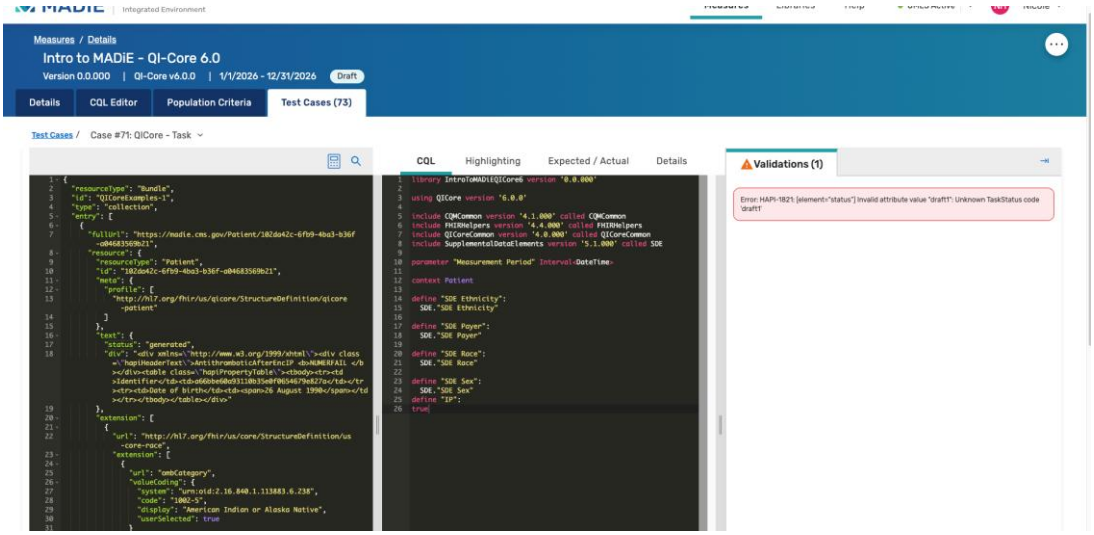

Image: QI-Core STU6 Invalid Test Case Example

MADiE uses value set expansions via VSAC or an in-memory validation expansion bundle that was created specifically for MADiE. However, some value sets may not expand successfully due to size, external hosting, or structure. In these cases, MADiE cannot confirm whether a code is part of the intended value set, and a warning will be shown, even if the code is technically correct. These warnings will not be able to be resolved by the user, but they are informational and do not block a test case being valid or measure release.

Note: Validation does not test the logic or expected outcomes of the measure. That is handled through the Run Test Case feature and the pass/fail percentage. Please refer to section 6 for information on how to execute test cases regardless of validation status.

6 Test Case Execution

Test case execution in MADiE allows measure developers to validate the behavior of measures by running structured patient scenarios against the measure logic. MADiE supports both authoring and testing within a single integrated environment, replacing the previous workflow that required separate tools. This combined approach streamlines the process of verifying that a measure accurately reflects its intended clinical purpose.

In MADiE, test cases operate as simulated patient profiles that include demographic characteristics, clinical conditions, services, and timing details. Measure Developers construct these scenarios based on the expected logic pathways defined in the measure's CQL, allowing them to predetermine the expected outcome for each case. When the test is executed, MADiE evaluates the measure logic against the constructed scenario and returns the calculated result, enabling comparison between expected and actual outcomes. A matching result confirms that the

measure logic is functioning as intended, while discrepancies indicate areas where further review or adjustment may be needed.

To execute test cases users can click the “Run Test(s)” button on the test case list page or the “Run Test” button on the edit test case page. **Note:** In QI-Core test cases, if a reference does not resolve to a valid profile within the current test case, the resource containing that reference will be excluded during execution. This behavior does not modify the test case itself; it simply ensures that unresolved references are not considered when the test logic runs. When users save a test case with this scenario, a warning message will be displayed to inform them that unresolved references were detected.

7 Import Test Cases

MADiE allows users to import one to multiple test cases for a measure that were previously exported from MADiE; see section 8 to learn how to export test cases from MADiE. Follow these steps to import multiple test cases for a measure:

1. Log in to MADiE and open the measure you wish to import the test cases to.
2. Navigate to the Test Case(s) tab.
3. Click the “Import” button (see image below).
4. A Test Case Import screen will appear. Click the “Select Files” button.
5. MADiE will prompt you to select a file using the browser’s default selector.
6. Select and open the .ZIP file containing the test cases exported from MADiE. The structure of the file must be retained from the MADiE export (i.e., the .ZIP file shall contain one folder for each test case whose name is the PatientID for that test case, and each folder shall contain one JSON test case file. A .madie file containing metadata on each test case will also be present).
7. Once you have selected a valid test case .ZIP file, click the “Import” button.
8. MADiE will replace the existing test case JSON with the imported JSON for all test cases whose PatientIDs match. Measure report information such as expected and actual values will not be updated during the import process for existing test cases, users will need to manually update those after the import is complete. Test cases that do not have a matching PatientID in the current measure will be imported as new test cases on the measure, including expected values. **Note:** The combination of Title and Group must be unique for the measure. Test cases that fail to import due to the Title and Group not being unique to the measure will be listed after the import processing is complete.

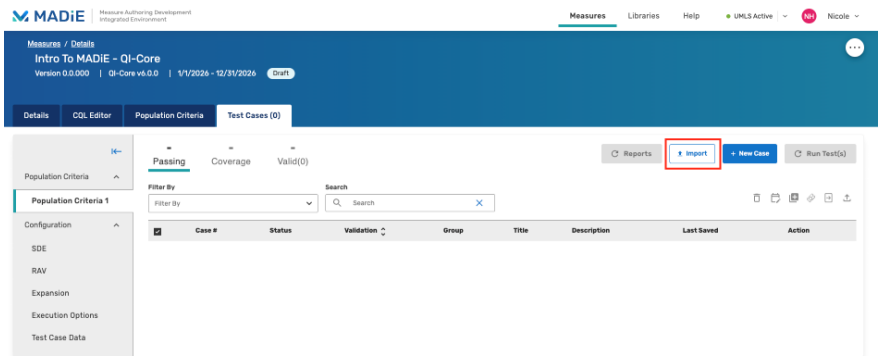

Image: QI-Core Test Case Tab – Import Test Cases from MADiE Button

Should the measure populations not match, expected values are not copied over, and a warning message will be displayed. Users can select the "Copy Test" icon, see screenshot below. Doing so will copy the warning and the test case IDs allowing users to copy it into a word document. This will allow users to see the test cases more easily they need to update to import expected values.

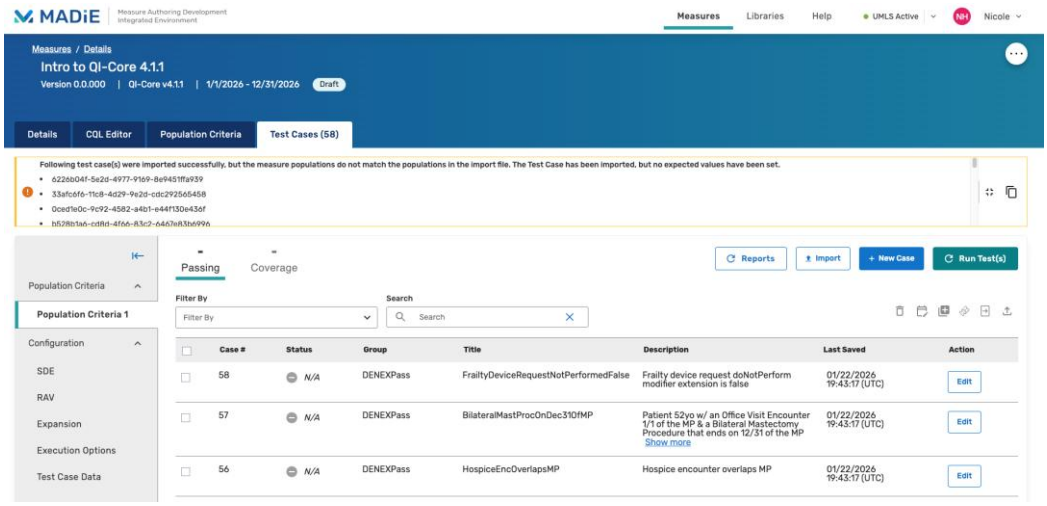

Image: QI-Core 4.1.1 Test Case Import Warning Message

8 Export Test Cases

MADiE allows users to export QI-Core test cases as either transaction bundles or collection bundles. The test cases will be exported in a JSON format with information about the measure and expected values added into a Measure Report Resource in the JSON file.

8.1 Export Test Case(s)

To export Test Case(s), navigate to the test Case List Page. First, select the test case(s) you want to export; you can select one to all. Next, click the Export icon and select "Transaction Bundle" to export the test case(s) as a transaction bundle, or "Collection Bundle" to export the test case(s) as a collection bundle. A .zip file will be created with the PatientID for the test case(s) selected

for export included in a folder name. The folder will contain the JSON file(s) for the test case(s). The .zip file will also include a read me file linking the Case Number and PatientID to the Test Case Title and Test Case Group. In the read me file will also be the Case Number, test case ID, measure Group, and measure title of any test case that could not be included in the export because it was invalid. The image below shows the correct order of required steps to follow to export test case(s).

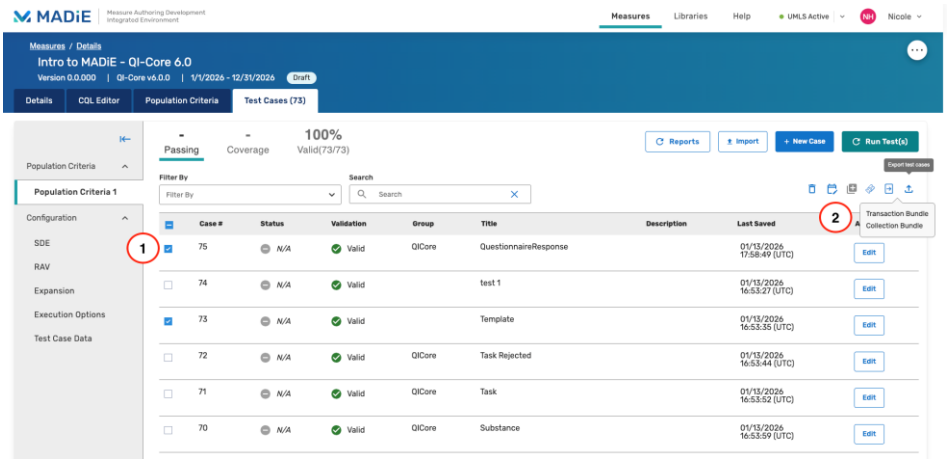

Image: Export QI-Core Test Case(s)

8.2 Export All Test Cases

To export all Test Cases associated with a measure navigate to the Test Case List Page and find the “Items per page” dropdown below the test case table. Select “All” from the dropdown. Next, scroll to the top of the test case table and click the select all checkbox. All test cases will now be selected. Finally, click the Export icon and select “Transaction Bundle” to export the test cases as a transaction bundle, or “Collection Bundle” to export the test cases as a collection bundle. A .zip file will be created with a folder for each test case which includes the PatientID in the folder name. The folder will contain a JSON file for each of the test cases. The .zip file will also include a read me file linking the PatientID to the Test Case Title and Test Case Group.

Image: Pagination – All Selected

9 Synchronize Test Case JSON with Test Case UI

The Synchronize Test Case JSON with Test Case UI feature is available only for QI-Core measures where the user has edit access. This feature updates the patient's name fields in the JSON to match the values the user has entered in the Test Case UI. Specifically, the title will be used to set the given name field, and group will be used to set the family name field.

To use the feature, select one or more test cases in the Test Case List and click the puzzle piece icon in the action center. A confirmation modal will appear, indicating how many test cases were selected and explaining that all family fields in the JSON will be set to the group value entered in the UI, and all given fields will be set to the title value. If you wish to proceed, click "Yes, Make JSON Match UI."

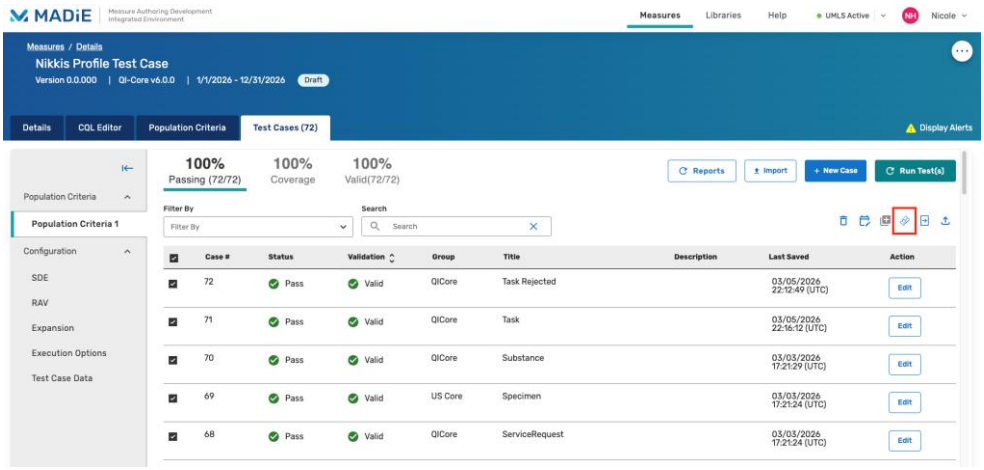

Image: Make JSON Match UI Icon

During processing, any test case that is locked by another user, contains a JSON syntax error, or lacks patient family or given fields will be skipped because it cannot be updated. After the operation completes, a success message will appear if all selected test cases were updated successfully. If any test cases could not be updated, an error message will display a list of the affected test cases by group and title so you can review and update their JSON manually.

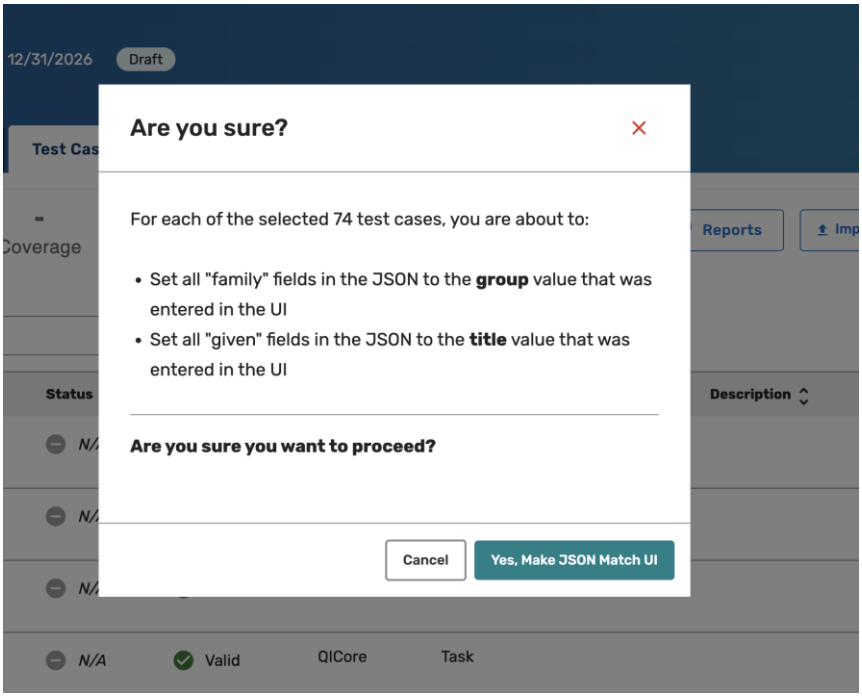

Image: Synchronize JSON and UI – Are You Sure Modal

10 Feedback and Support

An issue tracker and feedback email list are available to support the resolution of issues and to answer questions related to the MADiE application. The MADiE issue tracker is available on the ONC Jira System.

MADiE users should create a ticket in the issue tracker to report bugs, ask questions, or to request new features. To add an issue, users must create a login account in the Jira system. Once an issue has been entered, the MADiE team will review and prioritize it.

CMS Web Application Firewall (WAF) Errors can occur in MADiE. WAF errors occur when the attempted action (i.e., saving or exporting) is rejected by the CMS Security policy, blocking further action by the user. This can happen when doing any work within MADiE. If a WAF error is encountered, a pop-up message will display notifying the user they have encountered a WAF error, and they should contact the help desk. A support ID will be included in the pop-up. When a Support ID is provided in the error message, users should include that in the support ticket to help facilitate resolution by CMS.

Acronyms

Acronym	Definition
CMS	Centers for Medicare & Medicaid Services
CQL	Clinical Quality Language
FHIR	Fast Healthcare Interoperability Resources
JSON	JavaScript Object Notation
MADiE	Measure Authoring Development Integrated Environment
QI-Core	Quality Improvement Core
QDM	Quality Data Model
UCUM	Unified Code for Units of Measure
UI	User Interface
VSAC	Value Set Authority Center