C2C Message Validator Tool User Guide

Lesson Goal

This guide will walk you through how to use the Center-to-Center Message Validator Tool (C2C MVT) for the C2CRI (Center-to-Center Reference Implementation) projects. The purpose of this tool is to validate center-to-center messages against a standard or a schema, to ensure that they have the correct data format and conform to a standard. The initial standard being implemented is the next generation TMDD.

Benefits of the Tool

- **Enhanced Error Messages:** It provides a deeper understanding of the JSON schema, allowing for more detailed and precise error messages compared to other tools.
- **Future-Proof Flexibility:** Built with expandability in mind, the tool can implement other standards using the same interfaces. This flexibility enables validation of messages across different standards, making it easier for users to identify required modifications and resolve validation errors.

Topics

This guide will walk you through:

- Differences between ngTMDD Standard Schema and Tool Schema
- Configuring a message.
- Validating a message.
- Validating multiple messages.
- Failed message examples.

Installation

Follow the instructions in the installation guide to install the tool on your local machine.

Differences between ngTMDD Standard Schema and Message Validator Tool Schema

The current version (User Comment Draft) of the ngTMDD Standard's JSON Schema is a collection of individual message definitions and is not intended to define a unified root structure for API payloads, however there are plans to fix this in the next iteration. Therefore, the Message Validator Tool currently uses a modified version of the JSON Schema that defines a message payload and message type property for each message defined in the JSON Schema. To use the Message Validator Tool, users will have to modify their JSON messages to include the "message" and "messageType" properties. The "message" property must be contained in a root JSON object. The "messageType"

property must be contained in the object that is the value of the “message” property. Here is an example showing this structure with an “ActivityRequestLog” message:

Original JSON:

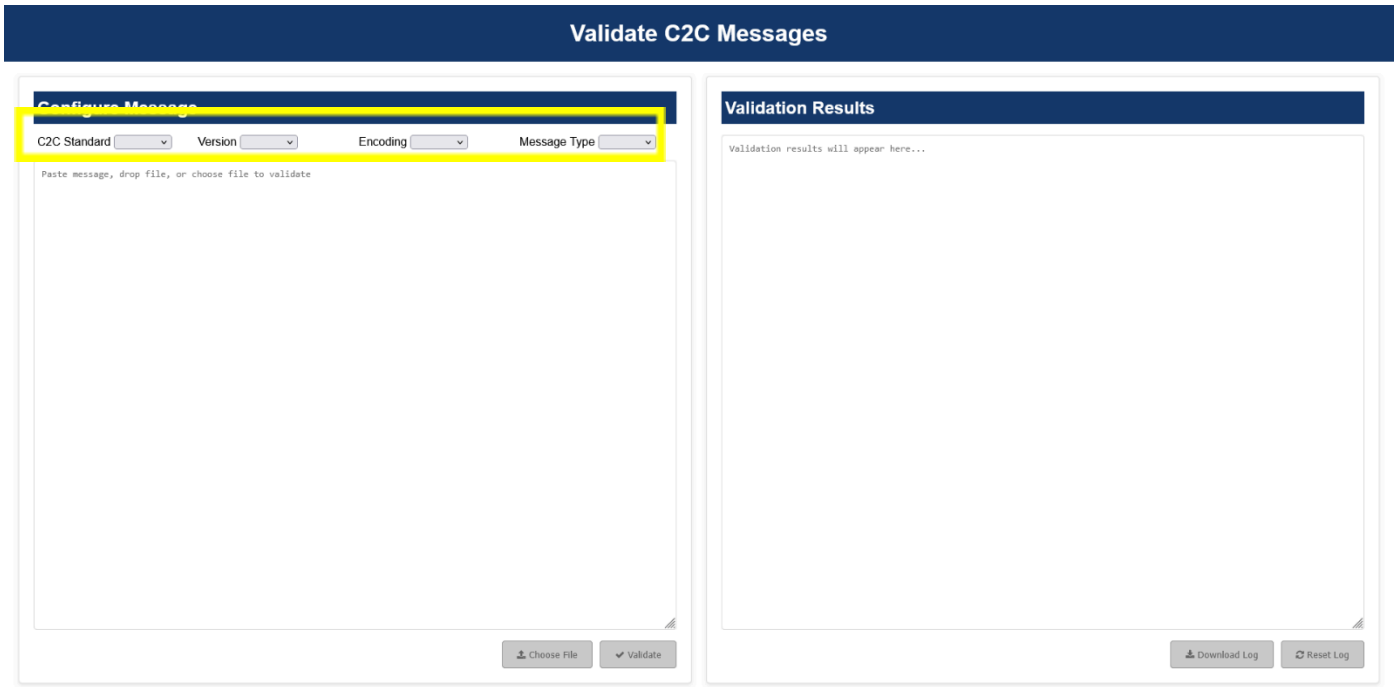
```
{
	"ownerOrganizationId": "org_id",
	"externalOrganizationId": "ext_org_id",
	"requestId": "request_id"
}
```

Modified JSON:

```
{
	"message":
	{
		"messageType": "ActivityLogRequest",
		"ownerOrganizationId": "org_id",
		"externalOrganizationId": "ext_org_id",
		"requestId": "request_id"
	}
}
```

Configure a Message

1. Navigate to the Message Validator Tool and open the user interface.
2. The interface is divided into sections: Configure Message and Validate Results. To configure a message, you must select the:
    a. C2C standard
    b. Version of the standard
    c. Data encoding
    d. Message Type



3. Click on the **C2C Standard** drop down menu and select **ngTMDD**.
4. Click on the **Version** drop down menu and select **1.0**.
5. Click on the **Encoding** drop down menu and select **UTF-8**.
6. Click on **Message Type** and select **Auto Detect**. A list of message types is not provided because the C2C standard automatically detects the message type.

Note: Make your selections from left to right, as the chosen standard determines the available versions, encodings, and message types. You will be able to choose a file and validate the message configuration when all options have been set.

7. After completing the configuration, you can validate a message by **pasting it**, **dropping a file**, or **choosing a file**. The Validate button will remain disabled until a message is entered into the text box.
8. To choose a file, click **Choose File**, select the file, and click **Open**.

9. After the message uploads, click **Validate**. The message will automatically validate, and the **Validation Results** will populate on the right side of the interface.

Note: The same steps apply whether you drop a file or paste a message in the Configure Message text box.

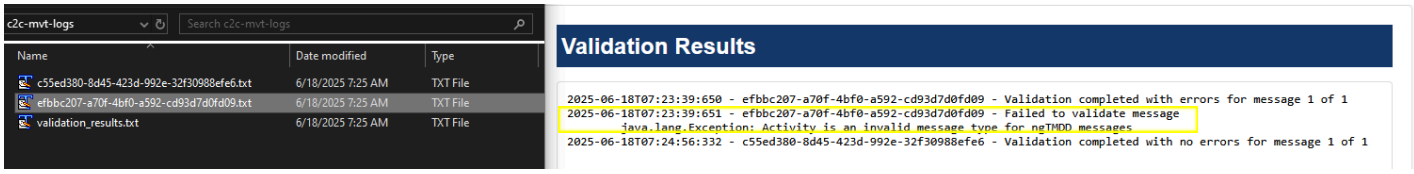
Validation Results

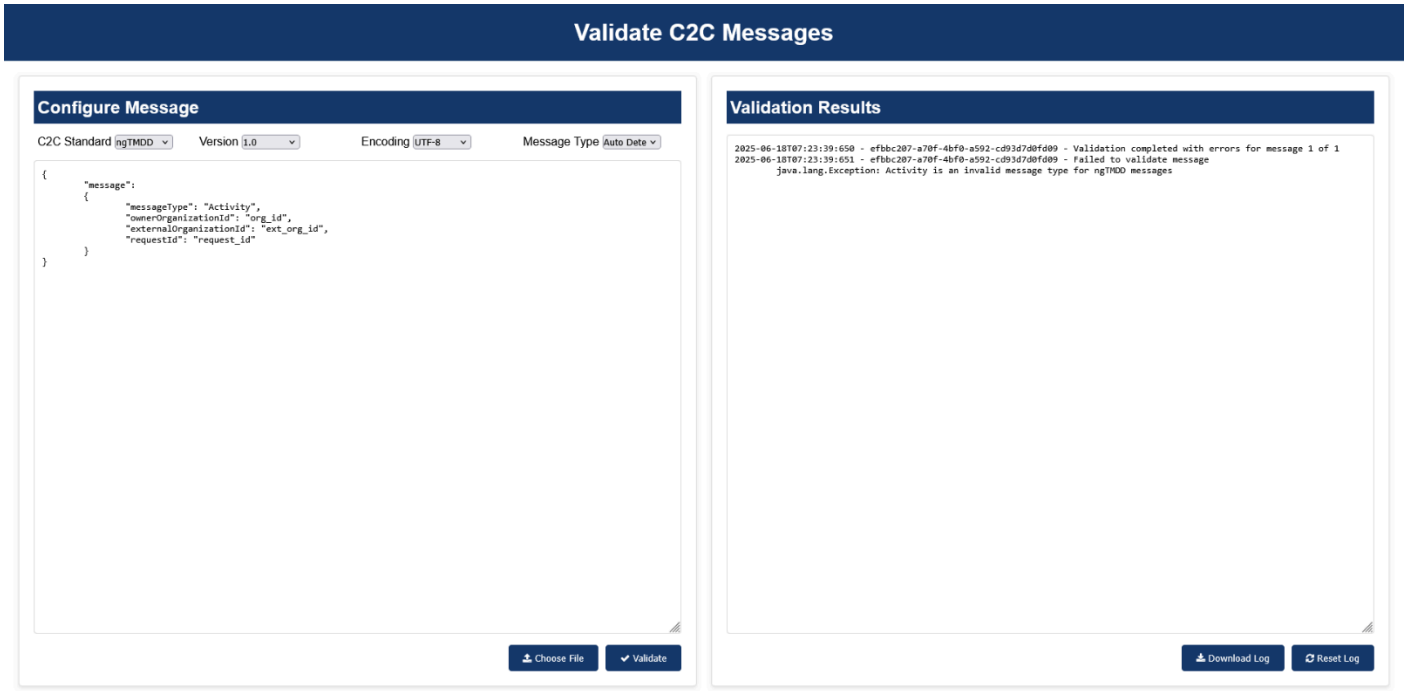
- **No Errors:** The validation completed without errors because the message is valid. If there were any issues, the validation results would indicate errors, and all detected errors would be listed in the text box.
- **UUID:** Each validation result includes a timestamp and an UUID. In this example, the UUID is assigned when the message is submitted. This is helpful if an error occurs, as the UUID allows you to identify and trace the specific message that caused the error when reviewing the downloaded zipped log files.

Download Log

The download logs are stored on your server and allow you to retrieve the messages you validated, including those displayed in the text box and any messages you sent, so you can review the results of your tests.

1. Click **Download Log**. A file explorer prompt will appear with a zip file name automatically generated. You can rename the file before saving it, if desired.
2. Click **Save**.
3. Locate the zip file, then **unzip** and extract its contents to create a new folder.
4. Inside the folder, you'll find a text file containing all the validation results, including the messages that were sent along with their associated UUIDs. This file allows you to review all submitted messages, including those that failed validation.
5. Open the file to examine any errors. In this example, the error occurred because the message type was set to "Activity", which is invalid. It should have been set to "ActivityLogRequest", which is the correct value. You can correct the message and revalidate it to see if the issue is resolved.





Validation Results Text File

The validation results text file includes all submitted messages, ensuring nothing is lost if your session ends unexpectedly. This file is included in the log bundle and stored on the server or wherever the application is running. Even if you refresh the page or accidentally close your browser, validation results remain available.

Validate Multiple Messages

Multiple messages can be validated using any of the three methods to submit messages by separating the messages by commas.

1. After submitting a message by dropping a file, choosing a file, or entering a message into the text and clicking **Validate**, the results will appear in the Validation Results section.
2. In the example shown below, two messages were entered into the text box.
3. The first message was valid and processed without errors.
4. The second message contained an invalid message type and was returned with an error.
5. The results indicate: "Message 1 of 2 completed with no errors" and "Message 2 of 2 completed with errors."

Reset Log

To avoid receiving duplicate messages in future downloads, it's recommended to reset the log after downloading it. However, this is optional and can be done at the user's discretion.

Failed Message Examples

The following examples show how to handle a failed message and explain the reasons for the failure.

Single Error

- For example, suppose you copy and paste a message into the Configure Message text box, but it contains a typo, such as a misspelled key or enumeration value. When you attempt to validate the message, an error will occur.
- The validation will complete with errors, and the details explaining the issue will appear in the text box.
- In this example, the error occurred because the system couldn't identify the message type due to the typo.

Multiple Errors

- For example, if you omit a required property and use an invalid value for enumeration, the validation will complete with multiple errors. In this case, the word "suppressed" is misspelled and the required property “cctvId” is missing.
- The Message Validator Tool clearly identifies these specific issues. In contrast, generic validation tools often return long, difficult-to-interpret error lists, making it harder to pinpoint the exact problems.

Validate C2C Messages

Configure Message

C2C Standard ngTMDD | Version 1.0 | Encoding UTF-8 | Message Type Auto Dete

```
{
        "message":
        {
                "messageType": "CCTVImageLinkRequest",
                "deviceInformationRequest":
                {
                        "ownerOrganization":
                        {
                                "organizationId": "myorg"
                        },
                        "deviceType": "cctv camera",
                        "deviceInformationType": "image link"
                },
                "imageType": "supresed stream"
        }
}
```

Choose File | Validate

Validation Results

```
2025-06-18T07:51:58:136 - 6c55f33d-b0c3-478d-a1e0-6821e91413c8 - Validation completed with errors for message 1 of 1
2025-06-18T07:51:58:136 - 6c55f33d-b0c3-478d-a1e0-6821e91413c8 - Failed to validate message
        java.lang.Exception: multiple validation failures
                RequiredValidationFailure: required properties are missing: cctvId
                EnumValidationFailure: the "supresed stream" is not equal to any enum values
```

Download Log | Reset Log