

Article

[Muhammad Waseem](#) · Jul 26, 2021 2m read

Creating and Validating Any HL7 FHIR Resource by using FHIR schema with the help of IntelliSense and auto complete functionality in VS Code

Healthcare interoperability is instrumental in improving patient care, decreasing healthcare provider costs, and providing a more accurate picture to providers. However, with so many different systems, data is formatted in many different ways. There are many standards that have been created to try to solve this problem, including HL7v2, HL7v3, and CDA but each one has its drawbacks.

FHIR, or Fast Healthcare Interoperability Resources, is a new format for healthcare data that aims at solving these problems. It is developed by Health Level Seven International (HL7), an organization that also developed HL7v2, HL7v3 and CDA.

Today we will explore how to Create and Validate FHIR Resource by using FHIR schema with the help of IntelliSense and auto complete functionality in VS Code.

Step 1 : Download JSON schema file for Resource validation from FHIR official website <https://www.hl7.org/fhir/>

hl7.org/fhir/documentation.html

HL7 FHIR Release 4

Home Getting Started Documentation Resources Profiles Extensions Operations Terminologies

Table of Contents > Documentation Index

This page is part of the FHIR Specification (v4.0.1: R4 - Mixed Normative and STU). This is the current published version. For a full list of available versions, see the [Directory of published versions](#)

1.1 Documentation Index

FHIR Infrastructure Work Group	Maturity Level: N/A	Standards Status: Informative
--	---------------------	-------------------------------

This page provides an index to the key commonly used documentation pages for FHIR.

- Framework**
 - Conformance Rules [N](#)
 - Resource Life Cycles
 - References between Resources [N](#)
 - Compartments
 - Narrative [N](#)
 - Extensibility [N](#)
 - Formats: [N](#) XML [N](#), [N](#) JSON [N](#), & [N](#) RDF
 - Terminologies [N](#) (Code Systems, Value Sets)
 - FHIRPath [N](#)
 - Mappings to other standards
- Version Management**
 - Change Management & Versioning [N](#)
 - Managing Multiple FHIR Versions
 - Version History
 - Differences to Release 3
- Exchanging Resources**
 - RESTful API (HTTP) [N](#)
 - Search [N](#) (Search Param Registry)
 - Operations [N](#)
 - Asynchronous Use
 - Using GraphQL
 - Documents
 - Messaging
 - Services
 - Persistence/Data bases
- Base Types**
 - Data Types (Base) [N](#)
 - Metadata Types [N](#)
 - Resource [N](#)
 - DomainResource [N](#)
 - Element [N](#)
- Adopting & Using FHIR**
 - Profiling FHIR [N](#)
 - FHIR Workflow
 - [Downloads - Schemas, Code, Tools](#)
 - Managing Multiple FHIR Versions
 - Validating Resources
 - Best Practices for Implementers
 - Mapping Language (tutorial)
 - Testing Implementations
- Safety & Security**
 - Security, Security Labels & Signatures
 - Clinical Safety
- Implementation Advice**
 - Managing Resource Identity
 - Guide to Resources

hl7.org/fhir/downloads.html

This page is part of the FHIR Specification (v4.0.1: R4 - Mixed Normative and STU). This is the current published version. For a full list of available versions, see the [Directory of published versions](#)

7.1 Downloads

FHIR Infrastructure [Work Group](#) Maturity Level: N/A Standards Status: Informative

Specification Downloads

FHIR Definitions All the value sets, profiles, etc. defined as part of the FHIR specification, and the included implementation guides:

- XML
- JSON
- Forge: Special version of definitions for [Forge](#) (temporary)

This is the master set of definitions that should be the first choice whenever generating any implementation artifacts. All the other forms below include only subsets of the information available in these definition files, and do not contain all of the rules about what makes resources valid. Implementers will still need to be familiar with the content of the specification and with any profiles that apply to the resources in order to make a conformant implementation.

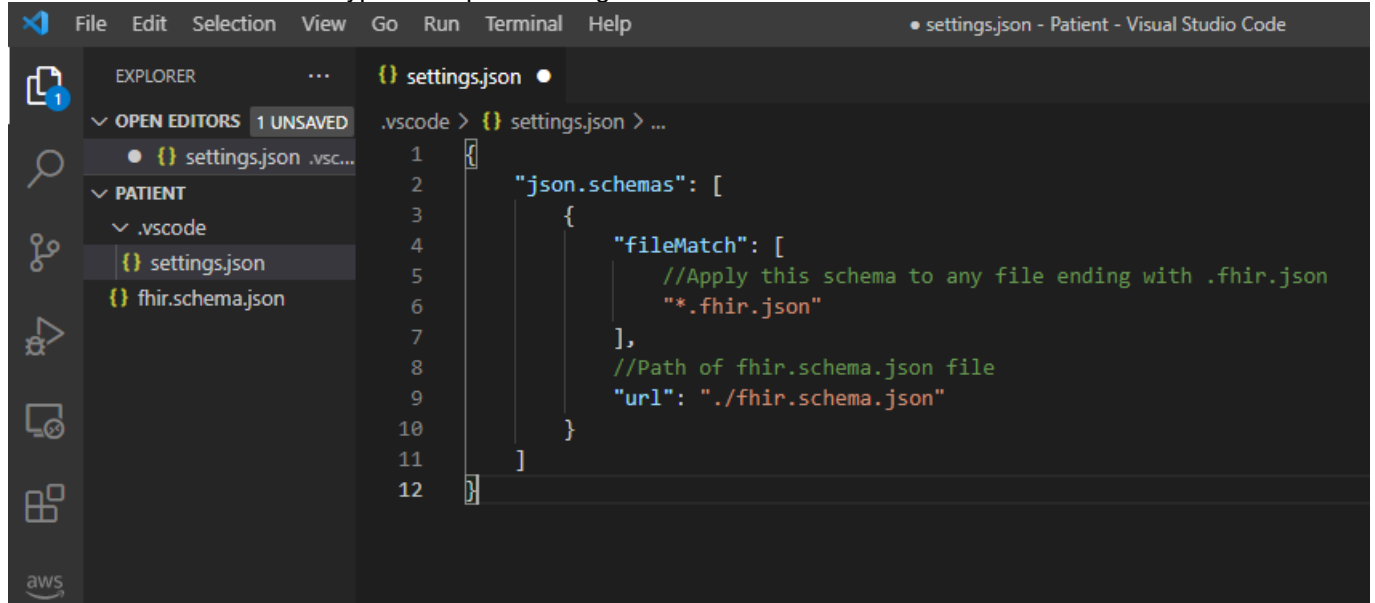
- XML
- [Examples](#) - all the example resources in XML format
 - [Validation Schemas](#) (includes support schemas, resource schemas, modular & combined schemas, and Schematrons)
 - [Code Generation Schemas](#) (see [notes about code-generation schemas](#))
Note that names relevant for code generation, including resource names, element & slice names, codes, etc. may collide with reserved words in the relevant target language, and code generators will need to handle this
- JSON
- [Examples](#) - all the example resources in JSON format
 - [JSON Schema \(Needs JSON Schema draft-06 or more recent\)](#)
 - [Examples](#) - all the example resources in ND-JSON format (bulk data format)
- RDF
- [Turtle Examples](#) - all the example resources in Turtle format
 - [ShEx Schemas - ShEx](#) definitions for validating RDF resources
 - [Definitions](#) - the formal definitions that define the predicates and classes used in the RDF format (not up to date)

Step 2: Create folder (in this example I am using Patient folder and Patient Resource) and copy extracted fhir.schema.json file to same folder, then open folder from VS Code

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "id": "http://hl7.org/fhir/json-schema/4.0",
  "description": "see http://hl7.org/fhir/json.html#schema for information about the FHIR Json Schemas",
  "discriminator": {
    "propertyName": "resourceType",
    "mapping": {
      "Account": "#/definitions/Account",
      "ActivityDefinition": "#/definitions/ActivityDefinition",
      "AdverseEvent": "#/definitions/AdverseEvent",
      "AllergyIntolerance": "#/definitions/AllergyIntolerance",
      "Appointment": "#/definitions/Appointment",
      "AppointmentResponse": "#/definitions/AppointmentResponse",
      "AuditEvent": "#/definitions/AuditEvent",
      "Basic": "#/definitions/Basic",
      "Binary": "#/definitions/Binary",
      "BiologicallyDerivedProduct": "#/definitions/BiologicallyDerivedProduct",
      "BodyStructure": "#/definitions/BodyStructure",
      "Bundle": "#/definitions/Bundle",
      "CapabilityStatement": "#/definitions/CapabilityStatement",
      "CarePlan": "#/definitions/CarePlan",
      "CareTeam": "#/definitions/CareTeam",
      "CatalogEntry": "#/definitions/CatalogEntry",
      "ChargeItem": "#/definitions/ChargeItem",
      "ChargeItemDefinition": "#/definitions/ChargeItemDefinition",
      "Claim": "#/definitions/Claim",
      "ClaimResponse": "#/definitions/ClaimResponse",
      "ClinicalImpression": "#/definitions/ClinicalImpression",
      "CodeSystem": "#/definitions/CodeSystem",
      "Communication": "#/definitions/Communication",
      "CommunicationRequest": "#/definitions/CommunicationRequest",
      "CompartmentDefinition": "#/definitions/CompartmentDefinition",
      "Composition": "#/definitions/Composition",
      "ConceptMap": "#/definitions/ConceptMap",
      "Condition": "#/definitions/Condition",
      "Consent": "#/definitions/Consent",
      "Contract": "#/definitions/Contract",
      "Coverage": "#/definitions/Coverage",
      "CoverageEligibilityRequest": "#/definitions/CoverageEligibilityRequest",
      "CoverageEligibilityResponse": "#/definitions/CoverageEligibilityResponse",
      "DetectedIssue": "#/definitions/DetectedIssue",
```

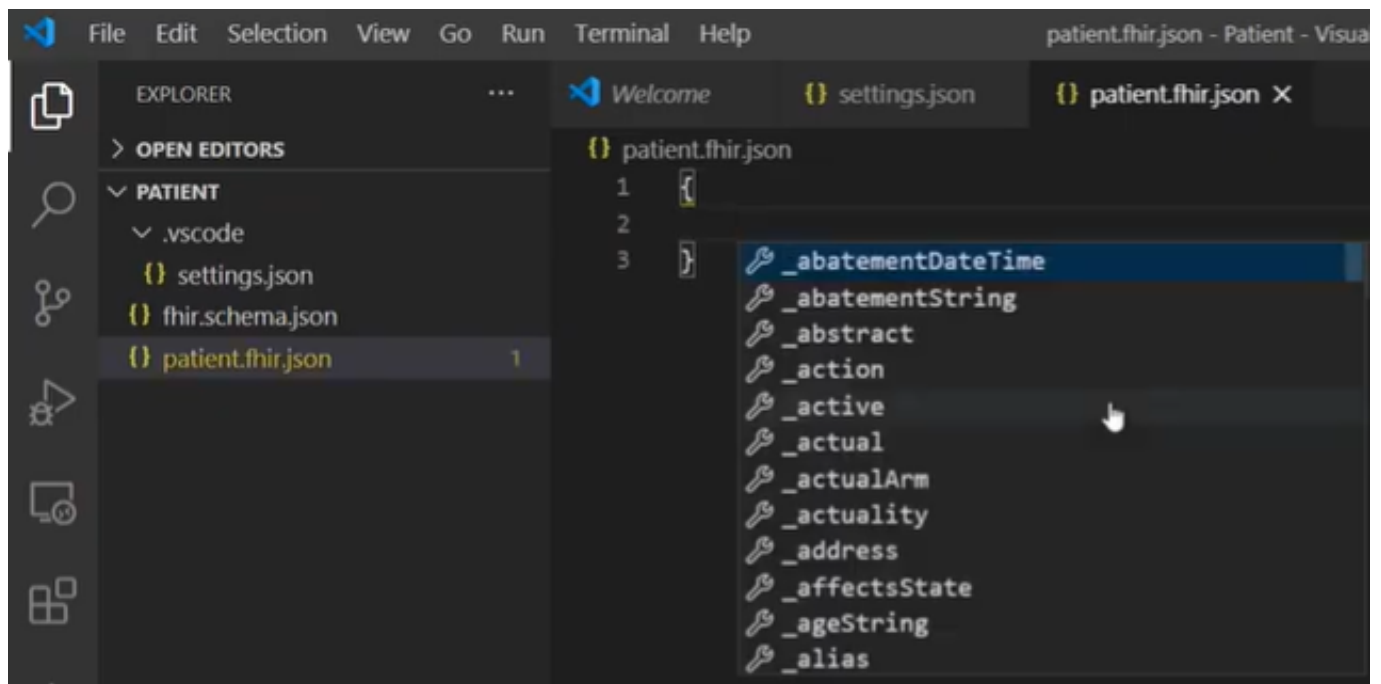
Step 3: Setup the VS code to recognize FHIR schema by modifying settings.json file.

Press CTRL+SHIFT+P and type workspace settings JSON

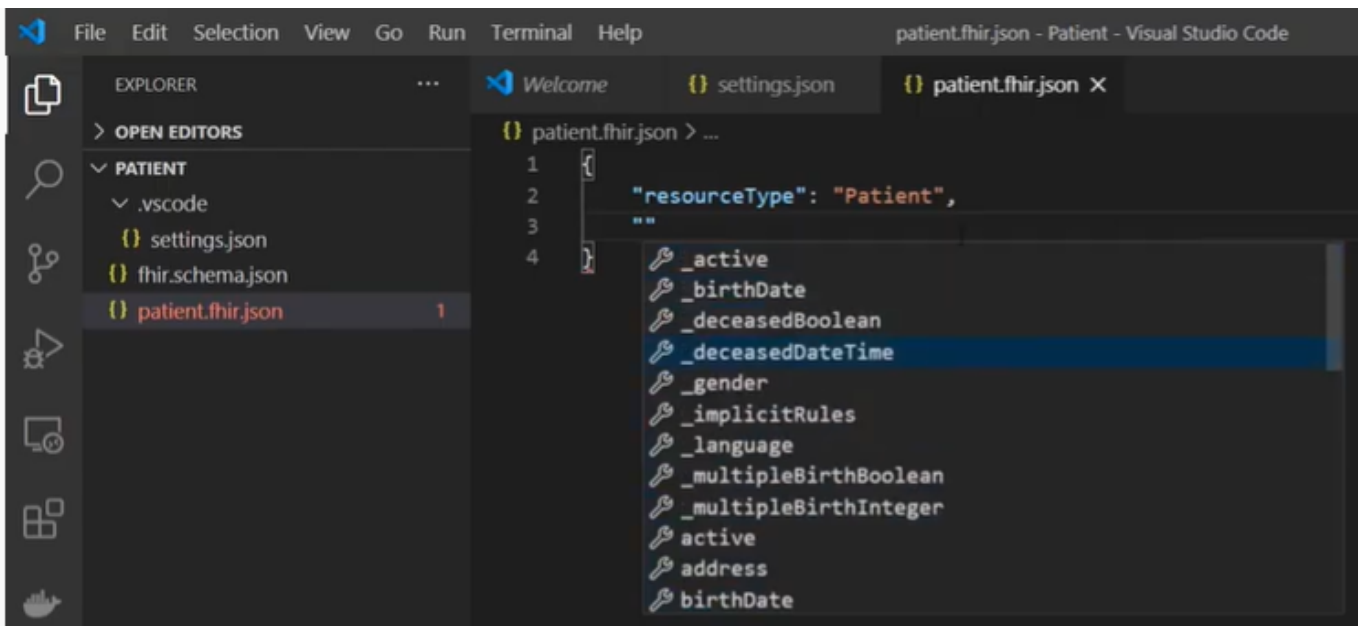


Step 4: Create a new file patient.fhir.json in the same folder.

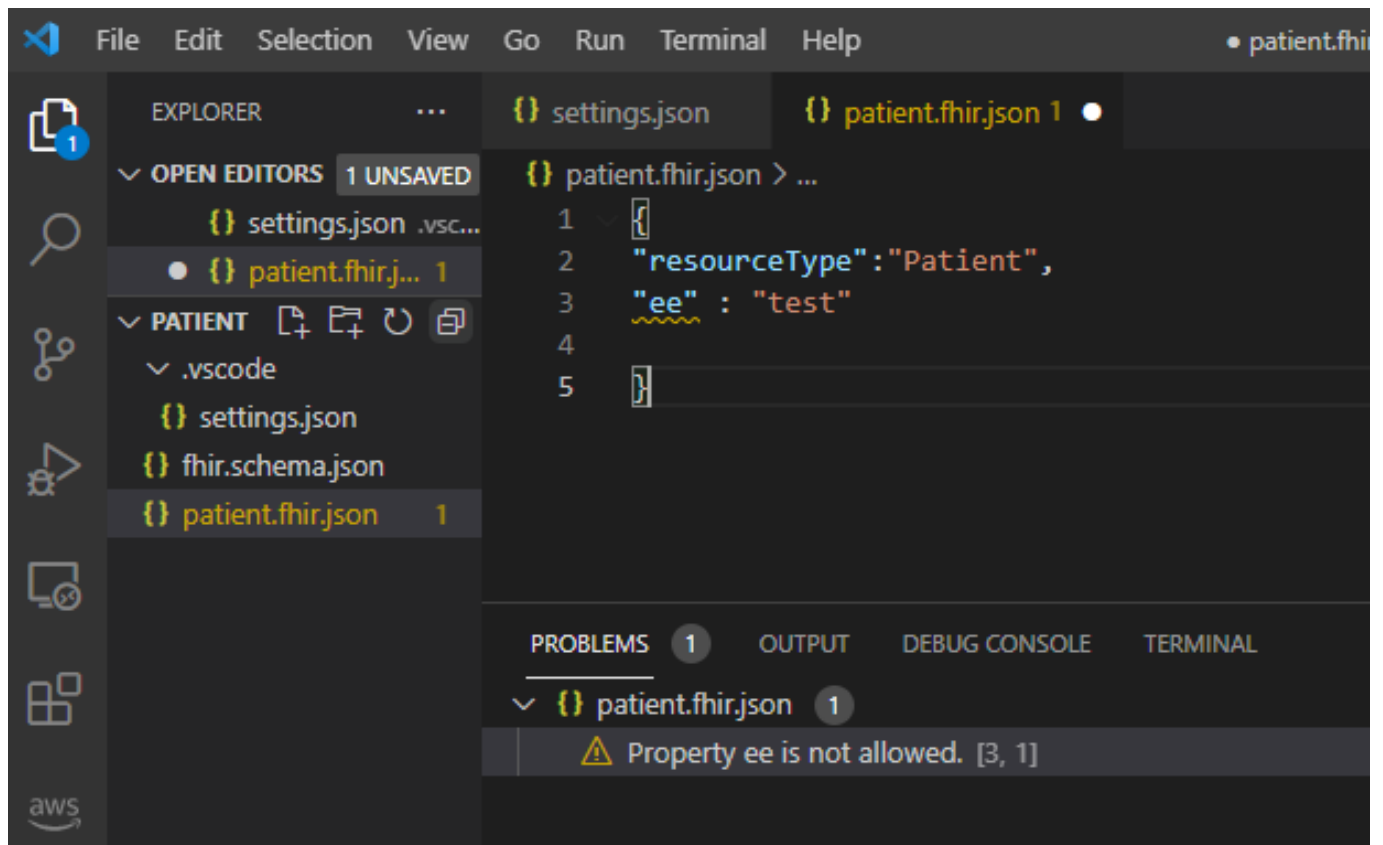
Press Ctrl+Space and you will get all the attributes of the FHIR resources through IntelliSense



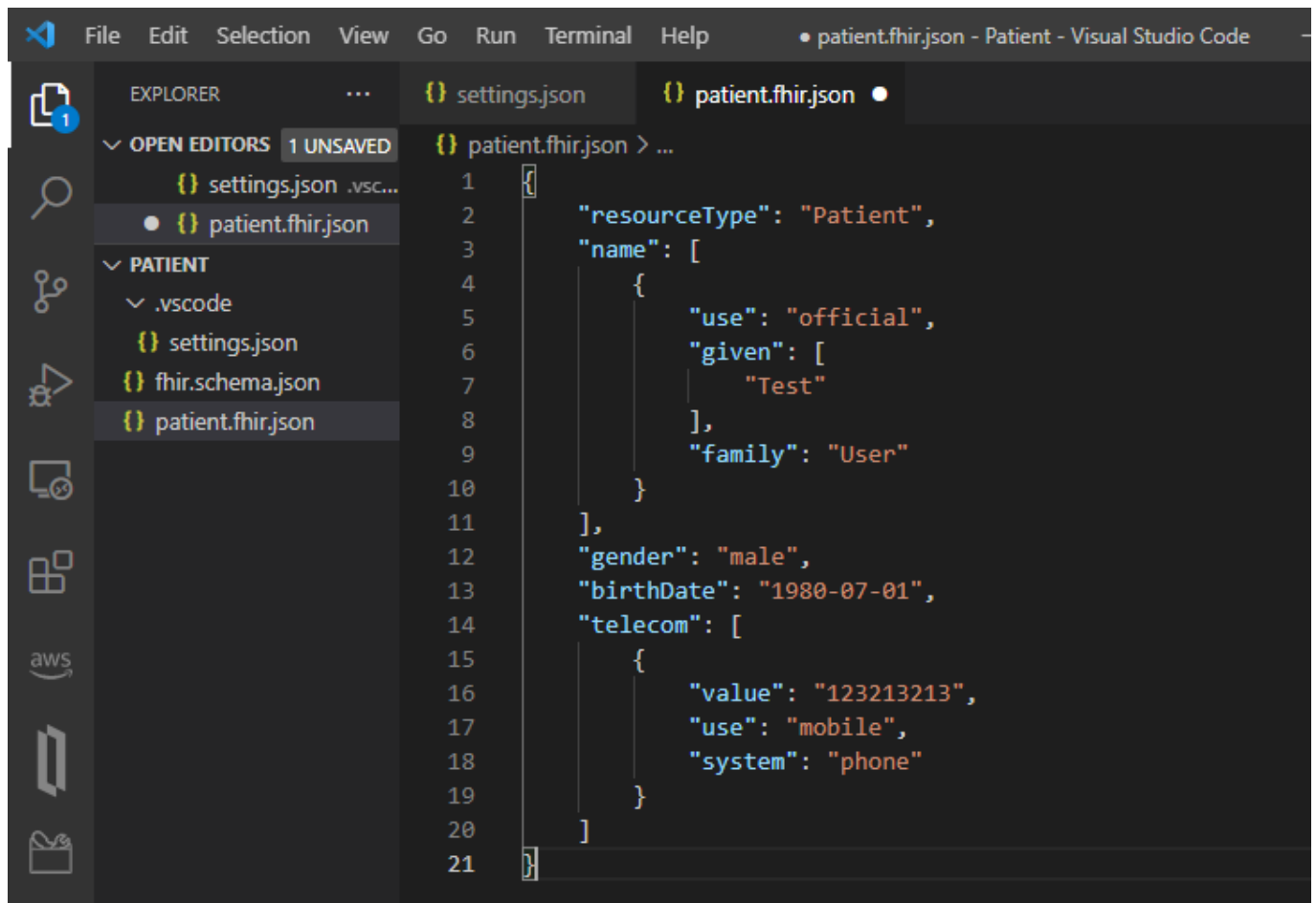
Add Resource type Patient and all the attributes related to patient resource will start appearing in the IntelliSense.



VS Code will automatically validate structure and syntax of the Resource.



By the help of IntelliSense and auto complete we have created and validated our patient resource.



The screenshot shows the Visual Studio Code editor interface. The Explorer sidebar on the left displays the project structure for 'PATIENT', including files like settings.json, fhir.schema.json, and patient.fhir.json. The main editor window shows the content of patient.fhir.json, which is a JSON object representing a FHIR Patient resource. The resource type is 'Patient', the name is 'Test' (official), the gender is 'male', the birth date is '1980-07-01', and there is a mobile phone number '123213213'.

```
1  {}  
2  "resourceType": "Patient",  
3  "name": [  
4    {  
5      "use": "official",  
6      "given": [  
7        "Test"  
8      ]  
9      "family": "User"  
10     }  
11  ],  
12  "gender": "male",  
13  "birthDate": "1980-07-01",  
14  "telecom": [  
15    {  
16      "value": "123213213",  
17      "use": "mobile",  
18      "system": "phone"  
19    }  
20  ]  
21  }
```

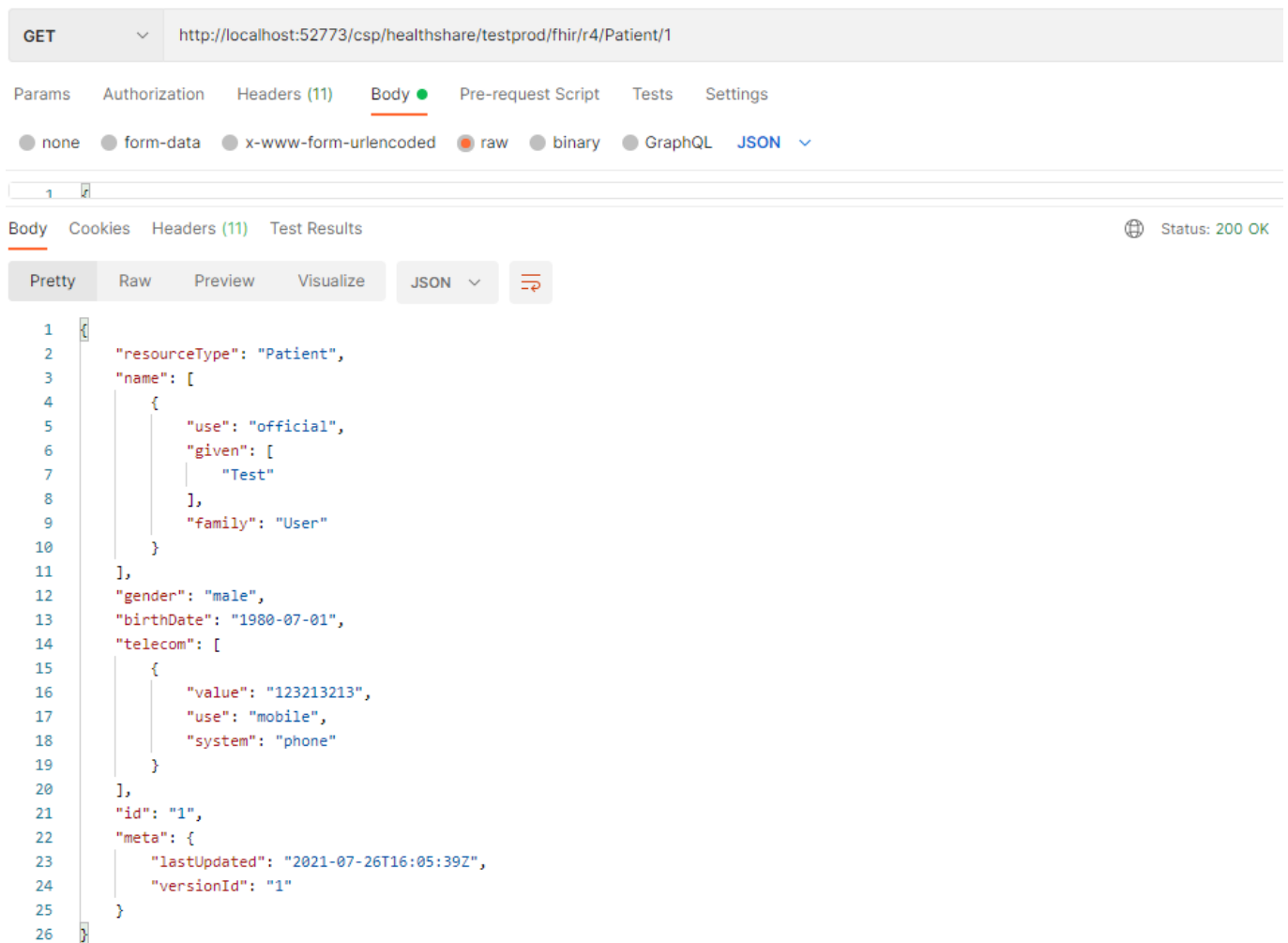
Step 5: Post the create resource in InterSystems FHIR server using Rest API from postman

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** `http://localhost:52773/csp/healthshare/testprod/fhir/r4/Patient` (indicated by a green arrow pointing left)
- Body:** A JSON payload representing a patient resource:

```
1 {
2   "resourceType": "Patient",
3   "name": [
4     {
5       "use": "official",
6       "given": [
7         "Test"
8       ],
9       "family": "User"
10    }
11  ],
12  "gender": "male",
13  "birthDate": "1980-07-01",
14  "telecom": [
15    {
16      "value": "123213213",
17      "use": "mobile",
18      "system": "phone"
19    }
20  ]
21 }
```
- Response:** Status: 201 Created (indicated by a green arrow pointing to the status bar)
- View Options:** Pretty, Raw, Preview, Visualize, HTML

Retrieve the Created Patient Resource by using Get method



The screenshot shows a Postman interface for a GET request to `http://localhost:52773/csp/healthshare/testprod/fhir/r4/Patient/1`. The response body is displayed in JSON format, showing a Patient resource with the following details:

```
1  {
2    "resourceType": "Patient",
3    "name": [
4      {
5        "use": "official",
6        "given": [
7          "Test"
8        ],
9        "family": "User"
10     }
11  ],
12  "gender": "male",
13  "birthDate": "1980-07-01",
14  "telecom": [
15    {
16      "value": "123213213",
17      "use": "mobile",
18      "system": "phone"
19    }
20  ],
21  "id": "1",
22  "meta": {
23    "lastUpdated": "2021-07-26T16:05:39Z",
24    "versionId": "1"
25  }
26 }
```

Congratulations!, We have created, validated our Patient Resource and successfully posted and retrieved to InterSystems FHIR Server using postman.
In this way we can easily create and validate any FHIR Resource.

[#FHIR](#) [#REST API](#) [#Caché](#) [#Ensemble](#) [#InterSystems IRIS for Health](#) [#VSCode](#)

Source

URL:<https://community.intersystems.com/post/creating-and-validating-any-hl7-fhir-resource-using-fhir-schema-help-intellisense-and-auto>