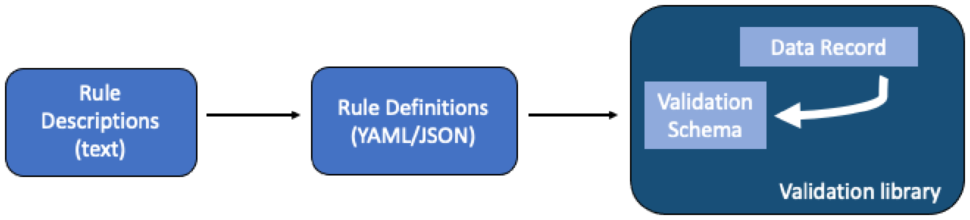
Data Quality Rule Definition Guidelines for UDS Forms

Setup



"Cerberus" python library is used as the basis for defining and validating data quality rules
https://docs.python-cerberus.org/en/stable/index.html
Usage
https://docs.python-cerberus.org/en/stable/usage.html

- Validation schema is a dictionary of key value pairs which can be specified using YAML or JSON format.
- Rule definitions are organized by forms. For each form, create a YAML/JSON file listing the validation rules for the variables in that form.
- Data records collected in the quarantine project will be evaluated against the validation schema.

Example:
Variable "**birthmo"** (subject's month of birth) must be present, and its value should be an integer between 1 – 12.

YAML Rule Definition:

```
birthmo:
    type: integer
    required: true
    min: 1
    max: 12
```

JSON Rule Definition:

```
"birthmo": {
    "type": "integer",
    "required": true,
    "min": 1,
    "max": 12
}
```

{"ptid"=101, "birthmo"=10} => pass validation
{"ptid"=102, "birthmo"=15} => fail validation

Validation Rules

Check the full list of built-in Cerberus rules here
https://docs.python-cerberus.org/en/stable/validation-rules.html

Keywords frequently used in UDS rules are described in the table below,

Keyword	**Description**	**JSON Example**

allowed	Specify the list of allowed values, validation will fail if any other value is given in the data record	schema = {“limit”: {“type”: “integer”, “allowed”: [-1, 10, 100]}}
		
		data = {“limit”: 10} => pass validation
		
		data = {“limit”: 20} => fail validation

forbidden	Specify the list of forbidden values, validation will fail if values in this list are included in the data record	schema = {“user”: {“type”: “string”, “forbidden”: [“viewer”, “editor”]}}
		
		data = {“user”: “admin”} => pass validation
		
		data = {“user”: “viewer”} => fail validation

min, max	Minimum and maximum value allowed (only applicable to object types which support comparison operations). Each keyword can be used independently. Use together to define a range.	Schema = {“length”: {“type”: “float”, “min”: 10.5, “max”: 20.5}}
		
		data = {“length”: 14} => pass validation
		
		data = {“length”: 20.8} => fail validation

nullable	If set to “true”, the field value is allowed to be empty. This rule will be checked on every variable, regardless it’s defined or not. The rule’s constraint defaults “false”.	schema = {“country”: {“type”: “string”, “nullable”: true}}
		
		data = {“country”: “USA”} => pass validation
		
		data = {“country”: “”} => pass validation
		
		schema = {“country”: {“type”: “string”}}
		
		data = {“country”: “”} => fail validation

required	If set to “true”, the field is mandatory, validation will fail when it is missing	schema = {“name”: {“type”: “string”, “required”: true}, “age”: {“type”: “integer”}}
		
		data = {“name”: “Steve”, “age”: 50} => pass validation
		
		data = {“name”: “Debby”} => pass validation
		
		data = {“age”: 40} => fail validation

type	Data type allowed for the value, check Cerberus documentation for the list of type names https://docs.python-cerberus.org/en/stable/validation-rules.html#type	schema = {“limit”: {“type”: “integer”}
		
		data = {“limit”: 10} => pass validation
		
		data = {“limit”: 11.5} => fail validation

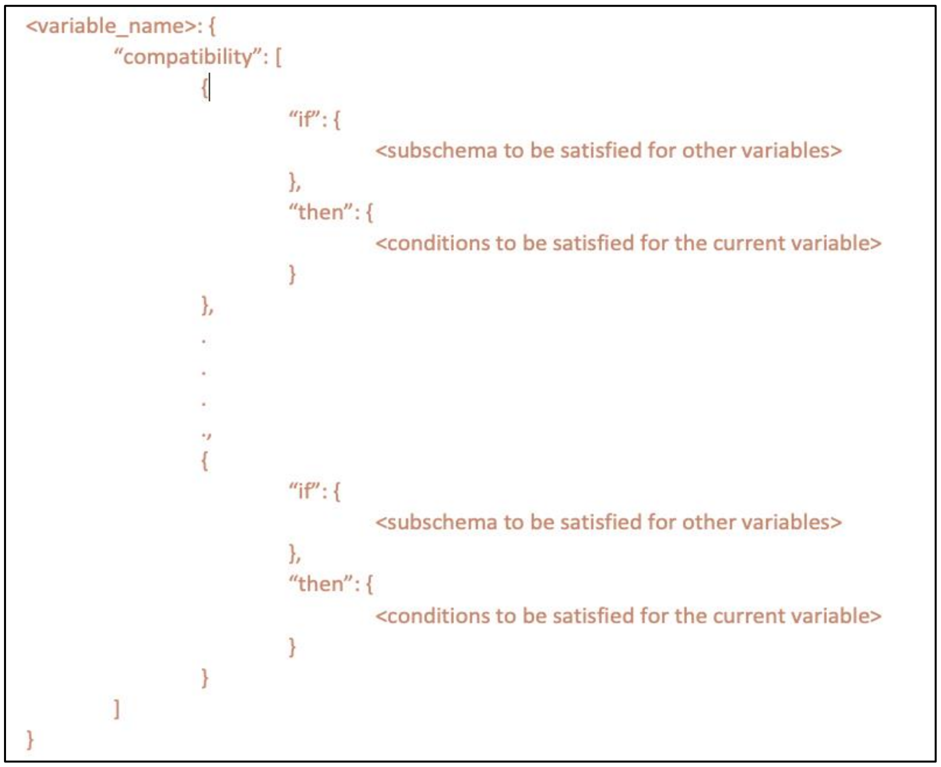
anyof	Allows to define different sets of rules to validate against, field will be considered valid if any of the provided constraints validates the field	schema = {“age”: {“type”: “integer, “anyof”: [{“min”: 0, “max”: 120}, {“allowed”:[999]}]}}
		
		data = {“age”: 40} => pass validation
		
		data = {“age”: 999} => pass validation
		
		data = {“age”: 200} => fail validation

Custom Rules Defined for UDS

(1) **compatibility** – used to specify the list of compatibility constraints for a given variable with other variables within the form or across multiple forms.

Each constraint specifies "if", "then" attributes to allow the application of a subschema based on the outcome of another schema (i.e., when the schema specified under "if" keyword evaluates to true for a given record, then the schema specified under "then" keyword will be evaluated)

The rule definition for "compatibility" keyword should follow the below format:



```
<variable_name>: {
        "compatibility": [
                {
                        "if": {
                                <subschema to be satisfied for other variables>
                        },
                        "then": {
                                <conditions to be satisfied for the current variable>
                        }
                },
                .
                .
                .
                .,
                {
                        "if": {
                                <subschema to be satisfied for other variables>
                        },
                        "then": {
                                <conditions to be satisfied for the current variable>
                        }
                }
        ]
}
```

Example:
If variable "**incntmod**" (primary contact mode w/participant)=6, then variable "**incntmdx**" (specify primary contact mode w/participant) cannot be blank.

YAML Rule Definition:

```
incntmdx:
    type: string
    nullable: true
    compatibility:
      - if:
          incntmod:
            allowed:
              - 6
        then:
          nullable: false
```

JSON Rule Definition:

```
" incntmdx": {
        "type": " string",
        "nullable": true,
        "compatibility": [
                {
                        "if": {"incntmod": {"allowed": [6]}},
                        "then": {"nullable": false}
                }
        ]
}
```

(2) **temporarules** – used to specify the list of longitudinal checks for a given variable.

"orderby" attribute specifies the variable name to order the longitudinal records.
"constraints" attribute specifies the list of checks to be performed on the previous records. Each constraint specifies "previous", "current" attributes to allow the application of a subschema based on the outcome of another schema.

The rule definition for this keyword should follow the below format:

```
<variable_name>: {
          "temporalrules": {
                    "orderby": <variable to order the records>
                    "constraints": [
                              {
                                        "previous": {
                                                  <conditions to be satisfied for the previous record>
                                        },
                                        "current": {
                                                  <conditions to be satisfied for the current record>
                                        }
                              },
                              .
                              .
                              .
                              .,
                              {
                                        "previous": {
                                                  <conditions to be satisfied for the previous record>
                                        },
                                        "current": {
                                                  <conditions to be satisfied for the current record>
                                        }
                              }
                    ]
          }
}
```

Example:
If variable **"taxes"** (difficulty with taxes, business, and other papers)=0 (Normal) at a previous visit, then taxes cannot be =8 (Not applicable/Never did) at the follow-up visit.

YAML Rule Definition:

```
taxes:
    type: integer
    temporalrules:
        orderby: visit_date
        constraints:
           - previous:
                allowed:
                    - 0
             current:
                forbidden:
                    - 8
```

JSON Rule Definition:

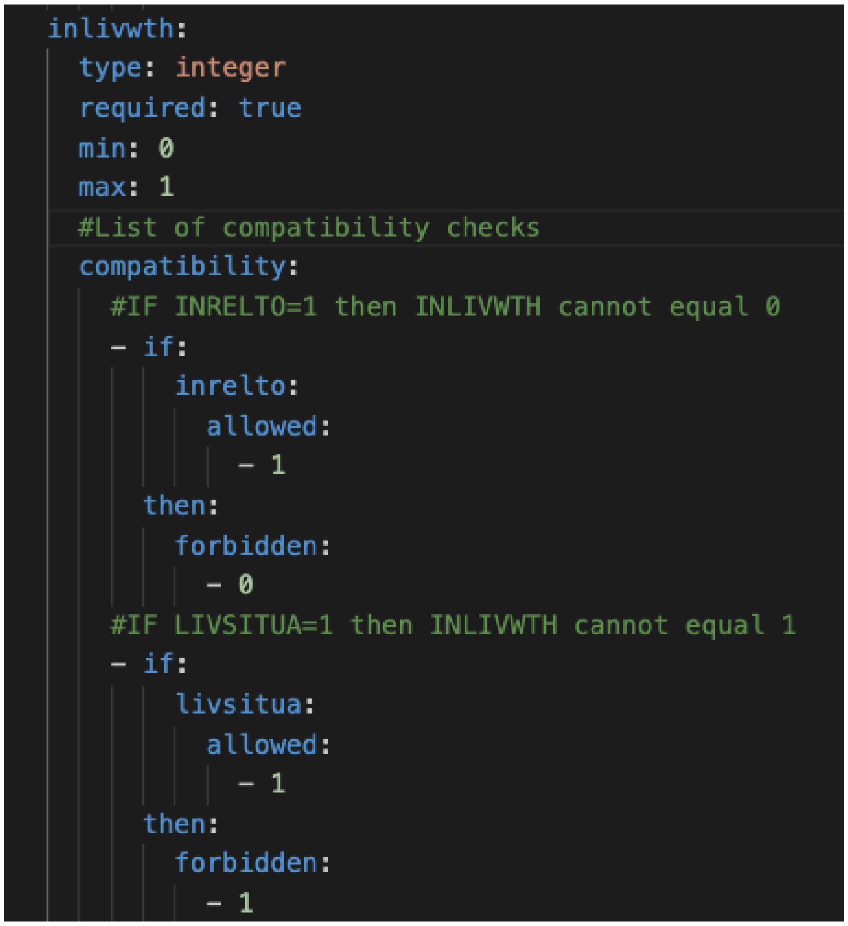
```
"taxes": {
        "type": "integer",
        "temporalrules": {
                "orderby": "visit_date",
                "constraints": [
                        {
                                "previous": {"allowed": [0]},
                                "then": {"forbidden": [8]}
                        }
                ]
        }
}
```

Example from UDSv4 Form A2

Rule descriptions for variable "inlivwth"

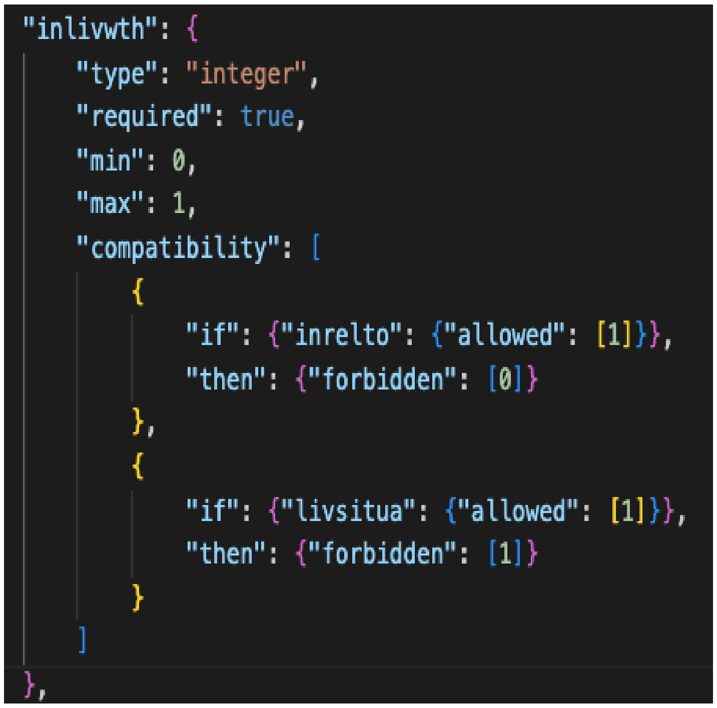
Variable	Test Type	Test Description
inlivwith	Missingness	Q3. inlivwith (lives with participant?) cannot be blank
inlivwith	Conformity	Q3. inlivwith (lives with participant?) must be an integer between 0 and 1
inlivwith, inrelto	Plausibility	IF Q1. inrelto (relationship to participant)=1 *(spouse, partner, companion)* and Q3. inlivwith (lives with participant?) cannot equal to 0 *(no)*
inlivwith, livsitua	Plausibility	IF Q3. inlivwith (lives with participant?)=1 *(yes)* then Form A1, Q6. livsitua (participant's living situation) cannot equal 1 *(lives alone)*

YAML Definition:



```yaml
inlivwth:
  type: integer
  required: true
  min: 0
  max: 1
  #List of compatibility checks
  compatibility:
    #IF INRELTO=1 then INLIVWTH cannot equal 0
    - if:
        inrelto:
          allowed:
            - 1
      then:
        forbidden:
          - 0
    #IF LIVSITUA=1 then INLIVWTH cannot equal 1
    - if:
        livsitua:
          allowed:
            - 1
      then:
        forbidden:
          - 1
```

JSON Definition:



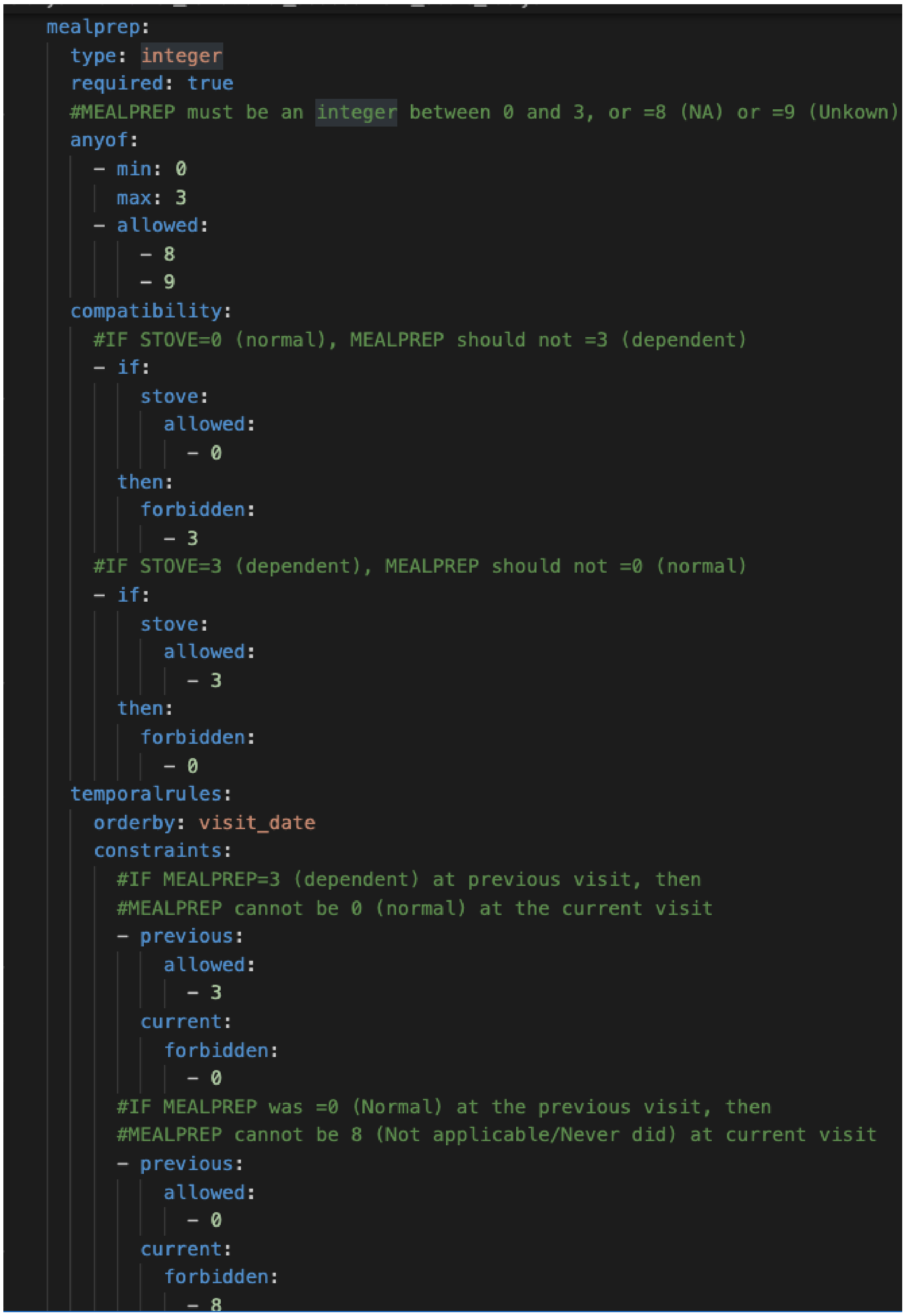
```
"inlivwth": {
    "type": "integer",
    "required": true,
    "min": 0,
    "max": 1,
    "compatibility": [
        {
            "if": {"inrelto": {"allowed": [1]}},
            "then": {"forbidden": [0]}
        },
        {
            "if": {"livsitua": {"allowed": [1]}},
            "then": {"forbidden": [1]}
        }
    ]
},
```

Example from UDSv4 Form B7

Rule descriptions for variable "mealprep"

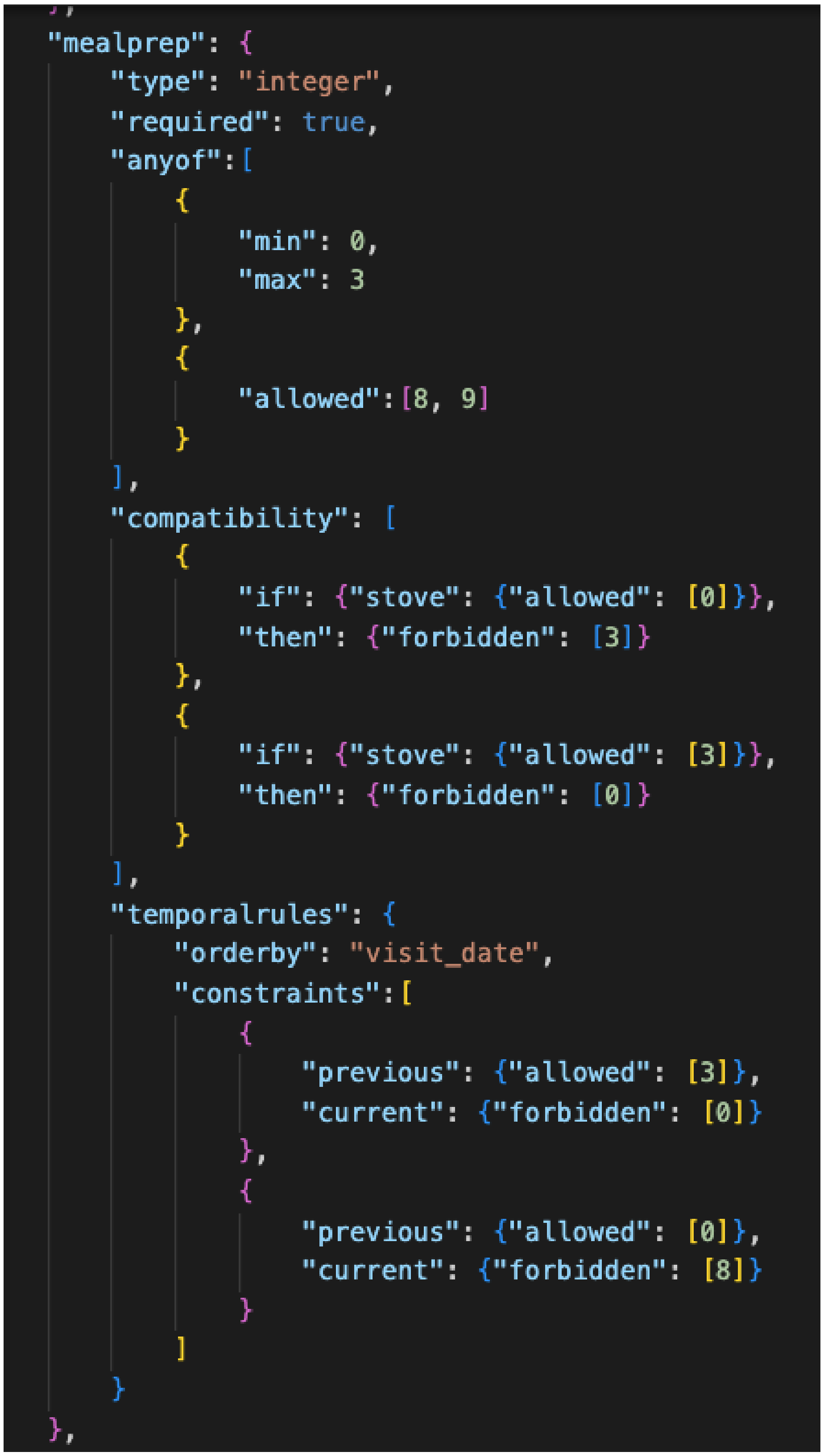
Variable	Test Type	Test Description
mealprep	Missingness	Q6. mealprep (difficulty with preparing meals) cannot be blank
mealprep	Conformity	Q6. mealprep (difficulty with preparing meals) must be an integer between 0 and 3, or =8 (NA) or =9 (Unknown)
mealprep, stove	Plausibility	Q5. stove (difficulty with using stove or kitchen appliances) =0 (normal), but Q6. mealprep (difficulty with preparing meals) =3 (dependent)
mealprep, stove	Plausibility	Q5. stove (difficulty with using stove or kitchen appliances) =3 (dependent), but Q6. mealprep (difficulty with preparing meals) =0 (normal)
mealprep	Plausibility	Q6. mealprep (difficulty with preparing meals) =3 (dependent) at one visit and then Q6. mealprep (difficulty with preparing meals) =0 (normal) at the following visit
mealprep	Plausibility	Q6. mealprep (difficulty with preparing meals) =8 (Not applicable/Never did) at a follow-up visit but Q6. mealprep (difficulty with preparing meals) was =0 (Normal) at the preceding visit

YAML Definition:



```
  mealprep:
    type: integer
    required: true
    #MEALPREP must be an integer between 0 and 3, or =8 (NA) or =9 (Unkown)
    anyof:
      - min: 0
        max: 3
      - allowed:
          - 8
          - 9
    compatibility:
      #IF STOVE=0 (normal), MEALPREP should not =3 (dependent)
      - if:
          stove:
            allowed:
              - 0
        then:
          forbidden:
            - 3
      #IF STOVE=3 (dependent), MEALPREP should not =0 (normal)
      - if:
          stove:
            allowed:
              - 3
        then:
          forbidden:
            - 0
    temporalrules:
      orderby: visit_date
      constraints:
        #IF MEALPREP=3 (dependent) at previous visit, then
        #MEALPREP cannot be 0 (normal) at the current visit
        - previous:
            allowed:
              - 3
          current:
            forbidden:
              - 0
        #IF MEALPREP was =0 (Normal) at the previous visit, then
        #MEALPREP cannot be 8 (Not applicable/Never did) at current visit
        - previous:
            allowed:
              - 0
          current:
            forbidden:
              - 8
```

JSON Definition:



```
    },
    "mealprep": {
        "type": "integer",
        "required": true,
        "anyof":[
            {
                "min": 0,
                "max": 3
            },
            {
                "allowed":[8, 9]
            }
        ],
        "compatibility": [
            {
                "if": {"stove": {"allowed": [0]}},
                "then": {"forbidden": [3]}
            },
            {
                "if": {"stove": {"allowed": [3]}},
                "then": {"forbidden": [0]}
            }
        ],
        "temporalrules": {
            "orderby": "visit_date",
            "constraints":[
                {
                    "previous": {"allowed": [3]},
                    "current": {"forbidden": [0]}
                },
                {
                    "previous": {"allowed": [0]},
                    "current": {"forbidden": [8]}
                }
            ]
        }
    },
```