JSONPath Syntax Reference

REFERENCE

Symbol	Description	Example	Notes/Result Example
$	The **root** of the JSON document	$	Always start here
.	Child operator (dot notation)	$.store	Simple keys; equivalent to ['store']
[]	Bracket notation (child or array access)	$['store'] or $.book[0]	Required for special chars or indexes
*	Wildcard (all direct children)	$.store.* or $.book[*]	All properties or all array items
[n]	Array index (nth item)	$.book[0]	**0-based**: [0] = first, [1] = second
[-n]	From end	$.book[-1]	Last item
[start:end]	Array slice	$.book[0:2]	Items 0 to 1 (end exclusive)
[?( )]	Filter expression (on arrays)	$.book[?(@.price < 10)]	@ = current item; returns matching array items
..	Recursive descent (deep search)	$..title or $..[?(@.price<10)]	Finds at any depth

- Arrays are zero-indexed — first item is always [0]
- Filters [?()] only work on arrays (not objects directly)
- Combine freely: $.store.book[*].title or $..book[?(@.price <= $.expensive)].title

REFERENCE

Accessing JSON Objects

Sample JSON Object

```
{
  "customer": {
    "id": "C12345",
    "profile": {
      "firstName": "Sarah",
      "lastName": "Johnson",
      "email": "sarah.j@email.com"
    },
    "accountType": "premium"
  }
}
```

JSONPath Queries

`$.customer.id` => Returns: `"C12345"`

`$.customer.profile.firstName` => Returns: `"Sarah"`

`$.customer.profile.email` => Returns: `"sarah.j@email.com"`

`$.customer.accountType` => Returns: `"premium"`

Key Insight: Each dot takes you one level deeper into the nested structure. Follow the hierarchy from root to the specific field you need.

Working with JSON Arrays

REFERENCE

Sample JSON with Array

```
{
  "customer": {
    "name": "John Smith",
    "phoneNumbers": [
      "+1-555-0100",
      "+1-555-0101",
      "+1-555-0102"
    ],
    "orders": [
      {"id": "ORD001", "total": 99.99},
      {"id": "ORD002", "total": 149.50},
      {"id": "ORD003", "total": 75.00}
    ]
  }
}
```

Array JSONPath Examples

`$.customer.phoneNumbers[0]` => Returns: `"+1-555-0100"`

`$.customer.phoneNumbers[2]` => Returns: `"+1-555-0102"`

`$.customer.orders[1].id` => Returns: `"ORD002"`

`$.customer.orders[*].total` => Returns: `[99.99, 149.50, 75.00]`

Insight: Arrays use zero-based indexing, meaning the first element is [0], the second is [1], and so on. The wildcard `[*]` is powerful for extracting the same field from every array element.

REFERENCE

Looping Through JSON Objects

When you need to iterate through all properties of an object, JSONPath provides flexible options for accessing multiple fields at once.

```
{
  "customer": {
    "firstName": "Maria",
    "lastName": "Garcia",
    "email": "m.garcia@email.com",
    "phone": "+1-555-0200",
    "tier": "gold"
  }
}
```

Object Iteration Techniques

`$.customer.*` => Returns all values: `["Maria", "Garcia", "m.garcia@email.com", "+1-555-0200", "gold"]`

`$.customer['firstName','email']` => Returns multiple specific fields: `["Maria", "m.garcia@email.com"]`

In contact center applications, you might use object iteration to populate multiple screen pop fields with a single JSONPath expression, improving efficiency.

REFERENCE

Looping Through JSON Arrays

Arrays of objects are common in API responses. Master these patterns to extract data from order histories, contact lists, and transaction records.

```
{
  "transactions": [
    {
      "date": "2026-02-10",
      "amount": 250.00,
      "status": "completed"
    },
    {
      "date": "2026-02-15",
      "amount": 175.50,
      "status": "pending"
    },
    {
      "date": "2026-02-18",
      "amount": 89.99,
      "status": "completed"
    }
  ]
}
```

Array Loop Examples

`$.transactions[*]` => Returns: All transaction objects

`$.transactions[*].amount` => Returns: `[250.00, 175.50, 89.99]`

`$.transactions[*].date` => Returns: All transaction dates

`$.transactions[-1].status` => Returns: `"completed"` (last item)

Remember: Curly braces {} indicate an object (a collection of key-value pairs), while square brackets [] indicate an array (an ordered list of values). Arrays can be the value in a name-value pair—in the example above, "transactions" is the name, and the array [] containing multiple transaction objects {} is its value.