Building Blocks for HTTP APIs
Internet-Draft
Intended status: Standards Track
Expires: 7 June 2026

C. Vasters
Microsoft Corporation

4 December 2025

JSON Structure: Import
draft-vasters-json-structure-import-01

Abstract

This document specifies the $import and $importdefs keywords as extensions to JSON Structure Core. These keywords allow a schema to import definitions from external schema documents.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://json-structure.github.io/import/draft-vasters-json-structure-import.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-vasters-json-structure-import/.

Source for this draft and an issue tracker can be found at https://github.com/json-structure/import.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 7 June 2026.

Table of Contents

1. Introduction

This document specifies the $import and $importdefs keywords, as extensions to JSON Structure Core [JSTRUCT-CORE]. These keywords allow a schema to import definitions from external schema documents.

All type reference expressions in JSON Structure Core, $ref and $extends and $addins, are limited to references within the current schema document.

The $import and $importdefs keywords enable schema authors to incorporate external schema documents into a schema.

Imports do not establish a reference relationship between the importing schema and the imported schema. Imports _copy_ definitions from the imported schema into the importing schema and those definitions are then treated as if they were defined locally.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. The $import and $importdefs Keywords

The $import and $importdefs keywords are used to import definitions from external schema documents into a local namespace within the current schema document.

A schema processor MUST process the $import and $importdefs keywords before processing any other keywords in the schema document.

The result of importing definitions is that the imported definitions are merged into the local definitions section under the designated namespace as if they were defined locally.

A schema that uses $import or $importdefs MAY _shadow_ any imported definitions with local definitions of the same name and in the same namespace, replacing the imported definition entirely. Local definitions take precedence over the imported definitions. A shadowing type cannot reference the imported type that it shadows.

When importing definitions into a local namespace, the processor MUST ensure that all imported cross-references are resolved within the imported definitions themselves and not to the local schema. That means that any jsonpointer instance ($ref or $extends or $addins) within imported definitions MUST be prefixed with the local namespace under which the definitions were imported. This applies recursively to any imported schema that itself contains imports.

3.1. $import Keyword

The $import keyword is a reference expression whose value is an absolute URI pointing to an external schema document. It is used to import all type definitions of the external schema into a local namespace within the current schema document.

When the keyword is used at the root level of a schema, the imported definitions are available in the schema's root namespace. When used within the definitions section, the imported definitions are available in the respective local namespace.

> *Reminder*: Any type declaration at the root level of a schema and any type declaration at the root level of the definitions section is placed in the schema's root namespace per section 3.3 of [JSTRUCT-CORE].

Example for $import at the root level:

```
{
  "$schema": "https://json-structure.org/meta/core/v0/#",
  "$import": "https://example.com/people.json"
}
```

Importing into the root namespace within the definitions section is equivalent to the prior example:

```
{
  "$schema": "https://json-structure.org/meta/core/v0/#",
  "definitions": {
    "$import": "https://example.com/people.json"
  }
}
```

One can also import into any local namespace within the definitions section:

```
{
  "definitions": {
    "People": {
      "$import": "https://example.com/people.json"
    }
  }
}
```

The result of the import is that _all_ definitions from the external schema are available under the People namespace. The namespace structure and any cross-references that exist within an imported schema, including any imports that it may have, are unaffected by being imported.

The $import keyword MAY be used many times within a schema to import multiple external schemas into distinct local namespaces.

3.2. $importdefs Keyword

The $importdefs keyword is a reference expression whose value is an absolute URI pointing to an external schema document.

$importdefs works the same as $import, with the exception that it only imports the definitions section of the external schema and not the root type.

The purpose of $importdefs is to use the type definitions from an external schema as a library of types that can be referenced from within the local schema without importing the root type of the external schema.

4. Examples

4.1. Example: Using $import to import an external schema

Let the external schema be defined as follows:

```
{
  "$schema": "https://json-structure.org/meta/core/v0/#",
  "$id": "https://example.com/people.json",
  "name": "Person",
  "type": "object",
  "properties": {
    "firstName": { "type": "string" },
    "lastName": { "type": "string" },
    "address": { "$ref": "#/definitions/Address" }
  },
  "definitions": {
    "Address": {
      "type": "object",
      "properties": {
        "street": { "type": "string" },
        "city": { "type": "string" }
      }
    }
  }
}
```

The importing schema uses $import to import the external schema into the "People" namespace. The imported Person type is then used in the local schema as the type of the person property:

```
{
  "$schema": "https://json-structure.org/meta/core/v0/#",
  "type": "object",
  "properties": {
    "person": {
      "type": { "$ref": "#/definitions/People/Person" }
    },
    "shippingAddress": {
      "type": { "$ref": "#/definitions/People/Address" }
    }
  },
  "definitions": {
    "People": {
      "$import": "https://example.com/people.json"
    }
  }
}
```

The imported Person type from the root of the external schema is available under the People namespace in the local schema, alongside the imported Address type.

The external schema can also be imported into the root namespace of the local schema by using $import at the root level of the schema document— in which case the imported definitions are available in the root namespace of the local schema:

```
{
  "$schema": "https://json-structure.org/meta/core/v0/#",
  "$import": "https://example.com/people.json",
  "type": "object",
  "properties": {
    "person": {
      "type": { "$ref": "#/definitions/Person" }
    },
    "shippingAddress": {
      "type": { "$ref": "#/definitions/Address" }
    }
  }
}
```

The following schema is equivalent to the prior example:

```
{
  "$schema": "https://json-structure.org/meta/core/v0/#",
  "type": "object",
  "properties": {
    "person": {
      "type": { "$ref": "#/definitions/Person" }
    },
    "shippingAddress": {
      "type": { "$ref": "#/definitions/Address" }
    }
  },
  "definitions": {
    "$import": "https://example.com/people.json"
  }
}
```

4.2. Example: Using $import with shadowing

The external schema remains the same as in Section 4.1.

The importing schema uses $import to import the external schema into the "People" namespace. The imported Person type is then used in the local schema as the type of the person property. The local schema then also defines an Address type that shadows the imported Address type within the same namespace:

```
{
  "$schema": "https://json-structure.org/meta/core/v0/#",
  "type": "object",
  "properties": {
    "person": {
      "type": { "$ref": "#/definitions/People/Person" }
    }
  },
  "definitions": {
    "People": {
      "$import": "https://example.com/people.json",
      "Address": {
        "type": "object",
        "properties": {
          "street": { "type": "string" },
          "city": { "type": "string" },
          "postalCode": { "type": "string" },
          "country": { "type": "string" }
        }
      }
    }
  }
}
```

4.3. Example: Using $importdefs to import the definitions section of an external schema

The external schema remains the same as in Example 4.1.

The importing schema uses $importdefs to import the definitions section of the external schema into the "People" namespace. The imported Address type is then used in the local schema as the type of the shippingAddress property as before. However, the Person type is not imported and available.

```
{
  "$schema": "https://json-structure.org/meta/core/v0/#",
  "type": "object",
  "properties": {
    "shippingAddress": {
      "type": { "$ref": "#/definitions/People/Address" }
    }
  },
  "definitions": {
    "People": {
      "$importdefs": "https://example.com/people.json"
    }
  }
}
```

5. Resolving URIs

When resolving URIs, schema processors MUST follow the rules defined in [RFC3986] and [RFC3987]

This specification does not define any additional rules for resolving URIs into schema documents.

5.1. Enabling the Extensions

The import extensions are available and enabled by default via the extended meta-schema:

```
{
  "$schema": "https://json-structure.org/meta/extended/v0/#",
  "$id": "myschema",
  "$import": "https://example.com/people.json",
  "type": "object",
  "properties": {
    "name": {
      "type": "string",
    },
    "address": {
      "type": { "$ref": "#/definitions/Person/Address" }
    }
  }
}
```

6. Security and Interoperability

* Schema processing engines MUST resolve the absolute URIs specified in $import and $importdefs, fetch the external schemas, and validate them to be schema documents.

* Implementations SHOULD employ caching and robust error handling for remote schema retrieval.

* External schema URIs SHOULD originate from trusted sources.

* Remote fetching of schemas SHOULD be performed over secure protocols (e.g., HTTPS) to mitigate tampering.

* Excessively deep or circular import chains MUST be detected and mitigated to avoid performance degradation and potential denial-of-service conditions.

7. IANA Considerations

This document does not require any IANA actions.

8. Normative References

[JSTRUCT-CORE] Vasters, C., "JSON Structure Core", n.d., <https://json-structure.github.io/core/draft-vasters-json-structure-core.html>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, DOI 10.17487/RFC3987, January 2005, <https://www.rfc-editor.org/rfc/rfc3987>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

Acknowledgments

TODO acknowledge.

Author’s Address

Clemens Vasters
Microsoft Corporation
Email: clemensv@microsoft.com