The Lynx Media Type

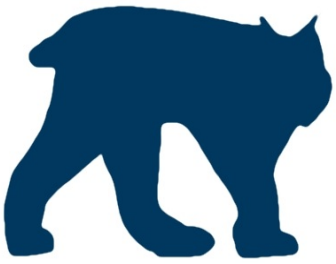

Table of Contents

Introduction

This document defines a new hypertext media type `application/lynx+json` (a Lynx document), and a related metadata media type `application/lynx-spec+json` (a Lynx specification), designed to represent and describe documents and the connections among them, with enough semantic detail to allow the development of client applications without imposing strict display constraints or unnecessary coupling between clients and servers. By reducing display constraints and coupling, developers have more freedom to create client applications that take better advantage of the unique characteristics of each platform and device.

Authors

- Dan Mork dan@humanstuff.com
- Anson Goldade anson@goldade.us
- John Howes john@humanstuff.com

Acknowledgements

We would like to thank the following people for their contributions to this document.

- Dale Sande
- Arun Batchu
- Joe McRae
- Mark Brose

Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Terminology

JSON

Refers to the media type JSON as defined in RFC 4627.

URI

Refers to a URI as defined in RFC 3986.

URL

Refers to a URL as defined in RFC 3986.

URN

Refers to a URN as defined in RFC 3986.

Media Type Name

Refers to a media type name as defined in RFC 6838.

Base Hint

Refers to one of the following hints: `container`, `link`, `form`, `submit`, `content`, and `text`.

Display

The conveyance of a value to a user in a manner appropriate for the user agent and the user.

Resource

See Section 5.2.1.1 "Resources and Resource Identifiers" in the Fielding Dissertation.

Representation

See Section 5.2.1.2 "Representations" in the Fielding Dissertation.

Compliance

An implementation is "non-compliant" if it fails to satisfy one or more of the MUST or REQUIRED level requirements. An implementation that satisfies all of the MUST or REQUIRED and all of the SHOULD level requirements is said to be "unconditionally compliant"; an implementation that satisfies all of the MUST level requirements but not all of the SHOULD level requirements is said to be "conditionally compliant."

Parameters for `application/lynx+json`

base

- The parameter is OPTIONAL.
- If present, its value MUST be an absolute URI to be used for relative URI resolution. For more information, see Section 5.1.2 "Base URI from the Encapsulating Entity" in RFC 3986.

Example

```
HTTP/1.1 200 OK
Content-Type: application/lynx+json;base="http://example.com/greetings/hello-world"


{
  "value": "Hello, World!",
  "spec": "http://example.com/specs/greeting"
}
```

realm

- The parameter is OPTIONAL.
- If present, its value MUST comply with the rules defined for Realm URI.

Example

```
HTTP/1.1 200 OK
Content-Type: application/lynx+json;realm="http://example.com/entrance/greeter"

{
  "value": "Hello, World!",
  "spec": "http://example.com/specs/greeting"
}
```

Parameters for `application/lynx-spec+json`

None

Interpretation of URI Fragments for `application/lynx+json`

TODO: provide documentation for how the user agent interprets fragments. This may be here or it may be here or it may be in the "process for displaying content" or another section.

Interpretation of URI Fragments for `application/lynx-spec+json`

URI fragments MUST be ignored.

Registration Status

The registration status of this media type is as follows:

- `application/lynx+json` (unregistered)
- `application/lynx-spec+json` (unregistered)

File Extensions

- `.lnx` => `application/lynx+json`
- `.lnxs` => `application/lynx-spec+json`

Documents

While a JSON document may begin with an object or an array, a Lynx document MUST begin with a value/spec pair.

Content

The content of all documents defined by this media type are extensions of JSON and MUST comply with the rules of that media type. All usages of the JSON keywords "string", "number", "object", "array", "true", "false", "null", and "value" refer to the concepts in the JSON specification unless otherwise stated. The word "property" is used to refer to the concept of "pair", "name/value pair", or "member" in the JSON specification. The word "integer" is used to refer to a JSON "number" with no fractional part. The phrase "container value" is used to refer to objects and arrays. The phrase "simple value" is used to refer to strings, numbers, `true`, `false`, or `null`.

Value/Spec Pairs

Lynx is built on a simple JSON object that conveys a value (data) and a specification (metadata). This object is called a "value/spec pair". A user agent uses the information conveyed in the value/spec pair to effectively display the information to a user.

Format Rules

A value/spec pair:

- MUST be a JSON object
- MUST have a `value` property whose value MAY be any JSON value.
- MUST have a `spec` property whose value MUST be a specification object or a URL to a remote specification object.

A Lynx document MUST start with a value/spec pair. If a value/spec pair is a document:

- it SHOULD have a `realm` property. If present, the value MUST comply with the rules defined for realm URI.
- it MAY have a `base` property. If present, the value MUST be an absolute URI to be used to represent the URI of the document for relative URI resolution. For more information, see Section 5.1.1 "Base URI Embedded in Content" in RFC 3986.

If the value of the `value` property is an object, the value/spec pair MAY be condensed by replacing the `value` property with all of the properties of the object.

Examples

Value/Spec Pair

```
{
  "value": {
    "title": "Groundhog Day",
    "rating": 8.1
  },
  "spec": "http://www.example.com/specs/movie-summary"
}
```

Condensed Value/Spec Pair

```
{
  "title": "Groundhog Day",
  "rating": 8.1,
  "spec": "http://www.example.com/specs/movie-summary"
}
```

Document

```
{
  "base": "http://www.example.com/movies/what-about-bob/title",
  "realm": "http://www.example.com/movies/titles",
  "value": "What About Bob?",
  "spec": "/specs/movies/title"
}
```

User Agent Rules

If the `spec` property value is a specification URL, then the user agent MUST fetch the remote specification object using the default retrieval action for the protocol (e.g. GET for HTTP).

If the user agent encounters a relative URI it MUST resolve an absolute URI using the process defined in Section 5 "Reference Resolution" in RFC 3986. For example, the `spec` property in the document above would resolve to `http://www.example.com/specs/movies/title`.

Values

A value can be any JSON value. They can be divided into two groups: simple values and container values.

Simple Values

Simple values include strings, numbers, `true`, `false`, and `null`.

Container Values

Container values include objects and arrays.

Format Rules

The JSON media type specification controls the format of JSON values.

Examples

Strings

```
{
  "value": "Fletch",
  "spec": {
    "hints": [ "text" ]
  }
}
```

Numbers

```
{
  "value": 42,
  "spec": {
    "hints": [ "text" ]
  }
}
```

Literals

```
{
  "value": true,
  "spec": {
    "hints": [ "text" ]
  }
}
```

Containers

```
{
  "value": {},
  "spec": {
    "hints": [ "container" ]
  }
}
```

```
{
  "value": [],
  "spec": {
    "hints": [ "container" ]
  }
}
```

User Agent Rules

If a value does not have a spec, then its value is not understood. If a value is not understood, the user agent MAY choose to not display the value.

Specifications

A specification is a JSON object that describes a JSON value. The media type for a specification object is `application/lynx-spec+json`.

Format Rules

A specification object MAY have any number of properties. This document defines the properties in the Lynx vocabulary (e.g. `hints`, `children`, etc.). Authors may also define extended properties in their own vocabularies.

Example

```
{
  "value": "Fletch",
  "spec": {
    "hints": [ "label", "text" ]
  }
}
```

Specification Properties

A specification property is used to provide metadata for a JSON value. A specification property name is a URI. If a specification property name is unqualified (relative URI), it MUST be assumed to be in the `http://lynx-json.org/` namespace. For example, a specification property of `visibility` MUST be considered to be `http://lynx-json.org/visibility`.

Extensions

This media type may be extended to include new specification properties. The name of an extended specification property MUST be a qualified name (absolute URI).

This media type may also be extended to include new hints. The name of an extended hint MUST be a qualified name (absolute URI).

Visibility Specification Property

Name

`http://lynx-json.org/visibility`

Meaning

The `visibility` property of a specification describes how a value should or should not be displayed to the user.

Format Rules

The `visibility` property is OPTIONAL. If present, the value MUST be one of the following: `visible`, `hidden`, or `concealed`; otherwise, its value MUST be presumed to be `visible`.

Example

```
{
  "title": "Movie Trivia: What was Fletch's First Name?",
  "answer": "Irwin",
  "spec": {
    "hints": [ "container" ],
    "children": [
      {
        "name": "title",
        "hints": [ "label", "text" ]
      },
      {
        "name": "answer",
        "hints": [ "text" ],
        "visibility": "concealed"
      }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

- If `hidden`, the user agent MUST NOT display the value to the user.
- If `visible`, the user agent MUST display the value to the user.
- If `concealed`, the user agent SHOULD conceal the value. The user agent SHOULD provide the user with a control to reveal the value.
- If the user agent does not understand the value of the `visibility` property, it MUST consider the value to be `visible`.

User Agent Considerations

The user agent must anticipate that the value may be input by the user. If `visibility` is `concealed`, the user agent should conceal the user's input.

Name Specification Property

Name

`http://lynx-json.org/name`

Meaning

The `name` property of a specification identifies the property value or array item value that the specification describes.

Format Rules

- If the specification describes a property of an object value or an item in an array value, then `name` is REQUIRED.
- If the `name` property refers to a property of an object value, then its value MUST be the name of the property. The referenced property MAY be undefined.
- If the `name` property refers to an item in an array value, then its value MUST be the index of the item in the array value.

Example

Defined Property

```
{
  "actor": "Chevy Chase",
  "spec": {
    "hints": [ "container" ],
    "children": [
      {
        "name": "actor",
        "hints": [ "text" ]
      }
    ]
  }
}
```

Undefined Property

```
{
  "spec": {
    "hints": [ "container" ],
    "children": [
      {
        "name": "actor",
        "hints": [ "text" ]
      }
    ]
  }
}
```

Array Item

```
{
  "value": [
    "Chevy Chase"
  ],
  "spec": {
    "hints": [ "container" ],
    "children": [
      {
        "name": "0",
        "hints": [ "text" ]
      }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

None

User Agent Considerations

None

Hints Specification Property

Name

`http://lynx-json.org/hints`

Meaning

The `hints` property of a specification describes the meaning of a value.

Format Rules

The `hints` property is REQUIRED when referring to a value that does not have its own `spec` property. Otherwise, the `hints` property is OPTIONAL. If present, it MUST comply with the rules below for a hint array.

Hint Object

- MUST have a `name` property to identify the hint. If a `name` value is unqualified, it MUST be assumed to be in the `http://lynx-json.org/` namespace. For example, a value of `text` MUST be considered to be `http://lynx-json.org/text`.
- MAY have a `documentation` property. If present, its value MUST be a URL defining the location of human-readable documentation for the hint.

Hint Array

- MUST be ordered from more specific to less specific.
- MAY contain a string instead of a hint object. If a string value is present, it is assumed to be a hint object with a name equal to the string value. For example, a hint of `"text"` is equivalent of `{ "name": "text" }`.
- SHOULD end with a base hint.

Example

```
{
  "name": "Irwin M. Fletcher",
  "spec":{
    "hints": [ "container" ],
    "children": [
      {
        "name": "name",
        "hints": [
          {
            "name": "http://www.example.com/character-name",
            "documentation": "http://www.example.com/hints/character-name"
          },
          "text"
        ]
      }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

- The user agent MUST process hints in order from more specific to less specific. The user agent MUST find the most specific hint that it understands and process the value based on that understanding.
- If the user agent does not understand any hints it SHOULD ignore the value.
- SHOULD understand all of the base hints.

User Agent Considerations

None

Text Hint

Name

`http://lynx-json.org/text`

Meaning

A `text` hint describes a value that should be interpreted as text.

Related Hints

None

Synonyms

None

Format Rules

If the value is present, it must be a simple value.

Examples

```
{
  "lastName": "Cronauer",
  "spec": {
    "hints": [ "container" ],
    "children": [
      {
        "name": "lastName",
        "hints": [ "text" ]
      }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

None

User Agent Considerations

A user agent must anticipate that a value described by a `text` hint may contain any length and combination of characters, including line breaks.

Line Hint

Name

`http://lynx-json.org/line`

Meaning

A `line` hint describes a value that should be interpreted as text without breaks.

Related Hints

- ["line", "text"]

Synonyms

None

Format Rules

- MUST NOT contain a line feed (U+000A) character
- MUST NOT contain a carriage return (U+000D) character

Examples

```
{
  "password": "",
  "spec": {
    "hints": [ "container" ],
    "children": [
      {
        "name": "password",
        "hints": [ "line", "text" ],
        "visibility": "concealed",
        "input": true
      }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

None

User Agent Considerations

None

Label Hint

Name

`http://lynx-json.org/label`

Meaning

A `label` hint describes a value that represents a distinguishing name for another value.

Related Hints

- `[ "label", "text" ]`
- `[ "label", "content" ]`
- `[ "label", "container" ]`

Synonyms

- `title`

Format Rules

None

Examples

Text

```
{
  "value": {
    "label": {
      "value": "First Name",
      "spec": {
        "hints": [
          "label",
          "text"
        ]
      }
    },
    "firstName": {
      "value": "",
      "spec": {
        "hints": [
          "text"
        ],
        "labeledBy": "label"
      }
    }
  },
  "spec": {
    "hints": [ "container" ]
  }
}
```

Content

```
{
  "value": {
    "label": {
      "value": {
        "src": "./title.jpg",
        "type": "image/jpeg",
        "alt": "First Name"
      },
      "spec": {
        "hints": [
          "label",
          "content"
        ]
      }
    },
    "firstName": {
      "value": "",
      "spec": {
        "hints": [
          "text"
        ],
        "labeledBy": "label"
      }
    }
  },
  "spec": {
    "hints": [ "container" ]
  }
}
```

Container

```
{
  "value": {
    "label": {
      "value": {
        "heading": {
          "value": "The Hateful Eight",
          "spec": {
            "hints": [ "text" ]
          }
        },
        "subHeading": {
          "value": "No One Comes Up Here Without a Damn Good Reason",
          "spec": {
            "hints": [ "text" ]
          }
        }
      },
      "spec": {
        "hints": [
          "label",
          "container"
        ],
        "children": [
          { "name": "heading" },
          { "name": "subHeading" }
        ]
      }
    },
    "synopsis": {
      "value": "Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do e
iusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam
, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequa
t.",
      "spec": {
        "hints": [
          "text"
        ],
        "labeledBy": "label"
      }
    }
  },
  "spec": {
    "hints": [ "container" ]
  }
}
```

Authoring Rules

None

Authoring Considerations

- Consider labeling content to help the user to identify the content.

User Agent Rules

The user agent should consider the text content of the value described by the `label` hint to be the distinguishing name. For example, in the "Container" section above, the user agent would consider the distinguishing name to be:

The Hateful Eight
No One Comes Up Here Without a Damn Good Reason

User Agent Considerations

- Consider displaying `label` content more prominently.
- Consider allowing users to select or navigate to content based on its relationship to a label.

Content Hint

Name

`http://lynx-json.org/content`

Meaning

A `content` hint describes a value that contains or references content that is to be considered a part of the containing representation (embedded).

Related Hints

None

Synonyms

None

Format Rules

If the value is present, it must comply with the following rules:

- MUST be an object.
- MUST have an `src` property for remote content or a `data` property for inline content.
- SHOULD have an `alt` property for alternate text to be displayed if the content cannot be displayed or if the user cannot view it.
- MAY have a `scope` property whose value specifies the content realm intended for display. If present, the value MUST comply with the rules defined for realm URI.
- MUST NOT contain any other properties that are described by a specification.
- MAY contain other properties that are not described by a specification.

`src` present

If the value has an `src` property:

- the `src` property MUST be a valid URI.
- the value MAY have a `type` property whose value must be a valid media type name to indicate the expected media type of the content.
- the value MUST NOT have `data` or `encoding` properties.

`data` present

If the value has a `data` property:

- the `data` property MUST be a string representing the content to be embedded.
- the value MUST have a `type` property whose value must be a valid media type name to indicate the media type of the content encoded in the `data` property.
- the value MAY have an `encoding` property whose value MUST be `utf-8` or `base64`. If not present, the value is `utf-8`.
- the value MUST NOT have an `src` property.

Examples

`src` present

```
{
  "src": "http://www.example.com/fletch-movie-review.pdf",
  "alt": "Movie Review of Fletch",
  "spec": {
    "hints": [ "content" ]
  }
}
```

`data` present

```
{
  "data": "# Review of Fletch\n##Pros\n\nToo many to list.\n##Cons\n\nNone!",
  "type": "text/markdown",
  "alt": "Movie Review of Fletch",
  "spec": {
    "hints": [ "content" ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

- If the user agent is unable to display the content, it MUST display a link to the content, with the text of the `alt` property, if present.
- If the type of the content is not in the range `text/*`, then the user agent SHOULD consider the value of the `alt` property to be the text content for the value.

Input

If the specification describing the value has an `input` property, the user agent MUST provide the user with a control to supply the content.

User Agent Considerations

The user agent must anticipate that a `content` object may contain no visible content.

Image Hint

Name

`http://lynx-json.org/image`

Meaning

An `image` hint describes a value that contains or references image content that is to be considered a part of the containing representation (embedded).

Related Hints

- `[ "image", "content" ]`

Synonyms

None

Format Rules

If the value is present, it must comply with the following rules:

- MUST comply with the `content` hint format rules.
- MAY have a `height` property whose value MUST be a number representing the natural height of the image in pixels.
- MAY have a `width` property whose value MUST be a number representing the natural width of the image in pixels.

Examples

```
{
  "title": "Bill Murray",
  "photo": {
    "src": "http://www.fillmurray.com/g/300/400",
    "alt": "Picture of Bill Murray",
    "width": 300,
    "height": 400,
  },
  "spec": {
    "hints": [ "container" ],
    "children": [
        {
            "name": "title",
            "hints": [ "label", "text" ]
        },
        {
          "name": "photo",
          "hints": [ "image", "content" ]
        }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

If the user agent resizes the image, it MUST maintain the natural aspect ratio and it SHOULD provide a control to view the image in its natural size.

User Agent Considerations

None

Container Hint

Name

`http://lynx-json.org/container`

Meaning

A `container` hint describes a value that should be interpreted as a container of other values.

Related Hints

None

Synonyms

- `object`
- `array`

Format Rules

If the value is present, it must comply with the following rules:

- if the value is an object, it MAY have a `scope` property whose value specifies the content realm intended for display. If present, the value MUST comply with the rules defined for realm URI.

Examples

Object Value

```
{
  "title": "Groundhog Day",
  "spec": {
    "hints": [ "container" ],
    "children": [
        {
            "name": "title",
            "hints": [ "label", "text" ]
        }
    ]
  }
}
```

Array Value

```
{
  "value": [
    "Print this!",
    "Act like you don't give a crap and you fit right in.",
    "You know if you shoot me, you're liable to lose all of those humanitarian awa
rds."
  ],
  "spec": {
    "hints": [ "container" ],
    "children": {
      "hints": [ "http://example.com/movie-quote", "text" ]
    }
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

- TBD - Need to add rules for container of inputs.

- The user agent SHOULD consider the sum of the text content for all values in the container to be the text content for the container.

User Agent Considerations

- The user agent must anticipate that a container may contain simple values, container values, or a combination of both.
- A user agent must anticipate that a value described by an `container` hint may contain any length and combination of values, including repeated values.

Header Hint

Name

`http://lynx-json.org/header`

Meaning

A `header` hint describes a value that introduces other content.

Related Hints

- `[ "header", "container" ]`

Synonyms

None

Format Rules

None

Example

```
{
  "header": {
    "label": "Review of Fletch"
  },
  "description": "Fletch is fantastic!",
  "spec": {
    "hints": [ "container" ],
    "children": [
        {
            "name": "header",
            "hints": [ "header", "container" ],
            "children": [
              {
                "name": "label",
                "hints": [ "label", "text" ]
              }
            ]
        },
      {
"name": "description",
"hints": [ "text" ]
      }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

None

User Agent Considerations

In order for a header to effectively introduce content, the user agent should display the header in a way that accomplishes that goal. For example, in print, a header often stands on its own line and is often followed by a rule.

Complement Hint

Name

`http://lynx-json.org/complement`

Meaning

A `complement` hint describes a value that is secondary in nature to the sibling values before it, after it, or both.

Related Hints

- `["complement", "container"]`

Synonyms

None

Format Rules

None

Example

```
{
  "spec": {
    "hints": [ "container" ],
    "children": [
      {
        "name": "bio",
        "hints": [ "text" ]
      },
      {
        "name": "sidebar",
        "hints": [ "complement", "container" ],
        "children": {
          "hints": [ "text" ]
        }
      }
    ]
  },
  "value": {
    "bio": "Cornelius Crane \"Chevy\" Chase is an American comedian, actor, writer
 and producer.",
    "sidebar": [
      "Generally, Chevy is short for Cheviot which is a breed of sheep.",
      "Cheviot Hills, also known as Chevy Chase, is a region of England near the S
cottish border."
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

None

User Agent Considerations

None

Group Hint

Name

`http://lynx-json.org/group`

Meaning

A `group` hint describes a value that is a container of other values that should be interpreted as an idea. Values in a group are closely related to each other in some way.

Related Hints

- `[ "group", "container" ]`
- `[ "group", "link" ]`
- `[ "group", "submit" ]`
- `[ "group", "form" ]`

Synonyms

None

Format Rules

None

Example

```
{
  "spec": {
    "hints": [ "group", "container" ],
    "children": {
      "hints": [ "text" ]
    }
  },
  "value": [
    "The",
    "quick",
    "brown",
    "fox…"
  ]
}
```

Authoring Rules

None

Authoring Considerations

Use a `group` hint when the emphasis is more on the group than on the values in the group.

User Agent Rules

None

User Agent Considerations

In order to emphasize the group more than the values in the group, a user agent should display the group in a way that accomplishes that goal. For example, a user agent may choose to display the values in a group in the primary direction of reading order, such as left-to-right in English (as opposed to the secondary direction of reading order top-to-bottom).

List Hint

Name

`http://lynx-json.org/list`

Meaning

A `list` hint describes a value that is a container of independent values that are related to each other in some way.

Related Hints

- `[ "list", "container" ]`
- `[ "list", "link" ]`
- `[ "list", "submit" ]`
- `[ "list", "form" ]`

Synonyms

None

Format Rules

None

Example

```
{
  "spec": {
    "hints": [ "list", "container" ],
    "children": {
      "hints": [ "text" ]
    }
  },
  "value": [
    "Milk",
    "Eggs",
    "Flour"
  ]
}
```

Authoring Rules

None

Authoring Considerations

Use a `list` hint when more emphasis should be placed on the individual values in the list.

User Agent Rules

None

User Agent Considerations

In order to emphasize the individual values in a list while still conveying the closeness of their relationship, a user agent should display the list in a way that accomplishes that goal. For example, a user agent may choose to display the values in a list in the secondary direction of reading order, such as top-to-bottom in English (as opposed to the primary direction of reading order of left-to-right) to emphasize each value, but reduce the space between the values to convey their interrelationships.

Table Hint

Name

`http://lynx-json.org/table`

Meaning

A `table` hint describes a `list` of values that are comparable to each other in some way.

Related Hints

- `[ "table", "list", ... ]`

Synonyms

None

Format Rules

If the value is present, it must comply with the following rules:

- MUST contain only `header` or `group` values.

Example

```
{
  "spec": {
    "hints": [ "table", "list", "container" ]
  },
  "value": [
    {
      "spec": {
        "hints": [ "header", "container" ],
        "children": {
          "hints": [ "text" ]
        }
      },
      "value": [
        "Rank",
        "Movie",
        "Inflation Adjusted Gross"
      ]
    },
    {
      "spec": {
        "hints": [ "group", "container" ],
        "children": {
          "hints": [ "text" ]
        }
      },
      "value": [
        1,
        "How the Grinch Stole Christmas",
        "$406,213,839"
      ]
    },
    {
      "spec": {
        "hints": [ "group", "container" ],
        "children": {
          "hints": [ "text" ]
        }
      },
      "value": [
        2,
        "Despicable Me 2",
        "$381,646,568"
      ]
    },
  ]
}
```

Authoring Rules

None

Authoring Considerations

Use a `table` hint when values in a list are comparable to each other.

User Agent Rules

None

User Agent Considerations

The user agent should display the table in a way that allows easy comparison of values. This is usually accomplished by organizing the table into rows and columns.

Card Hint

Name

`http://lynx-json.org/card`

Meaning

A `card` hint describes a value that is a container of other values that should be interpreted as a small amount of discrete information about a subject that has more depth.

Related Hints

- `[ "card", "container" ]`

Synonyms

None

Format Rules

None

Example

```
{
  "spec": {
    "hints": [ "card", "container" ],
    "children": {
      "hints": [ "text" ]
    }
  },
  "value": [
    "Clark Griswold\n1326 West Kenneth Road\nAurora, IL 60502"
  ]
}
```

Authoring Rules

None

Authoring Considerations

Put content on a `card` when it requires more attention from the user. A card should usually contain interactive controls to enable the user to get more information about the subject of the card.

User Agent Rules

None

User Agent Considerations

The user agent should display the card in a way that emphasizes the card relative to the other content — preferably in a way that is reminiscent of a physical card.

Form Hint

Name

`http://lynx-json.org/form`

Meaning

A `form` hint describes a value that should be interpreted as a form.

Related Hints

None

Synonyms

None

Format Rules

If the value is present, it must comply with the following rules:

- MUST be an object.
- MAY have a `scope` property whose value specifies the content realm intended for display. If present, the value MUST comply with the rules defined for realm URI.
- MAY contain other properties.

Examples

```
{
  "title": "Add a Comment",
  "comment": null,
  "spec": {
    "hints": [ "form" ],
    "children": [
      {
        "name": "title",
        "hints": [ "label", "text" ]
      },
      {
        "name": "comment",
        "hints": [ "text" ],
        "input": true
      }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

A form should contain at least one value described by a `submit` hint.

User Agent Rules

None

User Agent Considerations

- The user agent must anticipate that a value described by a `form` hint may not contain visible content.
- The user agent must anticipate that a value described by a `form` hint may contain other properties that are also described by a specification.

Link Hint

Name

`http://lynx-json.org/link`

Meaning

A `link` hint describes a value that represents a hyperlink.

Related Hints

None

Synonyms

None

Format Rules

If the value is present, it must comply with the following rules:

- MUST be an object.
- MUST have an `href` property whose value is a valid URI.
- MAY have a `type` property whose value must be a valid media type name to indicate the expected media type of the content.
- MAY have a `scope` property whose value specifies the content realm intended for display. If present, the value MUST comply with the rules defined for realm URI.
- MAY contain other properties.

Examples

```
{
  "title": "Fletch",
  "href": "http://www.example.com/movies/fletch",
  "spec": {
    "hints": [ "link" ],
    "children": [
        {
            "name": "title",
            "hints": [ "label", "text" ]
        }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

While the concept of a link relationship ("rel") has been intentionally omitted from these rules, that does not mean that it has no value or that it should be omitted from authored documents. The purpose of this specification is to define the meaning of the values - not to define the relationship between the value and the resource identified by the context URI. The "rel" concept is still important and it could be included in a link object by extension.

User Agent Rules

- The user agent MUST provide the user with a control to interact with the hyperlink in order to follow it.
- If the user follows the hyperlink, then the user agent MUST fetch the target identified by the `href` using the default retrieval action for the protocol (e.g. GET for HTTP).
- The user agent SHOULD display or make the `href` accessible to the user so he/she can consider its value prior to following the hyperlink.

User Agent Considerations

- The user agent must anticipate that a value described by a `link` hint may not contain visible content.
- The user agent must anticipate that a value described by a `link` hint may contain other properties that are also described by a specification.

Submit Hint

Name

`http://lynx-json.org/submit`

Meaning

A `submit` hint describes a value that defines where and how to submit form data.

Related Hints

None

Synonyms

None

Format Rules

If the value is present, it must comply with the following rules:

- MUST be an object.
- MUST have an `action` property whose value is a valid URI.
- MAY have a `method` property indicating the protocol method used to submit the form data.
- MAY have an `enctype` property indicating how the form data is to be encoded. If not present, the default value is `application/x-www-form-urlencoded`.
- MAY have a `scope` property whose value specifies the content realm intended for display. If present, the value MUST comply with the rules defined for realm URI.
- MAY contain other properties.

Examples

```
{
  "title": "Submit Review",
  "action": "http://www.example.com/m/fletch/reviews",
  "method": "POST",
  "spec": {
    "hints": [ "submit" ],
    "children": [
        {
            "name": "title",
            "hints": [ "label", "text" ]
        }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

- The user agent MUST provide the user with a control to perform the submission.
- If the `method` property is not present, then the user agent MUST submit the form data using the default retrieval action for the protocol (e.g. GET for HTTP).
- When a `submit` control is invoked, the user agent MUST prepare a form data set, encode the form data set into a representation, and submit the representation, as defined in the Process for Submitting Form Data section.

User Agent Considerations

- The user agent must anticipate that a value described by a `submit` hint may not contain visible content.
- The user agent must anticipate that a value described by a `submit` hint may contain other properties that are also described by a specification.

Children Specification Property

Name

`http://lynx-json.org/children`

Meaning

The `children` property is used to provide a specification, or an array of specifications, to describe the values in a container value (object or array). If the `children` property is an array of specifications, then it also describes the order of the items in the container.

Format Rules

If all of the values in a container value have the same meaning and traits, then a single specification object may be used to describe them all. Otherwise, they must be described individually.

Single Specification

If present, the `children` property MUST be a specification or the URL of a specification.

Array of Specifications

- Each item MUST be a specification or the URL of a specification.
- Each specification MUST have a `name` property.

Examples

Properties of an Object

```
{
  "actor": "Chevy Chase",
  "rating": "PG",
  "title": "Fletch",
  "spec": {
    "hints": [ "container" ],
    "children": [
      {
        "name": "title",
        "hints": [ "http://example.com/movie-title", "text" ]
      },
      {
        "name": "rating",
        "hints": [ "http://example.com/mpaa-rating", "text" ]
      },
      {
        "name": "actor",
        "hints": [ "http://example.com/actor", "text" ]
      }
    ]
  }
}
```

Items in an Array with Same Meaning

```
{
  "value": [
    "G",
    "PG",
    "PG-13",
    "R"
  ],
  "spec": {
    "hints": [ "container" ],
    "children": {
      "hints": [ "http://example.com/mpaa-rating", "text" ]
    }
  }
}
```

Items in an Array with Different Meanings

```
{
  "value": [
    "Fletch"
    "PG",
    "Chevy Chase"
  ],
  "spec": {
    "hints": [ "container" ],
    "children": [
      {
        "name": "0",
        "hints": [ "http://example.com/movie-title", "text" ]
      },
      {
        "name": "1",
        "hints": [ "http://example.com/mpaa-rating", "text" ]
      },
      {
        "name": "2",
        "hints": [ "http://example.com/actor", "text" ]
      }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

If a value in a container is a value/spec pair, then the user agent MUST use the specification object of the value/spec pair instead of the specification object found in the `children` property value.

Order of Values

If the container value is an object and the `children` property is a specification, then the order of the values in the container is undefined.

If the container value is an array and the `children` property is a specification, then the order of the values in the container is their natural order.

If the `children` property is an array of specifications, then the user agent MUST use the order of the specifications in the `children` property to determine the order of the values (ignoring their natural order if the container value is an array).

User Agent Considerations

None

Labeled By Specification Property

Name

`http://lynx-json.org/labledBy`

Meaning

The `labeledBy` property of a specification identifies another value that labels the current value.

Format Rules

- The `labeledBy` property is OPTIONAL. If present, its value MUST be the property name of the value that labels the value being described by the specification.
- The property referenced by the `labeledBy` property SHOULD be described by a `label` hint.

Example

```
{
  "label": "What did you think of the movie?",
  "comments": null,
  "spec":{
    "hints": [ "form" ],
    "children": [
      {
        "name": "label",
        "hints": [ "label", "text" ]
      },
      {
        "name": "comments",
        "hints": [ "text" ],
        "input": true,
        "labeledBy": "label"
      }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

- When finding the value identified by the `labledBy` property, the user agent MUST use the process for finding values.
- If the value referenced by the `labeledBy` property is not described by a `text` hint, then the textual value of the label is the text content for the value.

User Agent Considerations

None

Input Specification Property

Name

`http://lynx-json.org/input`

Meaning

The `input` property of a specification describes a value that captures a value from the user.

Format Rules

The `input` property is OPTIONAL. If present, its value:

- MAY be `true`.
- MAY be a string to indicate the name of the form data value.
- MAY be an object with a `name` property whose value MUST be a string to indicate the name of the form data value.

Examples

Text Input

```
{
  "rating": "PG",
  "spec": {
    "hints": [ "form" ],
    "children": [
      {
        "name": "rating",
        "input": true,
        "hints": [ "text" ]
      }
    ]
  }
}
```

```
{
  "rating": "PG",
  "spec": {
    "hints": [ "form" ],
    "children": [
      {
        "name": "rating",
        "input": "mpaa-rating",
        "hints": [ "text" ]
      }
    ]
  }
}
```

```
{
  "rating": "PG",
  "spec": {
    "hints": [ "form" ],
    "children": [
      {
        "name": "rating",
        "input": {
          "name": "mpaa-rating"
        },
        "hints": [ "text" ]
      }
    ]
  }
}
```

Multi-Valued Text Input

```
{
  "scores": [],
  "spec": {
    "hints": [ "form" ],
    "children": [
      {
        "name": "scores",
        "hints": [ "container" ],
        "input": true,
        "children": {
          "hints": [ "text" ],
          "input": true
        }
      }
    ]
  }
}
```

Authoring Rules

In order to represent a multi-valued input, the specification describing the container:

- MUST have a `container` hint
- MUST have an `input` property
- MUST have a `children` property that is a specification which MUST also have an `input` property.

Authoring Considerations

None

User Agent Rules

If the container value is considered a multi-valued input, the user agent MUST provide the user with controls to add or remove values to or from the container.

User Agent Considerations

None

Validation Specification Property

Name

`http://lynx-json.org/validation`

Meaning

The `validation` property of a specification is used to indicate that a value has validation state and identify related content that should be displayed when the validation state changes. The `validation` property is also used to express constraints on an input value, or input values, and identify content that should be displayed when a constraint is, or when a set of constraints are, in a certain state.

Format Rules

Validation Constraint Set Object

The value of the `validation` property must be an object that complies with the following rules:

- MUST not have `state` property.
- MAY have any of following properties: `invalid`, `valid`, and `unknown`. If present, their value MUST be a property reference targeting the content to be displayed when the validation constraint set object is in that state.
- MAY have a `required` property. If present, its value must be a required validation constraint value.
- MAY have a `text` property. If present, its value must be a text validation constraint value.
- MAY have a `number` property. If present, its value must be a number validation constraint value.
- MAY have a `content` property. If present, its value must be a content validation constraint value.
- MAY have any other properties whose values MUST be a validation constraint object or an array of validation constraint objects.

Validation Constraint Object

The value of a validation constraint object must be an object that complies with the following rules:

- MAY have a `state` property. If present, its value MUST be `invalid`, `valid`, or `unknown`.
- MAY have any of following properties: `invalid`, `valid`, and `unknown`. If present, their value MUST be a property reference targeting the content to be displayed when the validation constraint object is in that state.
- MAY have any other property.

Examples

```
{
  "label": "First Name",
  "firstName": "Chevy",
  "requiredMessage": "The 'First Name' is required.",
  "spec": {
    "children": [
      {
        "name": "label",
        "hints": [ "text" ]
      },
      {
        "name": "firstName",
        "hints": [ "text" ],
        "input": true,
        "validation": {
          "required": {
            "state": "invalid",
            "invalid": "requiredMessage"
          }
        }
      },
      {
        "name": "requiredMessage",
        "hints": [ "text" ]
      }
    ]
  }
}
```

Authoring Considerations

When the server performs validation of user input values and responds to invalid input with a form, consider expressing the validation constraints that were violated and related `invalid` and `unknown` content to help the user resolve the issue.

If any content is targeted by a validation state property (`invalid` , `valid` , or `unknown`), its `visibility` will be determined by the user agent upon initial rendering of the document and upon change of user input values.

Authoring Rules

None

User Agent Considerations

The user agent must anticipate that unrecognized validation constraint names may be included in a validation constraint set. The user agent must respect the `state` value of those constraints and update content visibility accordingly.

User Agent Rules

For a validation constraint set object, the user agent MUST consider it to have a validation state whose value is derived from the validation state of the constraints contained within the set according to the following rules:

- if any constraints in the set have a validation state of `invalid` or if the value being described is a container with a descendent with a validation state of `invalid` , then the validation state of the set is `invalid` ;
- otherwise, if any constraints in the set have a validation state of `unknown` or if the value being described is a container with a descendent with a validation state of `unknown` , then the validation state of the set is `unknown` ;
- otherwise, if any constraints in the set have a validation state of `valid` or if the value being described is a container with a descendent with a validation state of `valid` , then the validation state of the set is `valid` ;
- otherwise, the validation state of the set is `unknown` .

For all properties of the validation constraint set object that are not already reserved and defined:

- if the value is an object the user agent MUST consider the property to be a validation constraint object, or
- if the value is an array the user agent MUST consider the property to be an array of validation constraint objects.
    - If the `state` property of a validation constraint object is not present, the user agent MUST consider the state of the validation constraint to be `unknown`.

Upon rendering a document,

- the user agent MUST honor the initial state of the validation constraint object. In other words, if the user agent has the capability to evaluate and set the `state` of a validation constraint object, it MUST NOT do so, and
- the user agent MUST update the `visibility` of content that is referenced by an `invalid`, `valid`, or `unknown` property based on the rules below.

Upon change of a user input value,

- if the user agent has the capability to evaluate and set the `state` of a validation constraint object, it MUST do so; otherwise it MUST set the `state` to `unknown`, and
- the user agent MUST update the `visibility` of content that is referenced by an `invalid`, `valid`, or `unknown` property based on the rules below.

Upon change of a validation constraint object's `state`, the user agent must reevaluate the validation state of ancestors.

Rules for updating content `visibility`:

If a property for the current validation state is present, then set the `visibility` of the content targeted by that property to `visible`. Set the `visibility` of content targeted by properties for all other validation states to `hidden`.

Required Validation Constraint Value

Name

`http://lynx-json.org/validation/required`

Meaning

The required validation constraint value expresses a required constraint for an input value.

Format Rules

A required validation constraint value MUST be a required validation constraint object that complies with the following rules:

- MUST comply with the validation constraint object rules.

Examples

```
{
  "actor": null,
  "actorRequiredError": "The value is required.",
  "spec": {
    "hints": [ "container" ],
    "children": [
      {
        "name": "actor",
        "input": true,
        "validation": {
          "required": {
            "invalid": "actorRequiredError"
          }
        }
      },
      {
        "name": "actorRequiredError",
        "hints": [ "text" ]
      }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

// TODO: remove "ignore" and decide on wording

The user agent MAY ignore the `required` property, otherwise it MUST comply with the following rules:

- If the input value is `undefined`, `null`, `""`, or `[]`, then the validation state of the constraint is `invalid`; otherwise, the validation state of the constraint is `valid`.

User Agent Considerations

None

Text Validation Constraint Value

Name

`http://lynx-json.org/validation/text`

Meaning

The text validation constraint value expresses a text constraint or an array of text constraints for an input value.

Format Rules

A text validation constraint value MUST be a text validation constraint object or an array of text validation constraint objects that complies with the following rules:

- MUST comply with the validation constraint object rules.
- MAY have a `minLength` property. If present, the value of the property MUST be a nonnegative integer.
- MAY have a `maxLength` property. If present, the value of the property MUST be a nonnegative integer.
- MAY have a `pattern` property. If present, the value of the property MUST be a string representing a valid JavaScript regular expression as defined in ECMA 262, Section 15.10.

Examples

Text Validation Constraint Object

```
  "actor": null,
  "actorPatternError": "The value must be 'Chevy Chase' or 'Bill Murray'.",
  "spec": {
    "hints": [ "container" ],
    "children": [
      {
        "name": "actor",
        "input": true,
        "validation": {
          "text": {
            "invalid": "actorPatternError",
            "pattern": "Chevy Chase|Bill Murray"
          }
        }
      },
      {
        "name": "actorPatternError",
        "hints": [ "text" ]
      }
    ]
  }
}
```

Array of Text Validation Constraint Objects

```
{
  "actor": null,
  "actorMinLengthError": "The value must be 2 or more characters.",
  "actorMaxLengthError": "The name must be 100 or fewer characters.",
  "spec": {
    "hints": [ "container" ],
    "children": [
      {
        "name": "actor",
        "input": true,
        "validation": {
          "text": [
            {
              "invalid": "actorMinLengthError",
              "minLength": 2
            },
            {
              "invalid": "actorMaxLengthError",
              "maxLength": 100
            }
          ]
        }
      },
      {
        "name": "actorMinLengthError",
        "hints": [ "text" ]
      },
      {
        "name": "actorMaxLengthError",
        "hints": [ "text" ]
      }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

// TODO: remove "ignore" and decide on wording

The user agent MAY ignore the `text` property, otherwise it MUST comply with the following rules for each text validation constraint object in its value:

- If the input value is `undefined`, `null`, or `""`, then the validation state of the constraint is `valid` and the following tests should not be performed.
- If the `minLength` property is present and its value is greater than the length of the input value, then the validation state of the constraint is `invalid`.
- If the `maxLength` property is present and its value is less than the length of the input value, then the validation state of the constraint is `invalid`.
- If the `pattern` property is present, the user agent MUST compile the regular expression specified in its value with the global, ignoreCase, and multiline options disabled and test the input value for a match. If the pattern does not start with '^', it is implied. If the pattern does not end with '$', it is implied. If the input value does not match the regular expression, then the validation state of the constraint is `invalid`.
- If the previous tests have not determined the validation state of the constraint to be `invalid`, then the validation state of the constraint is `valid`.

User Agent Considerations

None

Number Validation Constraint Value

Name

`http://lynx-json.org/validation/number`

Meaning

The number validation constraint value expresses a numerical constraint or an array of numerical constraints for an input value.

Format Rules

A number validation constraint value MUST be a number validation constraint object or an array of number validation constraint objects that complies with the following rules:

- MUST comply with the validation constraint object rules.
- MAY have a `min` property. If present, the value MUST be a number.
- MAY have a `max` property. If present, the value MUST be a number.
- MAY have a `step` property. If present, the value MUST be a positive number.

Examples

Number Validation Constraint Object

```
{
  "rating": null,
  "spec": {
    "hints": [ "container" ],
    "children": [
      {
        "name": "rating",
        "input": true,
        "validation": {
          "number": {
            "invalid": "ratingError",
            "min": 1,
            "max": 5
          }
        }
      }
    ]
  }
}
```

Array of Number Validation Constraint Objects

```
{
  "rating": null,
  "ratingMinError": "The value must be 1 or more.",
  "ratingMaxError": "The value must be 5 or less.",
  "spec": {
    "hints": [ "container" ],
    "children": [
      {
        "name": "rating",
        "input": true,
        "validation": {
          "number": [
            {
              "invalid": "ratingMinError",
              "min": 1
            },
            {
              "invalid": "ratingMaxError",
              "max": 5
            }
          ]
        }
      },
      {
        "name": "ratingMinError",
        "hints": [ "text" ]
      },
      {
        "name": "ratingMaxError",
        "hints": [ "text" ]
      }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

// TODO: remove "ignore" and decide on wording

The user agent MAY ignore the `number` property, otherwise it MUST comply with the following rules for each number validation constraint object in its value:

- If the input value is `undefined`, `null`, or `""`, then the validation state of the constraint is `valid` and the following tests should not be performed.
- If the input value is not numeric, then the validation state of the constraint is `invalid`.
- If the `min` property is present and its value is greater than the input value, then the validation state of the constraint is `invalid`.
- If the `max` property is present and its value is less than the input value, then the input fails validation.
- If the `step` property is present and the input value is not evenly divisible by the `step` value, then the validation state of the constraint is `invalid`.
- If the previous tests have not determined the validation state of the constraint to be `invalid`, then the validation state of the constraint is `valid`.

User Agent Considerations

None

Content Validation Constraint Value

Name

`http://lynx-json.org/validation/content`

Meaning

The content validation constraint value expresses a content constraint or an array of content constraints for an input value.

Format Rules

A content validation constraint value MUST be a content validation constraint object or an array of content validation constraint objects that complies with the following rules:

- MUST comply with the validation constraint object rules.
- MAY have a `type` property. If present, the value of the property MUST be a string or an array of strings matching one of the following:
  - A file extension (e.g., `.jpg`, `.png`, `.doc`).
  - A media type range (e.g., `image/png`, `image/*`).
- MAY have a `maxLength` property representing the maximum length of the content in bytes. If present, the value of the property MUST be a nonnegative integer.

Examples

```
{
  "movie": null,
  "movieTypeAndLengthError": "Please select a video that is 4MB or less in size.",
  "spec": {
    "hints": [ "container" ],
    "children": [
      {
        "name": "movie",
        "hints": [ "content" ],
        "input": true,
        "validation": {
          "content": {
            "invalid": "movieTypeAndLengthMessage",
            "type": "video/*",
            "maxLength": 4000000
          }
        }
      },
      {
        "name": "movieTypeAndLengthError",
        "hints": [ "text" ]
      }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

// TODO: remove "ignore" and decide on wording

The user agent MAY ignore the `content` property, otherwise it MUST comply with the following rules for each content validation constraint object in its value:

- If the value does not match one or more of the values of the `type` property, then

the validation state of the constraint is `invalid`.

- If the length of the value in bytes exceeds the value of the `maxLength` property, then the validation state of the constraint is `invalid`.
- If the previous tests have not determined the validation state of the constraint to be `invalid`, then the validation state of the constraint is `valid`.

User Agent Considerations

When providing a control to the user to supply a value, the user agent may choose to assist the user in providing valid input. If the value is described by a hint of `content`, the user agent may choose to make items that match the `type` and/or `maxLength` properties more prominent.

Follow Specification Property

Name

`http://lynx-json.org/follow`

Meaning

The `follow` property of a specification describes a hyperlink that the user agent should follow on behalf of the user.

Format Rules

The `follow` property is OPTIONAL. If present, its value MUST be a positive number representing milliseconds.

Examples

```
{
  "title": "Preparing movie for viewing. Please wait...",
  "href": "http://www.example.com/movies/fletch/watch",
  "spec": {
    "hints": [ "link" ],
    "children": [
      {
        "name": "title",
        "hints": [ "label", "text" ]
      }
    ],
    "follow": 5000
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

After displaying the document containing the value described by the `follow` property, the user agent MUST wait the specified number of milliseconds and then MUST follow the hyperlink.

User Agent Considerations

None

Send Specification Property

Name

`http://lynx-json.org/send`

Meaning

The `send` property of a specification describes a submit control that the user agent should invoke on behalf of the user.

Format Rules

The `send` property is OPTIONAL. If present, its value MUST be `"change"`.

Examples

```
{
  "title": "Add a Comment",
  "comment": null,
  "postComment": {
    "title": "Submit",
    "action": "/comments",
    "method": "POST"
  },
  "spec": {
    "hints": [ "form" ],
    "children": [
      {
        "name": "title",
        "hints": [ "label", "text" ]
      },
      {
        "name": "comment",
        "hints": [ "text" ],
        "input": true
      },
      {
        "name": "postComment",
        "hints": [ "submit" ],
        "send": "change",
        "labeledBy": "title",
        "children": [
          {
            "name": "title",
            "hints": [ "label", "text" ]
          }
        ]
      }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

After displaying the document containing the value described by the `send` property, the user agent MUST submit the related form data when the value of an input changes if the validation state of the form is not invalid.

User Agent Considerations

None

Submitter Specification Property

Name

`http://lynx-json.org/submitter`

Meaning

The `submitter` property of a specification identifies another value that is the submitter for the element.

Format Rules

- The `submitter` property is OPTIONAL. If present, its value MUST be the property name of the value that submits form data for the value being described by the specification.
- The property referenced by the `submitter` property SHOULD be described by a `submit` hint.

Example

```
{
  "label": "What did you think of the movie?",
  "comments": null,
  "dislike": {
    "method": "POST",
    "action": "/dislike/comments"
  },
  "like": {
    "method": "POST",
    "action": "/like/comments"
  },
  "spec":{
    "hints": [ "form" ],
    "children": [
      {
        "name": "label",
        "hints": [ "label", "text" ]
      },
      {
        "name": "comments",
        "hints": [ "text" ],
        "input": true,
        "labeledBy": "label",
        "submitter": "like"
      },
      {
        "name": "dislike",
        "hints": [ "submit" ]
      },
      {
        "name": "like",
        "hints": [ "submit" ]
      }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

- When finding the value identified by the `submitter` property, the user agent MUST use the process for finding values and invoke the value's control.

User Agent Considerations

None

Options Specification Property

Name

`http://lynx-json.org/options`

Meaning

The `options` property of a specification identifies another value that provides options for the current input value.

Format Rules

The `options` property is OPTIONAL. If present, its value MUST be the property name of the value that provides options for the value being described by the specification.

Examples

```
{
  "favoriteMovie": null,
  "listOfMovies": [
    "Fletch",
    "Caddyshack",
    "Christmas Vacation"
  ],
  "spec": {
    "hints": [ "form" ],
    "children": [
      {
        "name": "favoriteMovie",
        "visibility": "hidden",
        "hints": [ "http://example.com/movie-title", "text" ],
        "input": true,
        "options": "listOfMovies"
      },
      {
        "name": "listOfMovies",
        "hints": [ "container" ],
        "children": {
          "hints": [ "http://example.com/movie-title", "text" ],
          "option": true
        }
      }
    ]
  }
}
```

```
{
  "favoriteMovie": null,
  "listOfMovies": [
    { "title": "Fletch", "id": "1" },
    { "title": "Caddyshack", "id": "2" },
    { "title": "Christmas Vacation", "id": "3" }
  ],
  "spec": {
    "hints": [ "form" ],
    "children": [
      {
        "name": "favoriteMovie",
        "visibility": "hidden",
        "hints": [ "http://example.com/movie-id", "text" ],
        "input": true,
        "options": "listOfMovies"
      },
      {
        "name": "listOfMovies",
        "hints": [ "container" ],
        "children": {
          "hints": [ "container" ],
          "children": [
            {
              "name": "title",
              "hints": [ "text" ]
            },
            {
              "name": "id",
              "hints": [ "http://example.com/movie-id", "text" ],
              "visibility": "hidden"
            }
          ],
          "option": true
        }
      }
    ]
  }
}
```

```
{
  "favoriteMovie": null,
  "listOfMovies": {
    "src": "http://example.com/movies/fav/options",
    "type": "application/lynx+json"
  },
  "spec": {
    "hints": [ "form" ],
    "children": [
      {
        "name": "favoriteMovie",
        "visibility": "hidden",
        "hints": [ "http://example.com/movie-title", "text" ],
        "input": true,
        "options": "listOfMovies"
      },
      {
        "name": "listOfMovies",
        "hints": [ "content" ]
      }
    ]
  }
}
```

Multi-Valued Text Input with Options

```
{
  "favoriteMovies": null,
  "listOfMovies": {
    "src": "http://example.com/movies/fav/options",
    "type": "application/lynx+json"
  },
  "spec": {
    "hints": [ "form" ],
    "children": [
      {
        "name": "favoriteMovies",
        "visibility": "hidden",
        "input": true,
        "options": "listOfMovies",
        "children": {
          "hints": [ "http://example.com/movie-title", "text" ],
          "input": true
        }
      },
      {
        "name": "listOfMovies",
        "hints": [ "content" ]
      }
    ]
  }
}
```

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

An "option" is defined as a value that is described by a hint matching the most specific hint describing the input value.

- When finding the value identified by the `options` property, the user agent MUST use the process for finding values.

- For the value identified by the `options` property, the user agent MUST identify the options in the value (including the value itself).
- The user agent MUST provide the user with a control to select and deselect each identified option. Starting from the option and then its ancestors, the first value that is described by a specification with an `option` property MUST be the control.
- For multi-valued inputs, selecting an option adds the value to the input container and deselecting an option removes the value from the input container.

User Agent Considerations

None

Option Specification Property

Name

`http://lynx-json.org/option`

Meaning

The `option` property of a specification describes a value that represents an option for an `input` value.

Format Rules

The `option` property is OPTIONAL. If present, its value MUST be `true`.

Examples

See the examples in the `options` specification property section.

Authoring Rules

None

Authoring Considerations

None

User Agent Rules

None

User Agent Considerations

None

Realm URI

A realm is a field or domain of activity or interest that may be used to organize information (see Section 6.5.4 "Design of Media Types" in Fielding Dissertation).

A realm URI is used to identify and refer to a realm. By disclosing a realm, the server provides additional information to user agents, or other components, that they may use to apply special treatment to the content within the realm.

Format Rules

MUST be an absolute URI.

Normalization and Comparison

Given realm URI A and realm URI B, if A starts with B (using a simple string comparison after URI normalization as defined in Section 6 "Normalization and Comparison" in RFC 3986) then the realm identified by A is considered to be within the realm identified by B.

User Agent Processes

This section defines the processes that the user agent must support in order to properly create interactive interfaces for Lynx content.

Process for Finding Values

Some specification properties are used to target other values in the document including `labeledBy` and `options`. The value of these types of properties is the name of another value in the document. To find the value, use the following steps:

1. Let "origin" equal the value that is targeting another value and let "target" be the name of the target.
2. Iterate over the origin and its descendants, via depth-first traversal, and select the first value with a `name` that exactly matches the target.
3. If no value has been found but the origin has a parent, let origin equal the parent and repeat step 2 until a value is found or the origin has no remaining ancestors.

Process for Displaying Content

When a new document is retrieved as a result of the user's interaction with a `link` or `submit` in a document, the user agent MUST attempt to select an object whose view should be updated with a view of the new document according to the following steps:

1. Let "origin" equal the `link`, `submit`, or `content` object that is the source of the transition.
2. Iterate over the origin and its descendants, via depth-first traversal, and select the first object with a `scope` that includes the realm of the document, according to the "Normalization and Comparison" rules for realm uri.
3. If no object has been found but the origin has a parent, let origin equal the parent and repeat step 2 until an object is found or the origin has no remaining ancestors.

The user agent MUST update the view for the selected object to display the new document to the user. If no object is selected, then the user agent MUST update the root view.

Process for Submitting Form Data

When a submit control is invoked, the user agent MUST prepare a form data set, encode the form data set into a representation, and submit the representation.

Preparing a Form Data Set

To prepare a form data set:

- Let *form data set* be an empty set.
- Let *form* be the nearest ancestor of the submit control's value, if it exists.
- If *form* does not exist, stop preparing a form data set.
- Let *inputs* be the descendant values of the *form's* value (in depth-first order) that are described by an `input` specification property.

For each *input* in *inputs*:

- Let *result* be a tuple having *name* and *value* components.
  - Let *name* be the value of the *input* specification property's `name` property, if it exists. Otherwise, let *name* be the `name` of the input value, if it exists. Otherwise, let *name* be the `name` of the input value's first named ancestor.
  - Let *value* be a tuple having *data* and *type* components.
    - For `text` inputs, let *data* be the value's JSON value (as defined in RFC 4627, Section 2.1) if the value is not `null` or `undefined`; otherwise, let *data* be an empty string. Let *type* be an empty string.
    - For `content` inputs, let *data* and *type* be the value of the `content` value's `data` and `type` properties.
- Append *result* to *form data set*.

Encoding a Form Data Set

URL Encoding

When the value of the `submit` value's `enctype` property is `application/x-www-form-urlencoded`, the form data set MUST be encoded as follows:

- Let *result* be an empty string.

- For each *tuple* in the *form data set*:
  - Encode the *name* component according to URL Character Encoding and append to *result*.
  - Append the character '=' (U+003D) to *result*.
  - If the *value* component's *type* is an empty string, encode its *data* component according to URL Character Encoding and append to *result*; otherwise, continue to the next step.
  - If the current *tuple* is not the last, append the character '&' (U+0026) to *result*.
- Let the *content type* of *result* be `application/x-www-form-urlencoded`.

URL Character Encoding

Encode all characters that are not in the "unreserved" set of RFC 3986, Section 2.3, according to the rules of percent encoding documented in RFC 3986, Sections 2.1 and 2.4.

Multipart Form Encoding

When the value of the `submit` value's `enctype` property is `multipart/form-data`, the form data set MUST be encoded according to the rules of RFC 2388 as follows:

- This section uses the concepts *boundary*, *field*, *field name*, *field value*, and *field content type* as defined in RFC 2388, Section 4.1. Each *tuple* in the *form data set* is correlated to these concepts as follows:
  - The *name* component correlates to *field name*. The *field name* MUST be encoded as UTF-8.
  - The *data* component of the *value* component correlates to *field value*.
  - The *type* component of the *value* component correlates to *field content type*. If the *field content type* is an empty string, the resulting part MUST NOT have a Content-Type header specified, and the *field value* MUST be encoded as UTF-8.
- Let *result* be an empty sequence of bytes.
- For each *tuple* in the *form data set*, encode and append each *tuple* to *result* according to the rules above and RFC 2388, Section 4.1.
- Let the *content type* of *result* be `multipart/form-data; boundary={boundary}`, where `{boundary}` is the value of *boundary*.

Other Encodings

Other encodings may be supported through extensions to this specification. If the user agent does not understand the `submit` value's `enctype`, it MUST encode the form data set using Multipart Form Encoding.

Submitting Encoded Data

HTTP and HTTPS

If the method is `GET`, `DELETE`, or any other method that does not support an entity-body, submit the encoded data as follows:

- Let *result* be the encoded form data set.
- Let *method* be the `submit` value's `method` property value.
- Let *URL* be the submit value's `action` property value.
- Let the *query component* of the *URL* be *result*.
- Send an HTTP message, let its method be *method*, and let its retrieval URL be *URL*.

If the method is `POST`, `PUT`, or another method that supports an entity-body, submit the encoded data as follows:

- Let *result* be the encoded form data set.
- Let *method* be the `submit` value's `method` property value.
- Let *URL* be the submit value's `action` property.
- Let the URL's *query component* be empty.
- Send an HTTP message, let its method be *method*, let its retrieval URL be *URL*, let its "Content-Type" header be the content type of *result*, and let its entity body be *result*.

Other Schemes

For other schemes, user agents should act in a manner analogous to that defined in HTML, Section 4.10.22.3.

References

RFC 2119

"Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997

RFC 4627

"The application/json Media Type for JavaScript Object Notation (JSON)", D. Crockford, July 2006

RFC 3986

"Uniform Resource Identifier (URI): Generic Syntax", T. Berners-Lee, January 2005

RFC 6838

"Media Type Specifications and Registration Procedures", N.Freed, J. Klensin, and T.Hansen, January 2013

ECMA 262

ECMA, "ECMAScript® Language Specification", June 2011

RFC 2616

"Hypertext Transfer Protocol -- HTTP/1.1", R. Fielding, J. Gettys, J. Mogul, H. Frystyk Nielsen, L. Masinter, P. Leach and T. Berners-Lee, June 1999.

Fielding Dissertation

"Architectual Styles and the Design of Network-based Software Architectures" (https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf)", R. Fielding, 2000.

HTML

"HTML 5: A vocabulary and associated APIs for HTML and XHTML", W3C, October 2014