Hydrilla source package format

This page explains the format of source packages used to prepare Hydrilla custom site resources.

You might also want to read about the data format of built Hydrilla resources.

How Hydrilla packages are built

Hydrilla package builder expects a source package directory and a destination directory to be specified on the command line. If source package directory is missing, current directory is used as default.

Desination directory

The destination directory can be considered the same as Hydrilla's "malcontent" directory - built artifacts will be written in conformance with Hydrilla data format.

Source package directory

Source package directory is expected to contain an index.json file (although Hydrilla builder can be told to use another file and behave as if it was index.json). Hydrilla builder loads it (smartly ignoring "//" comments in it) and collects the definitions of site resources and pattern->payload mappings in it.

Format of an index.json

Example

To understand the format, look into this example file with explanatory comments in it:

```
// SPDX-License-Identifier: CC0-1.0

// Copyright (C) 2021, 2022 Wojtek Kosior <koszko@koszko.org>
// Available under the terms of Creative Commons Zero v1.0 Universal.

// This is an example index.json file describing Hydrilla packages. As you can
// see, for storing this information Hydrilla utilizes JSON with an additional
// extension in the form of '//' comments support.

// An index.json file conveys definitions of site resources and pattern->payload
// mappings. The definitions may reference files under index.json's containing
// directory, using relative paths. This is how scripts, license texts, etc. are
// included.
// File reference always takes the form of an object with "file" property
// specifying path to the file. In certain contexts additional properties may be
// allowed or required. Unix paths (using '/' as separator) are assumed. It is
// not allowed for an index.json file to reference files outside its directory.

// Certain objects are allowed to contain a "comment" field. Although '//'
// comments can be used in index.json files, they will not be included in
// generated JSON definitions. If a comment should be included in the
// definitions served by Hydrilla API, it should be put in a "comment" field of
// the proper object.

// Unknown object properties will be ignored. This is for compatibility with
// possible future revisions of the format.
{
    // Once our index.json schema changes, this field's value will change. Our
    // software will be able to handle both current and older formats thanks to
    // this information present in every index.json file. Schemas that differ by
    // the first (major) number are always incompatible (e.g. a Hydrilla builder
    // instance released at the time of 1.2 being the most recent schema version
    // will not understand version 2.1).
    // Schemas that are backwards-compatible will have the same major number
    // and might differ by the second (minor) version number. The third (patch)
    // and subsequent numbers are being ignored right now.
    "$schema": "https://hydrilla.koszko.org/schemas/package_source-1.schema.json",

    // Used when referring to this source package. Should be consize, unique
    // (among other source package names) and can only use a restricted set of
    // characters. It has to match: [-0-9a-z.]+
    "source_name": "hello",

    // This property lists files that contain copyright information regarding
    // this source package as well as texts of licenses used. Although no
    // specific format of these files is mandated, it is recommended to make
    // each source package REUSE-compliant, generate an spdx report for it as
    // `report.spdx` and list this report together with all license files here.
    "copyright":  [
        {"file": "report.spdx"},
        {"file": "LICENSES/CC0-1.0.txt"}
    ],

    // Where this software/work initially comes from.
    "upstream_url": "https://git.koszko.org/hydrilla-source-package-example",

    // Additional "comment" field can be used if needed.
    // "comment": ""

    // List of actual site resources and pattern->payload mappings. Each of them
    // is represented by an object. Meta-sites and replacement site interfaces
    // will also belong here once they get implemented.
    "definitions": [
        {
            // Value of "type" can currently be one of: "resource" and
            // "mapping". The one we have here, "resource", defines a list
            // of injectable scripts that can be used as a payload or as a
            // dependency of another "resource". In the future CSS style sheets
            // and WASM modules will also be composite parts of a "resource" as
            // scripts are now.
            "type": "resource",

            // Used when referring to this resource in "dependencies" list of
            // another resource or in "payload" field of a mapping. Should
            // be consize and can only use a restricted set of characters. It
            // has to match: [-0-9a-z]+
            "identifier": "helloapple",

            // "long_name" should be used to specify a user-friendly alternative
            // to an identifier. It should generally not collide with a long
            // name of some resource with a different uuid and also shouldn't
            // change in-between versions of the same resource, although
            // exceptions to both rules might be considered. Long name is
            // allowed to contain arbitrary unicode characters (within reason!).
            "long_name": "Hello Apple",

            // Item definitions contain version information. Version is
            // represented as an array of integers, with major version number
            // being the first array item. In case of resources, version is
            // accompanied by a revision field which contains a positive
            // integer. If versions specified by arrays of different length need
            // to be compared, the shorter array gets padded with zeroes on the
            // right. This means that for example version 1.3 could be given as
            // both [1, 3] and [1, 3, 0, 0] (aka 1.3.0.0) and either would mean
            // the same.
            // Different versions (e.g. 1.0 and 1.3) of the same resource can be
            // defined in separate index.json files. This makes it easy to
            // accidently cause an identifier clash. To help detect it, we
            // require that each resource has a uuid associated with it. Attempt
            // to define multiple resources with the same identifier and
            // different uuids will result in an error being reported. Defining
            // multiple resources with different identifiers and the same uuid
            // is disallowed for now (it may be later permitted if we consider
            // it good for some use-case).
            "uuid": "a6754dcb-58d8-4b7a-a245-24fd7ad4cd68",

            // Version should match the upstream version of the resource (e.g. a
            // version of JavaScript library). Revision number starts as 1 for
            // each new resource version and gets incremented by 1 each time a
            // modification to the packaging of this version is done. Hydrilla
            // will allow multiple definitions of the same resource to load, as
            // long as their versions differ. Thanks to the "version" and
            // "revision" fields, clients will know they have to update certain
            // resource after it has been updated. If multiple definitions of
            // the same version of given resource are provided, an error is
            // generated (even if those definitions differ by revision number).
            "version": [2021, 11, 10],
            "revision": 1,

            // A short, meaningful description of what the resource is and/or
            // what it does.
            "description": "greets an apple",

            // If needed, a "comment" field can be added to provide some
            // additional information.
            // "comment": "this resource something something",

            // Resource's "dependencies" array shall contain names of other
            // resources that (in case of scripts at least) should get evaluated
            // on a page before this resource's own scripts.
            "dependencies": [{"identifier": "hello-message"}],

            // Array of JavaScript files that belong to this resource.
            "scripts": [
                {"file": "hello.js"},
                {"file":   "bye.js"}
            ]
        }, {
            "type":        "resource",
            "identifier":  "hello-message",
            "long_name":   "Hello Message",
            "uuid":        "1ec36229-298c-4b35-8105-c4f2e1b9811e",
            "version":     [2021, 11, 10],
            "revision":    2,
            "description": "define messages for saying hello and bye",
            // If "dependencies" is empty, it can also be omitted.
            // "dependencies": [],
            "scripts": [{"file": "message.js"}]
        }, {
            "type": "mapping",

            // Has similar function to resource's identifier. Should be consize
            // and can only use a restricted set of characters. It has to match:
            // [-0-9a-z]+
            // It can be the same as some resource identifier (those are
            // different entities and are treated separately).
            "identifier": "helloapple",

            // "long name" and "uuid" have the same meaning as in the case of
            // resources. Uuids of a resource and a mapping can technically be
            // the same, but it is recommended to avoid even this kind of
            // repetition.
            "long_name": "Hello Apple",
            "uuid": "54d23bba-472e-42f5-9194-eaa24c0e3ee7",

            // "version" differs from its counterpart in resource in that it has
            // no accompanying revision number.
            "version": [2021, 11, 10],

            // A short, meaningful description of what the mapping does.
            "description": "causes apple to get greeted on Hydrillabugs issue tracker",

            // A comment, if necessary.
            // "comment": "blah blah because bleh"

            // The "payloads" object specifies which payloads are to be applied
            // to which URLs.
            "payloads": {
                // Each key should be a valid Haketilo URL pattern.
                "https://hydrillabugs.koszko.org/***": {
                    // Should be the name of an existing resource. The resource
                    // may, but doesn't have to, be defined in the same
                    // index.json file.
                    "identifier": "helloapple"
                },
                // More associations may follow.
                "https://hachettebugs.koszko.org/***": {
                    "identifier": "helloapple"
                }
            }
        }
    ],
    // We can also list additional files to include in the produced source
    // archive. Hydrilla builder will then include those together with all
    // script and copyright files used.
    "additional_files": [
        {"file": "README.txt"},
        {"file": "README.txt.license"},
        {"file": ".reuse/dep5"}
    ],
    // We can optionally tell Hydrilla builder to run the REUSE tool to generate
    // report.spdx file. Using this option requires REUSE to be installed and
    // and importable in the Python virtualenv used by Hydrilla builder.
    "reuse_generate_spdx_report": true
}
```

JSON schema

Valid index.json file should conform to the schema available here.