Configuration

Introduction

Many aspects of NOMAD and its operation can be modified through configuration. Most configuration items have reasonable defaults and typically only a small subset has to be
overwritten.

Configuration items get their value in the following order:

1. The item is read from the environment. This has the highest priority and will overwrite
values in a `nomad.yaml` file or the NOMAD source-code.
2. The value is given in a `nomad.yaml` configuration file.
3. There is no custom value, and the value hard-coded in the NOMAD sources will be used.

Configuration items are structured. The configuration is hierarchical and items are aggregated
in potentially nested section. For example the configuration item `services.api_host` denotes
the attribute `api_host` in the configuration section `services`.

Setting values from the environment

NOMAD services will look at the environment.

All environment variables starting with `NOMAD_` are considered. The rest of the name is interpreted as a configuration item. Sections and attributes are concatenated with a `_`.

For example, the environment variable `NOMAD_SERVICES_API_HOST` will set the value for
the `api_host` attribute in the `services` section.

Setting values from a `nomad.yaml`

NOMAD services will look for a `nomad.yaml` file. By default, they will look in the current working directory. This location can be overwritten with the `NOMAD_CONFIG` environment

variable.

The configuration sections and attributes can be denoted with YAML objects and attributes.

Here is an example `nomad.yaml` file:

```yaml
--8<-- "ops/docker-compose/nomad-oasis/configs/nomad.yaml"
```

When overwriting an *object* in the configuration, the new value will be merged with the default value. The new merged object will have all of the attributes of the new object in addition to any old attributes that were not overwritten. This allows you to simply change an individual setting without having to provide the entire structure again, which simplifies customization that happens deep in the configuration hierarchy. When overwriting anything else (numbers, strings, lists etc.) the new value completely replaces the old one.

User interface customization

Many of the UI options use a data model that contains the following three fields: `include`, `exclude` and `options`. This structure allows you to easily disable, enable, reorder and modify the UI layout with minimal config rewrite. Here are examples of common customization tasks using the search columns as an example:

Disable item:

```yaml
ui:
```

```yaml
  apps:
    options:
      entries:
        columns:
          exclude: ['upload_create_time']
```

Explicitly select the shown items and their order

```yaml
ui:
  apps:
    options:
      entries:
        columns:
          include: ['entry_id', 'upload_create_time']
```

Modify existing option

```yaml
ui:
  apps:
    options:
      entries:
        columns:
          options:
            upload_create_time:
              label: "Uploaded"
```

Add a new item that does not yet exist in options. Note that by default all options are shown in the order they have been declared unless the order is explicitly given in `include`.

```yaml
ui:
  apps:
    options:
      entries:
        columns:
          options:
            upload_id:
              label: "Upload ID"
```

The following is a reference of all configuration sections and attributes.

Services

{{ config_models(['services', 'meta', 'oasis', 'north']) }}

Files, databases, external services

{{ config_models(['fs', 'mongo', 'elastic', 'rabbitmq', 'keycloak', 'logstash', 'datacite', 'rfc3161_timestamp', 'mail'])}}

Processing

{{ config_models(['process', 'reprocess', 'bundle_export', 'bundle_import', 'normalize', 'celery', 'archive'])}}

User Interface

These settings affect the behaviour of the user interface. Note that the preferred way for creating custom apps is by using app plugin entry points (../howto/plugins/apps.md).

{{ config_models(['ui'])}}

Others

{{ config_models() }}