stix2-validator Documentation

Release 3.0.0

OASIS Open

Apr 29, 2021

Contents:

The STIX Validator checks that STIX JSON content conforms to the requirements specified in the latest STIX 2 specifications. In addition to checking conformance with the JSON schemas, the validator checks conformance with requirements that cannot be specified in JSON schema, as well as with established "best practices." This validator is non-normative; in cases of conflict with the STIX specification, the specification takes precedence.

The STIX 2 specification contains two types of requirements: mandatory "MUST" requirements, and recommended "SHOULD" best practice requirements. The validator checks documents against the "MUST" requirements using JSON schemas. Some of these mandatory requirements cannot be implemented in JSON Schema, however, so the validator uses Python functions to check them. The "SHOULD" requirements are all checked by Python functions, and options may be used to ignore some or all of these recommended "best practices."

The STIX Validator uses the stix2-patterns validator to check that Indicator patterns conform to the STIX Patterning language and only reference properties valid for the objects in the pattern.

The validator also color-codes its output to make it easier to tell at a glance whether validation passed.

CHAPTER 1

Installation

Note: The STIX 2 validator requires Python 2.7 or 3.4+.

The easiest way to install the STIX 2 validator is with pip:

```
$ pip install stix2-validator
```

Note that if you instead install it by cloning or downloading the repository, you will need to set up the submodules before you install it:

```
$ git clone https://github.com/oasis-open/cti-stix-validator.git
$ cd cti-stix-validator/
$ git submodule update --init --recursive
$ python setup.py install
```

CHAPTER 2

Usage

2.1 As A Script

The validator comes with a bundled script which you can use to validate a JSON file containing STIX content:

```
$ stix2_validator <stix_file.json>
```

2.2 As A Library

You can also use this library to integrate STIX validation into your own tools. You can validate a JSON file:

```
from stix2validator import validate_file, print_results

results = validate_file("stix_file.json")
print_results(results)
```

You can also validate a JSON string, and check if the input passed validation:

```
from stix2validator import validate_string, print_results

stix_json_string = "..."
results = validate_string(stix_json_string)
if results.is_valid:
    print_results(results)
```

If your STIX is already in a Python dictionary (for example if you have already run `json.loads()`), use `validate_instance()` instead:

```
import json
from stix2validator import validate_instance, print_results

```

(continues on next page)

(continued from previous page)

```
stix_json_string = "..."
stix_obj = json.loads(stix_json_string)
results = validate_instance(stix_obj)
if results.is_valid:
    print_results(results)
```

You can pass a ValidationOptions object into `validate_file()`, `validate_string()`, or `validate_instance()` if you want behavior other than the default:

```
from stix2validator import ValidationOptions

options = ValidationOptions(strict=True)
results = validate_string(stix_json_string, options)
```

2.3 STIX 2 Versions

By default the validator will check content against the latest version of the STIX 2 specification. However, older versions can be checked with the `version` option. For example:

```
$ stix2_validator --version=2.0 <stix_file.json>
```

or in Python:

```
options = ValidationOptions(strict=True, version="2.0")
results = validate_string(stix_json_string, options)
```

2.4 Additional Schemas

The validator uses the STIX 2 JSON schemas as the basis for its validation, but you can also validate with your own additional schemas. This can help if you want to validate STIX content using custom objects, properties, or observables.

To do this use the `--schema-dir` argument:

```
$ stix2_validator --schema-dir /path/to/my/schemas <stix_file.json>
```

or in Python:

```
from stix2validator import ValidationOptions

options = ValidationOptions(strict=True, version="2.0", schema_dir="/path/to/custom/
↪schemas")
results = validate_file("stix_file.json")
print_results(results)
```

You can see some examples of custom schemas here. Note that if you want to add a custom property to an existing object type, your custom schema only needs to contain that property; the validator's built-in schemas are still checked against and will handle the rest.

CHAPTER 3

Options

These are the different options that can be set, whether the validator is used as a command-line script or as a Python library. When using the validator as a library, these options can be passed as parameters to the `ValidationOptions` constructor.

Script	Library	Description
`FILES`	`files`	A whitespace separated list of STIX files or directories of STIX files to validate.
`-r, --recursive`	`recursive`	Recursively descend into input directories.
`-s SCHEMA_DIR, --schemas SCHEMA_DIR`	`schema_dir`	Custom schema directory. If provided, input will be validated against these schemas in addition to the STIX schemas bundled with this script.
`--version`	`version`	The version of the STIX specification to validate against (e.g. "2.0").
`-v, --verbose`	`verbose`	Print informational notes and more verbose error messages.
`-q, --silent`	`silent`	Silence all output to stdout.
`-d DISABLED, --disable DISABLED, --ignore DISABLED`	`disabled`	A comma-separated list of recommended best practice checks to skip. By default, no checks are disabled. Example: –disable 202,210
`-e ENABLED, --enable ENABLED, --select ENABLED`	`enabled`	A comma-separated list of recommended best practice checks to enable. If the –disable option is not used, no other checks will be run. By default, all checks are enabled. Example: –enable 218
`--strict`	`strict`	Treat warnings as errors and fail validation if any are found.
`--strict-types`	`strict_types`	Ensure that no custom object types are used, only those defined in the STIX specification.
`--strict-properties`	`strict_properties`	Ensure that no custom properties are used, only those defined in the STIX specification.
`--no-cache`	`no_cache`	Disable the caching of external source values.
`--refresh-cache`	`refresh_cache`	Clears the cache of external source values, then during validation downloads them again.
`--clear-cache`	`clear_cache`	Clear the cache of external source values after validation.
`--enforce-refs`	`enforce_refs`	Ensures that all SDOs being referenced by SROs are contained within the same bundle.

For the list of checks that can be used with the “enabled” or “disabled” options, see the *Best Practices page*.

CHAPTER 4

Checking STIX Content

The validator will always validate input against all of the mandatory "MUST" requirements from the spec. By default it will issue warnings if the input fails any of the "SHOULD" recommendations, but validation will still pass. To turn these "best practice" warnings into errors and cause validation to fail, use the `--strict` option with the command-line script, or create a ValidationOptions object with `strict=True` when using the library.

You cannot select which of the "MUST" requirement checks will be performed; all of them will always be performed. However, you may select which of the "SHOULD" checks to perform. Use the codes from the table below to enable or disable these checks. For example, to disable the checks for the report label and tool label vocabularies, use `--disable 218,222` or `disabled="218,222"`. All the other checks will still be performed. Conversely, to only check that custom property names adhere to the recommended format but not run any of the other "best practice" checks, use `--enable 103` or `enabled="103"`.

Enabling supersedes disabling. Simultaneously enabling and disabling the same check will result in the validator performing that check.

Some checks access Internet resources to determine valid values for certain properties. For instance, the 'mime-type' check accesses the IANA's list of registered MIME types. These web requests are cached to conserve bandwidth, will expire after one week, and are stored in a file called 'cache.sqlite' in the same directory the script is run from. The cache can be refreshed manually with the `--refresh-cache` or `refresh_cache=True`, or cleared with `--clear-cache` or `clear_cache=True`. This caching can be disabled entirely with `--no-cache` or `no_cache=True`.

4.1 Mandatory Checks - STIX 2.1

Name	Ensures…	Errors/Warnings
timestamp	timestamps contain sane months, days, hours, minutes, seconds	'<property>': '<timestamp>' is not a valid timestamp: <message> '<object>': '<property>': '<timestamp>' is not a valid timestamp: <message> '<object>': '<extension>': '<property>': '<timestamp>' is not a valid timestamp: <message> '<object>': '<property>': '<embedded-property>' is not a valid timestamp: <message>
timestamp_compare	timestamp properties with a comparison are valid	'<operand_1>' (<operand1_value>) must be <comparison_op> '<operand_2>' (<operand2_value)
observable_timestamp_compare	cyber observable timestamp properties with a comparison are valid	In object '<identifier>', '<operand_1>' (<operand1_value>) must be <comparison_op> '<operand_2>' (<operand2_value>)
object_marking_circular_refs	that marking definitions do not contain circular references (i.e., they do not reference themselves in the 'object_marking_refs' property	'object_marking_refs' cannot contain any references to this marking definition object (no circular references)
granular_markings_circular_refs	that marking definitions do not contain circular refs (i.e., they do not reference themselves in the 'granular_markings' property	'granular markings' cannot contain any references to this marking definition object (no circular references)
marking_selectors	selectors in granular markings refer to items which are actually present in the object	'<selector>' is not a valid selector because '<index>' is not a valid index '<selector>' is not a valid selector because '<selector_segment>' is not a list. '<selector>' is not a valid selector because '<selector_segment>' is not a property.
observable_object_references	certain observable object properties reference the correct type of object	'<property>' in observable object '<identifier>' can't resolve '<embedded-property>' reference '<identifier>' '<property>' in observable object '<identifier>' must refer to an object of type '<type(s)>'
artifact_mime_type	the 'mime_type' property of artifact objects comes from the Template column in the IANA media type registry	the 'mime_type' property of object '<identifier>' ('<mime_type>') must be an IANA registered MIME Type of the form 'type/subtype'.
character_set	certain properties of cyber observable objects come from the IANA Character Set list.	The 'path_enc' property of object '<identifier>' ('<path_enc>') must be an IANA registered character set. The 'name_enc' property of object '<identifier>' ('<name_enc>') must be IANA registered character set.
language	the 'lang' property of SDOs is a valid RFC 5646 language code	'<lang>' is not a valid RFC 5646 language code.
software_language	the 'language' property of software objects is a valid ISO 639-2 language code	The 'languages' property of object '<identifier>' contains an invalid code ('<lang>').
patterns	that the syntax of the pattern of an indicator is valid, and that objects and properties referenced by the pattern are valid. This runs the cti-pattern-validator (https://github.com/oasis-open/cti-pattern-validator) to check the syntax of the pattern.	'<object>' is not a valid observable type name Custom Observable Object type '<object>' should start with 'x-' followed by a source unique identifier (like a domain name with dots replaced by hyphens), a hyphen and then the name Custom Observable Object type '<object>' should start with 'x-' '<property>' is not a valid observable property name Cyber Observable Object custom property '<property>' should start with 'x_' followed by a source unique identifier (like a domain name with dots replaced by underscores), an underscore and then the name Cyber Observable Object custom property '<property>' should start with

4.2 Optional Checks - STIX 2.1

Code	Name	Ensures...	Errors/Warnings
1	format-checks	all 1xx checks are run. Specifically:	

Continued on next page

Table 1 – continued from previous page

101	custom-prefix	names of custom object types, properties, observable objects, observable object properties, and observable object extensions follow the correct format	Note: This checks functionality that has been deprecated and replaced by extensions. Thus, this check only runs if extensions-use (401) is disabled. custom object type '<object>' should start with 'x-' followed by a source unique identifier (like a domain name with dots replaced by hyphens), a hyphen and then the name. custom property '<property>' should have a type that starts with 'x_' followed by a source unique identifier (like a domain name with dots replaced by a hyphen), a hyphen and then the name. Custom Observable Object type '<observable_object>' should start with 'x-' followed by a source unique identifier (like a domain name with dots replaced by hyphens), a hyphen and then the name. Custom Cyber Observable Object extension type '<observable-object-extension>' should start with 'x-' followed by a source unique identifier (like a domain with dots replaced by hyphens), a hyphen and then the name. Cyber Observable Object custom property '<observable_object_property>' should start with 'x_' followed by a source unique identifier (like a domain name with dots replaced by hyphens), a hyphen and then the name. Cyber Observable Object custom property '<property>' in the <extension> extension should start with 'x_' followed by a source unique (like a domain name with dots

Table 1 – continued from previous page

102	custom-prefix-lax	same as 101 but more lenient; no source identifier needed in prefix	Note: This checks functionality that has been deprecated and replaced by extensions. Thus, this check only runs if extensions-use (401) is disabled. custom object type '<object>' should start with 'x-' in order to be compatible with future versions of the STIX 2 specification. custom property '<property>' should have a type that starts with 'x_' in order to be compatible with future versions of the STIX 2 specification. Custom Observable Object type '<observable_object>' should start with 'x-'. Custom Observable Object extension type '<observable-object_extension>' should start with 'x-'. Cyber Observable Object custom property '<property>' should start with 'x_'. Cyber Observable Object custom property '<embedded_property>' in the <property> of the <object> object should start with 'x_'. Cyber Observable Object custom property '<property>' in the <extension> extension should start with 'x_'. Cyber Observable Object custom property '<property>' in the <extension_property> property of the <extension> extension should start with 'x_'.

Continued on next page

Table 1 – continued from previous page

103	uuid-check	objects use the recommended versions of UUID (v5 for SCOs, v4 for the rest)	Cyber Observable ID value <identifier> is not a valid UUIDv5 ID. Given ID value <identifier> is not a valid UUIDv4 ID.
111	open-vocab-format	values of open vocabularies follow the correct format	Open vocabulary value '<value>' should be all lowercase and use hyphens instead of spaces or underscores as word separators.
121	kill-chain-names	kill-chain-phase name and phase follow the correct format	kill_chain_name '<chain_name>' should be all lowercase and use hyphens instead of spaces or underscores as word separators. phase_name '<phase_name>' should be all lowercase and use hyphens instead of spaces or underscores as word separators
141	observable-object-keys	observable object keys follow the correct format	'<key_value>' is not a good key value. Observable Objects should use non- negative integers for their keys.
142	observable-dictionary-keys	dictionaries in cyber observable objects follow the correct format	As a dictionary key, '<key_value>' should be lowercase.
143	malware-analysis-product	malware analysis product names follow the correct format	The 'product' property of object '<identifier>' should be all lowercase with words separated by dash.
149	windows-process-priority-format	windows-process-ext's 'priority' follows the correct format	The 'priority' property of object '<identifier>' should end in '_CLASS'.

Continued on next page

Table 1 – continued from previous page

150	hash-length	keys in 'hashes'-type properties are not too long	Object '<identifier>' has a 'hashes' dictionary with a hash of type '<hash_type>', which is longer than 30 characters. Object '<identifier>' has an NTFS extension with an alternate data stream that has a 'hashes' dictionary with a hash of type '<hash_type>', which is longer than 30 characters. Object '<identifier>' has a Windows PE Binary File extension with a file header hash of '<hash>', which is longer than 30 characters. Object '<identifier>' has a Windows PE Binary File extension with an optional header that has a hash of '<hash>', which is longer than 30 characters. Object '<identifier>' has a Windows PE Binary File extension with a section that has a hash of '<hash>', which is longer than 30 characters. Object '<identifier>' hash a 'hashes' dictionary with a hash of type '<hash_type>', which is longer than 30 characters.
2	approved-values	all 2xx checks are run. Specifically:	
201	marking-definition-type	marking definitions use a valid definition_type	Marking definition 'definition_type' should be one of: <marking-definition-type>.

Continued on next page

Table 1 – continued from previous page

202	relationship-types	relationships are among those defined in the specification	'<object>' is not a suggested relationship source object for the '<relationship>' relationship. '<relationship>' is not a suggested relationship type for '<object>' objects. '<object>' is not a suggested relationship target object for '<object>' objects with the '<relationship>' relationship.
203	duplicate-ids	objects in a bundle with duplicate IDs have different *modified* timestamps	Duplicate ID '<identifier>' has identical 'modified' timestamp. If they are different versions of the same object, they should have different 'modified' properties,
210	all-vocabs	all of the following open vocabulary checks are run	**'<property>' contains a value not in** the <vocab_name>-ov vocabulary.
211	attack-motivation	certain property values are from the attack-motivation vocabulary	'<property>' contains a value not in the attack-motivation-ov vocabulary
212	attack-resource-level	certain property values are from the attack-resource-level vocabulary	'<property>' contains a value not in the attack-resource-level-ov vocabulary
213	identity-class	certain property values are from the identity-class vocabulary	'<property>' contains a value not in the identity-class-ov vocabulary
214	indicator-types	certain property values are from the indicator-types vocabulary	'<property>' contains a value not in the indicator-types-ov vocabulary
215	industry-sector	certain property values are from the industry-sector vocabulary	'<property>' contains a value not in the industry-sector-ov vocabulary
216	malware-types	certain property values are from the malware-types vocabulary	'<property>' contains a value not in the malware-types-ov vocabulary
218	report-types	certain property values are from the report-types vocabulary	'<property>' contains a value not in the report-types-ov vocabulary
219	threat-actor-types	certain property values are from the threat-actor-types vocabulary	'<property>' contains a value not in the threat-actor-types-ov vocabulary

Continued on next page

Table 1 – continued from previous page

220	threat-actor-role	certain property values are from the threat_actor_role vocabulary	'<property>' contains a value not in the threat-actor-role-ov vocabulary
221	threat-actor-sophistication	certain property values are from the threat_actor_sophistication vocabulary	'<property>' contains a value not in the threat-actor-sophistication-ov vocabulary
222	tool-types	certain property values are from the tool_types vocabulary	'<property>' contains a value not in the tool-types-ov vocabulary
223	region	certain property values are from the region vocabulary	'<property>' contains a value not in the region-ov vocabulary
225	grouping-context	certain property values are from the grouping-context vocabulary	'<property>' contains a value not in the grouping-context-ov vocabulary
226	implementation-languages	certain property values are from the implementation-languages vocabulary	'<property>' contains a value not in the implementation-languages-ov vocabulary
227	infrastructure-types	certain property values are from the infrastructure-types vocabulary	'<property>' contains a value not in the infrastructure-types-ov vocabulary
228	malware-capabilities	certain property values are from the malware-capabilities vocabulary	'<property>' contains a value not in the malware-capabilities-ov vocabulary
230	processor-architecture	certain property values are from the processor-architecture vocabulary	'<property>' contains a value not in the processor-architecture-ov vocabulary
231	malware-result	certain property values are from the malware-result vocabulary	'<property>' contains a value not in the malware-result-ov vocabulary

Continued on next page

Table 1 – continued from previous page

241	hash-algo	certain property values are from the hash-algo vocabulary	Object '<identifier>' has a 'hashes' dictionary with a hash of type '<hash_type>', which is not a value in the hash-algorithm-ov vocabulary nor a custom value prepended with 'x_'. Object '<identifier>' has an NTFS extension with an alternate data stream that has a 'hashes' dictionary with a hash of type '<hash_type>', which is not a value in the hash- algorithm-ov vocabulary nor a custom value prepended with 'x_'. Object '<identifier>' has a Windows PE Binary File extension with a file header hash of '<hash_type>', which is not a value in the hash-algorithm- vocabulary nor a custom value prepended with 'x_'. Object '<identifier>' has a Windows PE Binary File extension with an optional header that has a hash of '<hash_type>', which is not a value in the hash-algorithm-ov vocabulary nor a custom value prepended with 'x_'. Object '<identifier>' has a Windows PE Binary File extension with a section that has a hash of '<hash_type>', which is not a value in the hash-algorithm-ov vocabulary nor a custom value prepended with 'x_'.

Continued on next page

Table 1 – continued from previous page

243	windows-pebinary-type	certain property values are from the windows-pebinary-type vocabulary	Object '<identifier>' has a Windows PE Binary File extension with a 'pe_type' of '<pe_type>', which is not a value in the windows-pebinary-type-ov vocabulary.
244	account-type	certain property values are from the account-type vocabulary	Object '<identifier>'is a User Account Object with an 'account_type' of '<account_type>', which is not a value in the account-type-ov vocabulary.
245	indicator-pattern-types	certain property values are from the pattern-type vocabulary	'<property>' contains a value not in the pattern-type-ov vocabulary
270	all-external-sources	all of the following external source checks are run	
271	mime-type	file.mime_type is a valid IANA MIME type	The 'mime_type' property of object '<identifier>' ('<mime_type>') should be an IANA registered MIME Type of the form 'type/subtype'.
272	protocols	certain property values are valid IANA Service and Protocol names	The 'protocols' property of object '<identifier>' contains a value ('<protocol>') not in IANA Service Name and Transport Protocol Port Number Registry.
273	ipfix	certain property values are valid IANA IP Flow Information Export (IPFIX) Entities	The 'ipfix' property of object '<identifier>' contains a key ('<ipfix>') not in IANA IP Flow Information Export (IPFIX) Entities Registry.
274	http-request-headers	certain property values are valid HTTP request header names	The 'request_header' property of object '<identifier>' contains an invalid HTTP header ('<http_request_header>').
275	socket-options	certain property values are valid socket options	The 'options' property of object '<identifier>' contains a key ('<option>') that is not a valid socket option (SO|ICMP|ICMP6|IP|IPV6|MCAST|TCP|IRLMP)_*.

Continued on next page

Table 1 – continued from previous page

276	pdf-doc-info	certain property values are valid PDF Document Information Dictionary keys	The 'document_info_dict' property of object '<identifier>' contains a key ('<key>') that is not a valid PDF Document Information Dictionary key.
277	countries	certain property values are valid ISO 3166-1 ALPHA-2 codes	Location 'country' should be a valid ISO 3166-1 ALPHA-2 Code.
301	network-traffic-ports	network-traffic objects contain both src_port and dst_port	The Network Traffic object '<identifier>' should contain both the 'src_port' and 'dst_port' properties.
302	extref-hashes	external references SHOULD have hashes if they have a url	External reference '<src>' has a URL but no hash.
303	indicator-properties	Indicator objects have both name and description properties	Both the name and description properties SHOULD be present.
304	deprecated-properties	certain properties which have been deprecated are not being used	Included property '<property>' is deprecated within the indicated spec version.
305	extension-description	Extension Definitions have a description property	The 'description' property SHOULD be populated.
306	extension-properties	Ensure toplevel-property-extensions include the extension_properties property	For extensions of the 'toplevel- property-extension' type, the 'extension_properties' property SHOULD include one or more property names.
401	extensions-use	custom objects, properties, and observable extensions have been implemented with Extension Definitions	Custom object type '<object>' should be implemented using an extension with an 'extension_type' of 'new-sdo'. Custom property '<property>' should be 'implemented using an extension with an 'extension_type' of 'property- extension' or 'toplevel-property- extension'. Custom Cyber Observable Object extension type '<extension>' should be implemented using an 'extension_type' of 'property-extension'.

CHAPTER 5

Contributing

We're thrilled that you're interested in contributing to the stix2-validator! Here are some things you should know:

- contribution-guide.org has great ideas for contributing to any open-source project (not just this one).
- All contributors must sign a Contributor License Agreement. See CONTRIBUTING.md in the project repository for specifics.
- If you are planning to implement a major feature (vs. fixing a bug), please discuss with a project maintainer first to ensure you aren't duplicating the work of someone else, and that the feature is likely to be accepted.

Now, let's get started!

5.1 Setting up a development environment

We recommend using a virtualenv.

1. Clone the repository. If you're planning to make pull request, you should fork the repository on GitHub and clone your fork instead of the main repo:

```
$ git clone https://github.com/yourusername/cti-stix-validator.git
```

2. Install develoment-related dependencies and set up git submodules:

```
$ cd cti-stix-validator
$ pip install -r requirements.txt
$ git submodule update --init --recursive
```

3. Install pre-commit git hooks:

```
$ pre-commit install
```

At this point you should be able to make changes to the code.

5.2 Code style

All code should follow PEP 8. We allow for line lengths up to 160 characters, but any lines over 80 characters should be the exception rather than the rule. PEP 8 conformance will be tested automatically by Tox and Travis-CI (see below).

5.3 Testing

Note: All of the tools mentioned in this section are installed when you run `pip install -r requirements.txt`.

This project uses pytest for testing. We encourage the use of test-driven development (TDD), where you write (failing) tests that demonstrate a bug or proposed new feature before writing code that fixes the bug or implements the features. Any code contributions should come with new or updated tests.

To run the tests in your current Python environment, use the `pytest` command from the root project directory:

```
$ pytest
```

This should show all of the tests that ran, along with their status.

You can run a specific test file by passing it on the command line:

```
$ pytest stix2validator/test/test_<xxx>.py
```

To ensure that the test you wrote is running, you can deliberately add an `assert False` statement at the beginning of the test. This is another benefit of TDD, since you should be able to see the test failing (and ensure it's being run) before making it pass.

tox allows you to test a package across multiple versions of Python. Setting up multiple Python environments is beyond the scope of this guide, but feel free to ask for help setting them up. Tox should be run from the root directory of the project:

```
$ tox
```

We aim for high test coverage, using the coverage.py library. Though it's not an absolute requirement to maintain 100% coverage, all code contributions must be accompanied by tests. To run coverage and look for untested lines of code, run:

```
$ pytest --cov=stix2validator
$ coverage html
```

then look at the resulting report in `htmlcov/index.html`.

All commits pushed to the `master` branch or submitted as a pull request are tested with Travis-CI automatically.

5.4 Adding a dependency

One of the pre-commit hooks we use in our develoment environment enforces a consistent ordering to imports. If you need to add a new library as a dependency please add it to the *known_third_party* section of *.isort.cfg* to make sure the import is sorted correctly.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search