

---

# **zenoh-python**

***Release 0.6.0-beta.1***

**Sep 30, 2022**



---

## Contents

---

<b>1</b>	<b>module zenoh</b>	<b>1</b>
1.1	Config . . . . .	1
1.2	CongestionControl . . . . .	2
1.3	Encoding . . . . .	2
1.4	Hello . . . . .	3
1.5	KeyExpr . . . . .	3
1.6	ZenohId . . . . .	3
1.7	Priority . . . . .	4
1.8	Query . . . . .	4
1.9	Queryable . . . . .	4
1.10	QueryConsolidation . . . . .	5
1.11	QueryTarget . . . . .	5
1.12	Reliability . . . . .	5
1.13	Reply . . . . .	5
1.14	Sample . . . . .	5
1.15	SampleKind . . . . .	6
1.16	Selector . . . . .	6
1.17	Session . . . . .	7
1.18	Subscriber . . . . .	9
1.19	PullSubscriber . . . . .	9
1.20	Publisher . . . . .	9
1.21	Timestamp . . . . .	10
1.22	Value . . . . .	10
1.23	Info . . . . .	10
	<b>Python Module Index</b>	<b>11</b>
	<b>Index</b>	<b>13</b>



`zenoh.init_logger()`

Initialize the logger used by the Rust implementation of this API.

Once initialized, you can configure the logs displayed by the API using the `RUST_LOG` environment variable. For instance, start python with the *debug* logs available:

```
$ RUST_LOG=debug python
```

More details on the `RUST_LOG` configuration on [https://docs.rs/env\\_logger/latest/env\\_logger](https://docs.rs/env_logger/latest/env_logger)

## 1.1 Config

**class** `zenoh.Config`

**static** `from_file(filename: str)`

Reads the configuration from a file. The file's extension must be json, json5 or yaml.

**static** `from_json5(json: str)`

Reads the configuration from a JSON5 string.

JSON5 is a superset of JSON, so any JSON string is a valid input for this function.

**static** `from_obj(obj)`

Reads the configuration from *obj* as if it was a JSON file.

**get\_json** (*path: str*) → str

Returns the part of the configuration at *path*, in a JSON-serialized form.

**insert\_json5** (*path: str, value: str*) → str

Inserts the provided value (read from JSON) at the given path in the configuration.

## 1.2 CongestionControl

**class** zenoh.CongestionControl

Defines the network's behaviour regarding a message when heavily congested.

**static** BLOCK() → zenoh.enums.CongestionControl

Prevents the message from being dropped at all cost. In the face of heavy congestion on a part of the network, this could result in your publisher node blocking.

**static** DROP() → zenoh.enums.CongestionControl

Allows the message to be dropped if all buffers are full.

## 1.3 Encoding

**class** zenoh.Encoding

**static** APP\_CUSTOM() → zenoh.enums.Encoding

**static** APP\_FLOAT() → zenoh.enums.Encoding

**static** APP\_INTEGER() → zenoh.enums.Encoding

**static** APP\_JSON() → zenoh.enums.Encoding

**static** APP\_OCTET\_STREAM() → zenoh.enums.Encoding

**static** APP\_PROPERTIES() → zenoh.enums.Encoding

**static** APP\_SQL() → zenoh.enums.Encoding

**static** APP\_XHTML\_XML() → zenoh.enums.Encoding

**static** APP\_XML() → zenoh.enums.Encoding

**static** APP\_X\_WWW\_FORM\_URLENCODED() → zenoh.enums.Encoding

**static** EMPTY() → zenoh.enums.Encoding

**static** IMAGE\_GIF() → zenoh.enums.Encoding

**static** IMAGE\_JPEG() → zenoh.enums.Encoding

**static** IMAGE\_PNG() → zenoh.enums.Encoding

**static** TEXT\_CSS() → zenoh.enums.Encoding

**static** TEXT\_CSV() → zenoh.enums.Encoding

**static** TEXT\_HTML() → zenoh.enums.Encoding

**static** TEXT\_JAVASCRIPT() → zenoh.enums.Encoding

**static** TEXT\_JSON() → zenoh.enums.Encoding

**static** TEXT\_PLAIN() → zenoh.enums.Encoding

**static** TEXT\_XML() → zenoh.enums.Encoding

**append**(s: str)

**static** from\_str(s: str) → zenoh.enums.Encoding

## 1.4 Hello

**class** `zenoh.Hello`

Represents a single Zenoh node discovered through scouting.

**locators**

The locators through which this node may be addressed.

**whatami**

The node's type, returning either `None`, `'peer'`, `'router'`, or `'client'`.

**zid**

The node's Zenoh UUID.

## 1.5 KeyExpr

**class** `zenoh.KeyExpr`

Zenoh's address space is designed around keys which serve as the names of resources.

Keys are slash-separated lists of non-empty UTF8 strings. They may not contain the following characters: `$*#?`.

Zenoh's operations are executed on key expressions, a small language that allows the definition of sets of keys via the use of wildcards: - `*` is the single-chunk wildcard, and will match any chunk: `"a/c"` will match `"a/b/c"`, `"a/hello/c"`, etc... - `***` is the 0 or more chunks wildcard: `"a/*/c"` matches `"a/c"`, `"a/b/c"`, `"a/b/hello/c"`... - `$*` is the subchunk wildcard, it will match any amount of non-`/` characters: `"a/b$*"` matches `"a/b"`, `"a/because"`, `"a/blue"`... but not `"a/c"` nor `"a/blue/c"`

To allow for better performance and gain the property that two key expressions define the same set if and only if they are the same string, the rules of canon form are mandatory for a key expression to be propagated by a Zenoh network: - `**/*` may not exist, as it could always be replaced by the shorter `**`, - `**/*` may not exist, and must be written as its equivalent `*/**` instead, - `$*` may not exist alone in a chunk, as it must be written `*` instead.

The `KeyExpr.autocanonize` constructor exists to correct eventual infringements of the canonization rules.

A `KeyExpr` is a string that has been validated to be a valid Key Expression.

**static** `autocanonize(expr: str) → zenoh.keyexpr.KeyExpr`

This alternative constructor for key expressions will attempt to canonize the passed expression before checking if it is valid.

Raises a `zenoh.ZError` exception if `expr` is not a valid key expression.

**includes** (`other: zenoh.keyexpr.KeyExpr`) → bool

This method returns `True` if all of the keys defined by `other` also belong to the set defined by `self`.

**intersects** (`other: zenoh.keyexpr.KeyExpr`) → bool

This method returns `True` if there exists at least one key that belongs to both sets defined by `self` and `other`.

**undeclare** (`session: Session`)

Undeclares a key expression previously declared on the session.

## 1.6 ZenohId

**class** `zenoh.ZenohId`

A Zenoh UUID

## 1.7 Priority

**class** zenoh.**Priority**

The priority of a sending operation.

They are ordered à la Linux priority: *Priority.REAL\_TIME()* < *Priority.INTERACTIVE\_HIGH()* < *Priority.INTERACTIVE\_LOW()* < *Priority.DATA()* < *Priority.BACKGROUND()*

**static** **BACKGROUND** () → zenoh.enums.Priority

**static** **DATA** () → zenoh.enums.Priority

**static** **DATA\_HIGH** () → zenoh.enums.Priority

**static** **DATA\_LOW** () → zenoh.enums.Priority

**static** **INTERACTIVE\_HIGH** () → zenoh.enums.Priority

**static** **INTERACTIVE\_LOW** () → zenoh.enums.Priority

**static** **REAL\_TIME** () → zenoh.enums.Priority

## 1.8 Query

**class** zenoh.**Query**

**decode\_parameters** () → Dict[str, str]

Decodes the value selector into a dictionary.

Raises a ZError if duplicate keys are found, as they might otherwise be used for HTTP Parameter Pollution like attacks.

**key\_expr**

The query's targeted key expression

**parameters**

The query's value selector. If you'd rather not bother with parsing it yourself, use *self.decode\_parameters()* instead.

**reply** (*sample: zenoh.value.Sample*)

Allows you to reply to a query. You may send any amount of replies to a single query, including 0.

**selector**

The query's selector as a whole.

## 1.9 Queryable

**class** zenoh.**Queryable** (*inner: \_Queryable, receiver*)

A handle to a queryable.

Its main purpose is to keep the queryable active as long as it exists.

When constructed through *Session.declare\_queryable(session, keyexpr, handler)*, it exposes *handler*'s receiver through *self.receiver*.

**undeclare** ()

Stops the queryable.



## 1.10 QueryConsolidation

```
class zenoh.QueryConsolidation
```

## 1.11 QueryTarget

```
class zenoh.QueryTarget
```

## 1.12 Reliability

```
class zenoh.Reliability
```

Used by subscribers to inform the network of the reliability it wishes to obtain.

**static** **BEST\_EFFORT** () → zenoh.enums.CongestionControl  
 Informs the network that dropping some messages is acceptable

**static** **RELIABLE** () → zenoh.enums.CongestionControl  
 Informs the network that this subscriber wishes for all publications to reliably reach it.

Note that if a publisher puts a sample with the *CongestionControl.DROP()* option, this reliability requirement may be infringed to prevent slow readers from blocking the network.

## 1.13 Reply

```
class zenoh.Reply
```

**err**  
 The reply's error value.  
 Raises a ZError if the *self* is actually an *ok* reply.

**ok**  
 The reply's inner data sample.  
 Raises a ZError if the *self* is actually an *err* reply.

**replier\_id**  
 The reply's sender's id.

## 1.14 Sample

```
class zenoh.Sample
```

A KeyExpr-Value pair, annotated with the kind (PUT or DELETE) of publication used to emit it and a timestamp.

**encoding**  
 A shortcut to *self.value.encoding*

**key\_expr**  
 The sample's key expression

**kind**  
The sample's kind

**payload**  
A shortcut to *self.value.payload*

**timestamp**  
The sample's timestamp. May be None.

**value**  
The sample's value

## 1.15 SampleKind

**class** zenoh.**SampleKind**  
Similar to an HTTP METHOD: only PUT and DELETE are currently supported.

**static** **DELETE** () → zenoh.enums.SampleKind

**static** **PUT** () → zenoh.enums.SampleKind

## 1.16 Selector

**class** zenoh.**Selector**  
A selector is the combination of a [Key Expression](crate::prelude::KeyExpr), which defines the set of keys that are relevant to an operation, and a *parameters*, a set of key-value pairs with a few uses: \* specifying arguments to a queryable, allowing the passing of Remote Procedure Call parameters \* filtering by value, \* filtering by metadata, such as the timestamp of a value,

When in string form, selectors look a lot like a URI, with similar semantics: \* the *key\_expr* before the first ? must be a valid key expression. \* the *parameters* after the first ? should be encoded like the query section of a URL:

- key-value pairs are separated by &,
- the key and value are separated by the first =,
- in the absence of =, the value is considered to be the empty string,
- both key and value should use percent-encoding to escape characters,
- defining a value for the same key twice is considered undefined behavior.

Zenoh intends to standardize the usage of a set of keys. To avoid conflicting with RPC parameters, the Zenoh team has settled on reserving the set of keys that start with non-alphanumeric characters.

This document will summarize the standardized keys for which Zenoh provides helpers to facilitate coherent behavior for some operations.

Queryable implementers are encouraged to prefer these standardized keys when implementing their associated features, and to prefix their own keys to avoid having conflicting keys with other queryables.

Here are the currently standardized keys for Zenoh: \* *\_time*: used to express interest in only values dated within a certain time range, values for

    this key must be readable by the [Zenoh Time DSL](zenoh\_util::time\_range::TimeRange) for the value to be considered valid.

- *\_filter*: TBD Zenoh intends to provide helper tools to allow the value associated with this key to be treated as a predicate that the value should fulfill before being returned. A DSL will be designed by the Zenoh team to express these predicates.

**decode\_parameters** () → Dict[str, str]

Decodes the value selector part of the selector.

Raises a ZError if some keys were duplicated: duplicated keys are considered undefined behaviour, but we encourage you to refuse to process incoming messages with duplicated keys, as they might be attempting to use HTTP Parameter Pollution like exploits.

**key\_expr**

The key expression part of the selector.

**parameters**

The value selector part of the selector.

**set\_parameters**

The value selector part of the selector.

## 1.17 Session

**class zenoh.Session**

A Zenoh Session, the core interaction point with a Zenoh network.

**close** ()

Closes the Session

**declare\_keyexpr** (keyexpr: Union[KeyExpr, \_KeyExpr, str]) → zenoh.keyexpr.KeyExpr

Informs Zenoh that you intend to use the provided Key Expression repeatedly.

This function returns an optimized representation of the passed *keyexpr*.

**declare\_publisher** (keyexpr: Union[KeyExpr, \_KeyExpr, str], priority: zenoh.enums.Priority = None, congestion\_control: zenoh.enums.CongestionControl = None)

Declares a publisher, which you may use to send values repeatedly onto a same key expression.

**declare\_pull\_subscriber** (keyexpr: Union[KeyExpr, \_KeyExpr, str], handler: Union[zenoh.closures.IHandler[zenoh.value.Sample, typing.Any, typing.Any][zenoh.value.Sample, Any, Any], zenoh.closures.IClosure[zenoh.value.Sample, typing.Any][zenoh.value.Sample, Any], Tuple[zenoh.closures.IClosure, Any], Tuple[Callable[[zenoh.value.Sample], Any], Callable[[None], Any], Tuple[Callable[[zenoh.value.Sample], Any], Callable[[None], None]], Callable[[zenoh.value.Sample], Any]], reliability: zenoh.enums.Reliability = None) → zenoh.session.PullSubscriber

Declares a pull-mode subscriber, which will receive a single published sample with a key expression intersecting *keyexpr* any time its *pull* method is called.

These samples are passed to the *handler*'s closure as instances of the *Sample* class.

The *handler*'s receiver is returned as the *receiver* field of the return value.

```
declare_queryable (keyexpr: Union[KeyExpr, _KeyExpr, str], handler:
                    Union[zenoh.closures.IHandler[zenoh.queryable.Query,
                    typing.Any, typing.Any][zenoh.queryable.Query, Any, Any],
                    zenoh.closures.IClosure[zenoh.queryable.Query,
                    typing.Any][zenoh.queryable.Query, Any], Tuple[zenoh.closures.IClosure,
                    Any], Tuple[Callable[[zenoh.queryable.Query], Any], Callable[[], None],
                    Any], Tuple[Callable[[zenoh.queryable.Query], Any], Callable[[], None]],
                    Callable[[zenoh.queryable.Query], Any]], complete: bool = None)
```

Declares a queryable, which will receive queries intersecting with *keyexpr*.

These queries are passed to the *handler* as instances of the ‘Query’ class, which lets you respond when applicable.

The *handler*’s receiver is returned as the *receiver* field of the return value.

IMPORTANT: due to how RAI and Python work, you MUST bind this function’s return value to a variable in order for it to function as expected. This is because as soon as a value is no longer referenced in Python, that value’s destructor will run, which will undeclare your queryable, stopping it immediately.

```
declare_subscriber (keyexpr: Union[KeyExpr, _KeyExpr, str], han-
                    dler: Union[zenoh.closures.IHandler[zenoh.value.Sample,
                    typing.Any, typing.Any][zenoh.value.Sample, Any,
                    Any], zenoh.closures.IClosure[zenoh.value.Sample,
                    typing.Any][zenoh.value.Sample, Any], Tuple[zenoh.closures.IClosure,
                    Any], Tuple[Callable[[zenoh.value.Sample], Any], Callable[[], None],
                    Any], Tuple[Callable[[zenoh.value.Sample], Any], Callable[[], None]],
                    Callable[[zenoh.value.Sample], Any]], reliability: zenoh.enums.Reliability =
                    None) → zenoh.session.Subscriber
```

Declares a subscriber, which will receive any published sample with a key expression intersecting *keyexpr*.

These samples are passed to the *handler*’s closure as instances of the *Sample* class.

The *handler*’s receiver is returned as the *receiver* field of the return value.

IMPORTANT: due to how RAI and Python work, you MUST bind this function’s return value to a variable in order for it to function as expected. This is because as soon as a value is no longer referenced in Python, that value’s destructor will run, which will undeclare your subscriber, deactivating the subscription immediately.

```
delete (keyexpr: Union[KeyExpr, _KeyExpr, str], priority: zenoh.enums.Priority = None, conges-
        tion_control: zenoh.enums.CongestionControl = None)
        Deletes a value.
```

```
get (selector: Union[Selector, _Selector, KeyExpr, _KeyExpr, str], handler:
     Union[zenoh.closures.IHandler[zenoh.value.Reply, typing.Any, ~Receiver][zenoh.value.Reply,
     Any, Receiver], zenoh.closures.IClosure[zenoh.value.Reply, typing.Any][zenoh.value.Reply,
     Any], Tuple[zenoh.closures.IClosure, Receiver], Tuple[Callable[[zenoh.value.Reply], Any],
     Callable[[], None], Receiver], Tuple[Callable[[zenoh.value.Reply], Any], Callable[[], None]],
     Callable[[zenoh.value.Reply], Any]], consolidation: zenoh.enums.QueryConsolidation = None,
     target: zenoh.enums.QueryTarget = None) → Receiver
     Emits a query.
```

```
info ()
```

Returns an accessor for informations about this Session

```
put (keyexpr: Union[KeyExpr, _KeyExpr, str], value: Union[zenoh.value.IValue, bytes, str, int,
float, object], encoding=None, priority: zenoh.enums.Priority = None, congestion_control:
zenoh.enums.CongestionControl = None, sample_kind: zenoh.enums.SampleKind = None)
        Sends a value over Zenoh.
```

## 1.18 Subscriber

**class** zenoh.**Subscriber** (*s: \_Subscriber, receiver=None*)

A handle to a subscription.

Its main purpose is to keep the subscription active as long as it exists.

When constructed through *Session.declare\_subscriber(session, keyexpr, handler)*, it exposes *handler*'s receiver through *self.receiver*.

**undecclare** ()

Undeclares the subscription

## 1.19 PullSubscriber

**class** zenoh.**PullSubscriber** (*s: \_PullSubscriber, receiver=None*)

A handle to a pull subscription.

Its main purpose is to keep the subscription active as long as it exists.

When constructed through *Session.declare\_pull\_subscriber(session, keyexpr, handler)*, it exposes *handler*'s receiver through *self.receiver*.

Calling *self.pull()* will prompt the Zenoh network to send a new sample when available.

**pull** ()

Prompts the Zenoh network to send a new sample if available. Note that this sample will not be returned by this function, but provided to the handler's callback.

**undecclare** ()

Undeclares the subscription

## 1.20 Publisher

**class** zenoh.**Publisher** (*p: \_Publisher*)

Use *Publisher*'s (constructed with *'Session.declare\_publisher*) when you want to send values often for the same key expression, as declaring them informs Zenoh that this is your intent, and optimizations will be set up to do so.

**delete** ()

An optimised version of `'session.delete(self.key_expr)`

**key\_expr**

This *Publisher*'s key expression

**put** (*value: Union[zenoh.value.IValue, bytes, str, int, float, object], encoding: zenoh.enums.Encoding = None*)

An optimised version of `'session.put(self.key_expr, value, encoding=encoding)`

**undecclare** ()

Stops the publisher.

## 1.21 Timestamp

**class** zenoh.**Timestamp**

A timestamp taken from the Zenoh HLC (Hybrid Logical Clock).

These timestamps are guaranteed to be unique, as each machine annotates its perceived time with a UUID, which is used as the least significant part of the comparison operation.

**seconds\_since\_unix\_epoch**

Returns the number of seconds since the Unix Epoch.

You shouldn't use this for comparison though, and rely on comparison operators between members of this class.

## 1.22 Value

**class** zenoh.**Value**

A Value is a pair of a binary payload, and a mime-type-like encoding string.

When constructed with *encoding=None*, the encoding will be selected depending on the payload's type.

## 1.23 Info

**class** zenoh.**Info** (*session: \_Session*)

**peers\_zid**() → List[zenoh.value.ZenohId]

Returns the neighboring peers' identifiers

**routers\_zid**() → List[zenoh.value.ZenohId]

Returns the neighboring routers' identifiers

**zid**() → zenoh.value.ZenohId

Returns this Zenoh Session's identifier

**Z**

zenoh, [1](#)





## A

APP\_CUSTOM() (*zenoh.Encoding static method*), 2  
 APP\_FLOAT() (*zenoh.Encoding static method*), 2  
 APP\_INTEGER() (*zenoh.Encoding static method*), 2  
 APP\_JSON() (*zenoh.Encoding static method*), 2  
 APP\_OCTET\_STREAM() (*zenoh.Encoding static method*), 2  
 APP\_PROPERTIES() (*zenoh.Encoding static method*), 2  
 APP\_SQL() (*zenoh.Encoding static method*), 2  
 APP\_X\_WWW\_FORM\_URL\_ENCODED() (*zenoh.Encoding static method*), 2  
 APP\_XHTML\_XML() (*zenoh.Encoding static method*), 2  
 APP\_XML() (*zenoh.Encoding static method*), 2  
 append() (*zenoh.Encoding method*), 2  
 autocanonicalize() (*zenoh.KeyExpr static method*), 3

## B

BACKGROUND() (*zenoh.Priority static method*), 4  
 BEST\_EFFORT() (*zenoh.Reliability static method*), 5  
 BLOCK() (*zenoh.CongestionControl static method*), 2

## C

close() (*zenoh.Session method*), 7  
 Config (*class in zenoh*), 1  
 CongestionControl (*class in zenoh*), 2

## D

DATA() (*zenoh.Priority static method*), 4  
 DATA\_HIGH() (*zenoh.Priority static method*), 4  
 DATA\_LOW() (*zenoh.Priority static method*), 4  
 declare\_keyexpr() (*zenoh.Session method*), 7  
 declare\_publisher() (*zenoh.Session method*), 7  
 declare\_pull\_subscriber() (*zenoh.Session method*), 7  
 declare\_queryable() (*zenoh.Session method*), 7  
 declare\_subscriber() (*zenoh.Session method*), 8  
 decode\_parameters() (*zenoh.Query method*), 4  
 decode\_parameters() (*zenoh.Selector method*), 7

delete() (*zenoh.Publisher method*), 9  
 DELETE() (*zenoh.SampleKind static method*), 6  
 delete() (*zenoh.Session method*), 8  
 DROP() (*zenoh.CongestionControl static method*), 2

## E

EMPTY() (*zenoh.Encoding static method*), 2  
 Encoding (*class in zenoh*), 2  
 encoding (*zenoh.Sample attribute*), 5  
 err (*zenoh.Reply attribute*), 5

## F

from\_file() (*zenoh.Config static method*), 1  
 from\_json5() (*zenoh.Config static method*), 1  
 from\_obj() (*zenoh.Config static method*), 1  
 from\_str() (*zenoh.Encoding static method*), 2

## G

get() (*zenoh.Session method*), 8  
 get\_json() (*zenoh.Config method*), 1

## H

Hello (*class in zenoh*), 3

## I

IMAGE\_GIF() (*zenoh.Encoding static method*), 2  
 IMAGE\_JPEG() (*zenoh.Encoding static method*), 2  
 IMAGE\_PNG() (*zenoh.Encoding static method*), 2  
 includes() (*zenoh.KeyExpr method*), 3  
 Info (*class in zenoh*), 10  
 info() (*zenoh.Session method*), 8  
 init\_logger() (*in module zenoh*), 1  
 insert\_json5() (*zenoh.Config method*), 1  
 INTERACTIVE\_HIGH() (*zenoh.Priority static method*), 4  
 INTERACTIVE\_LOW() (*zenoh.Priority static method*), 4  
 intersects() (*zenoh.KeyExpr method*), 3

## K

`key_expr` (*zenoh.Publisher attribute*), 9  
`key_expr` (*zenoh.Query attribute*), 4  
`key_expr` (*zenoh.Sample attribute*), 5  
`key_expr` (*zenoh.Selector attribute*), 7  
`KeyExpr` (*class in zenoh*), 3  
`kind` (*zenoh.Sample attribute*), 5

## L

`locators` (*zenoh.Hello attribute*), 3

## O

`ok` (*zenoh.Reply attribute*), 5

## P

`parameters` (*zenoh.Query attribute*), 4  
`parameters` (*zenoh.Selector attribute*), 7  
`payload` (*zenoh.Sample attribute*), 6  
`peers_zid`() (*zenoh.Info method*), 10  
`Priority` (*class in zenoh*), 4  
`Publisher` (*class in zenoh*), 9  
`pull`() (*zenoh.PullSubscriber method*), 9  
`PullSubscriber` (*class in zenoh*), 9  
`put`() (*zenoh.Publisher method*), 9  
`PUT`() (*zenoh.SampleKind static method*), 6  
`put`() (*zenoh.Session method*), 8

## Q

`Query` (*class in zenoh*), 4  
`Queryable` (*class in zenoh*), 4  
`QueryConsolidation` (*class in zenoh*), 5  
`QueryTarget` (*class in zenoh*), 5

## R

`REAL_TIME`() (*zenoh.Priority static method*), 4  
`Reliability` (*class in zenoh*), 5  
`RELIABLE`() (*zenoh.Reliability static method*), 5  
`replier_id` (*zenoh.Reply attribute*), 5  
`Reply` (*class in zenoh*), 5  
`reply`() (*zenoh.Query method*), 4  
`routers_zid`() (*zenoh.Info method*), 10

## S

`Sample` (*class in zenoh*), 5  
`SampleKind` (*class in zenoh*), 6  
`seconds_since_unix_epoch` (*zenoh.Timestamp attribute*), 10  
`Selector` (*class in zenoh*), 6  
`selector` (*zenoh.Query attribute*), 4  
`Session` (*class in zenoh*), 7  
`set_parameters` (*zenoh.Selector attribute*), 7  
`Subscriber` (*class in zenoh*), 9

## T

`TEXT_CSS`() (*zenoh.Encoding static method*), 2  
`TEXT_CSV`() (*zenoh.Encoding static method*), 2  
`TEXT_HTML`() (*zenoh.Encoding static method*), 2  
`TEXT_JAVASCRIPT`() (*zenoh.Encoding static method*), 2  
`TEXT_JSON`() (*zenoh.Encoding static method*), 2  
`TEXT_PLAIN`() (*zenoh.Encoding static method*), 2  
`TEXT_XML`() (*zenoh.Encoding static method*), 2  
`Timestamp` (*class in zenoh*), 10  
`timestamp` (*zenoh.Sample attribute*), 6

## U

`undeclare`() (*zenoh.KeyExpr method*), 3  
`undeclare`() (*zenoh.Publisher method*), 9  
`undeclare`() (*zenoh.PullSubscriber method*), 9  
`undeclare`() (*zenoh.Queryable method*), 4  
`undeclare`() (*zenoh.Subscriber method*), 9

## V

`Value` (*class in zenoh*), 10  
`value` (*zenoh.Sample attribute*), 6

## W

`whatami` (*zenoh.Hello attribute*), 3

## Z

`zenoh` (*module*), 1  
`ZenohId` (*class in zenoh*), 3  
`zid` (*zenoh.Hello attribute*), 3  
`zid`() (*zenoh.Info method*), 10