featureprobe-server-sdk-python

FeatureProbe

Aug 30, 2022

CONTENTS:

Feature Probe is an open source feature management service. This SDK is used to control features in Python programs. This SDK is designed primarily for use in multi-user systems such as web servers and applications.

CHAPTER

ONE

TABLE OF CONTENTS

1.1 User Guide

This page has been moved to docs.featureprobe.io

1.2 API References

1.2.1 featureprobe package

featureprobe.client module

class featureprobe.client.**Client**(*server_sdk_key: str*, *config: ~featureprobe.internal.defaultable.defaultable.<locals>.check_or_create = <featureprobe.config.Config object>*)

Bases: `object`

A client for the FeatureProbe API. Client instances are thread-safe.

Applications should instantiate a single *Client* for the lifetime of their application.

close()

Safely shut down FeatureProbe client instance

flush()

Manually push events

value(*toggle_key: str*, *user:* User, *default*) → Any

Gets the evaluated value of a toggle.

Parameters

- **toggle_key** – The key of toggle in this environment.
- **user** – *User* to be evaluated.
- **default** – The default value to be returned.

Returns

Dependents on the toggle's type.

value_detail(*toggle_key: str*, *user:* User, *default*) → *Detail*

Gets the detailed evaluated results of a toggle.

Parameters

- **toggle_key** – The key of toggle in this environment.
- **user** – *User* to be evaluated.
- **default** – The default value to be returned.

Returns
Detail contains the *value*, *rule_index*, *version*, and *reason* of this evaluation.

featureprobe.config module

class featureprobe.config.**SyncMode**(*value*)

Bases: str, Enum

An enumeration.

FILE = 'file'

POOLING = 'pooling'

featureprobe.detail module

class featureprobe.detail.**Detail**(*value=None*, *rule_index: Optional[int] = None*, *version: Optional[int] = None*, *reason: Optional[str] = None*)

Bases: object

featureprobe.evaluation_result module

class featureprobe.evaluation_result.**EvaluationResult**(*value*, *rule_index: Optional[int]*, *variation_index: Optional[int]*, *version: int*, *reason: str*)

Bases: object

featureprobe.http_config module

featureprobe.user module

class featureprobe.user.**User**(*attrs: Optional[Dict[str, str]] = None*, *stable_rollout_key: Optional[str] = None*)

Bases: object

A collection of attributes that can affect toggle evaluation.

Usually corresponding to a user of your application.

property attrs: Dict[str, str]

Gets all attributes of a FeatureProbe User

has_attr(*attr: str*) → bool

Checks if an attribute exists.

Parameters
attr – Attribute name / key.

Returns
bool

property key: str

Gets FeatureProbe User unique identifier

stable_rollout(*key*)

to_dict() → dict

with_attr(*key: str*, *value: str*) → *User*

Adds an attribute to the user.

Parameters

- **key** – Attribute key / name.
- **value** – Attribute value.

Returns
User

Usage:

```
>>> import featureprobe as fp
>>> user = fp.User('unique id').with_attr('key1', 'value1').with_attr('key2',
↪'value2')
```

Module contents

class featureprobe.**AccessCounter**(*value: str*, *version: int*, *index: int*)

Bases: object

property count

increment()

property index

is_group(*event:* AccessEvent)

to_dict() → dict

property value

property version

class featureprobe.**AccessEvent**(*timestamp: int*, *user:* User, *key: str*, *value: str*, *version: int*, *index: int*)

Bases: Event

property index

property key

property value

property version

class featureprobe.**AccessRecorder**

Bases: object

add(*_event:* AccessEvent)

clear()

property counters

property end_time

snapshot()

property start_time

to_dict() → dict

class featureprobe.**Client**(*server_sdk_key: str*, *config: ~featureprobe.internal.defaultable.defaultable.<locals>.check_or_create = <featureprobe.config.Config object>*)

Bases: object

A client for the FeatureProbe API. Client instances are thread-safe.

Applications should instantiate a single *Client* for the lifetime of their application.

close()

Safely shut down FeatureProbe client instance

flush()

Manually push events

value(*toggle_key: str*, *user:* User, *default*) → Any

Gets the evaluated value of a toggle.

Parameters

- **toggle_key** – The key of toggle in this environment.
- **user** – *User* to be evaluated.
- **default** – The default value to be returned.

Returns

Dependents on the toggle's type.

value_detail(*toggle_key: str*, *user:* User, *default*) → *Detail*

Gets the detailed evaluated results of a toggle.

Parameters

- **toggle_key** – The key of toggle in this environment.
- **user** – *User* to be evaluated.
- **default** – The default value to be returned.

Returns

Detail contains the *value*, *rule_index*, *version*, and *reason* of this evaluation.

class featureprobe.**Condition**(*subject: str*, *type_: Union[*ConditionType*, str]*, *predicate: Union[Predicate, str]*, *objects: List[str]*)

Bases: object

classmethod from_json(*json_*)

match_objects(*user:* User, *segments: Optional[Dict[str,* Segment*]]*) → bool

property objects: List[str]

property predicate: Predicate

property subject: str

property type: *ConditionType*

class featureprobe.**ConditionType**(*value*)

Bases: str, Enum

An enumeration.

DATETIME = 'datetime'

NUMBER = 'number'

SEGMENT = 'segment'

SEMVER = 'semver'

STRING = 'string'

featureprobe.**Config**(**args*, ***kwargs*)

class featureprobe.**Context**(*sdk_key: str*, *config: Config*)

Bases: object

property event_url

property headers

property http_config

property location

property refresh_interval

property sdk_key

property synchronizer_url

class featureprobe.**DataRepository**

Bases: object

abstract close()

abstract classmethod from_context(*context:* Context) → *DataRepository*

abstract get_all_segment() → Dict[str, *Toggle*]

abstract get_all_toggle() → Dict[str, *Toggle*]

abstract get_segment(*key: str*) → *Toggle*

abstract get_toggle(*key: str*) → *Toggle*

abstract property initialized: bool

abstract refresh(*repo:* Repository)

class featureprobe.**DatetimePredicate**(*value*)

Bases: Predicate

An enumeration.

AFTER = 'after'

BEFORE = 'before'

class featureprobe.**Detail**(*value=None*, *rule_index: Optional[int] = None*, *version: Optional[int] = None*, *reason: Optional[str] = None*)

Bases: object

class featureprobe.**Distribution**(*distribution: List[List[int]]*)

Bases: object

property distribution

class featureprobe.**EvaluationResult**(*value*, *rule_index: Optional[int]*, *variation_index: Optional[int]*, *version: int*, *reason: str*)

Bases: object

class featureprobe.**HitResult**(*hit: bool*, *index: Optional[int] = None*, *reason: Optional[str] = None*)

Bases: object

property hit

property index

property reason

featureprobe.**HttpConfig**(**args*, ***kwargs*)

class featureprobe.**NumberPredicate**(*value*)

Bases: Predicate

An enumeration.

EQUAL = '='

GREATER_OR_EQUAL = '>='

GREATER_THAN = '>'

LESS_OR_EQUAL = '<='

LESS_THAN = '<'

NOT_EQUAL = '!='

class featureprobe.**Repository**(*toggles: Optional[Dict[str,* Toggle*]] = None*, *segments: Optional[Dict[str,* Segment*]] = None*)

Bases: object

classmethod from_json(*json_*)

property segments: Dict[str, *Segment*]

property toggles: Dict[str, *Toggle*]

class featureprobe.**Rule**(*serve: Optional[Serve] = None*, *conditions: Optional[List[Condition]] = None*)

Bases: object

property conditions: List[*Condition*]

classmethod **from_json**(*json_*)

hit(*user:* User, *segments: Dict[str,* Segment*]*, *toggle_key: str*) → *HitResult*

property serve: *Serve*

class featureprobe.**Segment**(*uid: str*, *version: int*, *rules: Optional[List[SegmentRule]] = None*)

Bases: object

contains(*user:* User, *segments: Dict[str,* Segment*]*)

classmethod **from_json**(*json_*)

property rules: List[*SegmentRule*]

property uid

property version: int

class featureprobe.**SegmentPredicate**(*value*)

Bases: Predicate

An enumeration.

IS_IN = 'is in'

IS_NOT_IN = 'is not in'

class featureprobe.**SegmentRule**(*conditions: Optional[List[Condition]] = None*)

Bases: object

property conditions: List[*Condition*]

classmethod **from_json**(*json_*)

hit(*user:* User, *segments: Dict[str,* Segment*]*) → *HitResult*

class featureprobe.**SemverPredicate**(*value*)

Bases: Predicate

An enumeration.

EQUAL = '='

GREATER_OR_EQUAL = '>='

GREATER_THAN = '>'

LESS_OR_EQUAL = '<='

LESS_THAN = '<'

NOT_EQUAL = '!='

class featureprobe.**Serve**(*select: int*, *split:* Split)

Bases: object

eval_index(*user:* User, *toggle_key: str*) → *HitResult*

classmethod from_json(*json_*)

property select: int

property split: *Split*

class featureprobe.**Split**(*distribution: List[List[List[int]]]*, *bucket_by: str*, *salt: str*)

Bases: object

property bucket_by: str

property distribution: List[List[List[int]]]

find_index(*user:* User, *toggle_key: str*) → *HitResult*

classmethod from_json(*json_*)

property salt: str

class featureprobe.**StringPredicate**(*value*)

Bases: Predicate

An enumeration.

CONTAINS = 'contains'

DOES_NOT_CONTAIN = 'does not contain'

DOES_NOT_END_WITH = 'does not end with'

DOES_NOT_MATCH_REGEX = 'does not match regex'

DOES_NOT_START_WITH = 'does not start with'

ENDS_WITH = 'ends with'

IS_NOT_ANY_OF = 'is not any of'

IS_ONE_OF = 'is one of'

MATCHES_REGEX = 'matches regex'

STARTS_WITH = 'starts with'

class featureprobe.**Toggle**(*key: str*, *enabled: bool*, *version: int*, *disabled_serve:* Serve, *default_serve:* Serve, *rules: List[*Rule*]*, *variations: list*, *for_client: bool*)

Bases: object

property default_serve: *Serve*

property disabled_serve: *Serve*

property enabled: bool

eval(*user:* User, *segments: Dict[str,* Segment*]*, *default_value: object*) → *EvaluationResult*

property for_client: bool

classmethod from_json(*json_*)

property key: str

property rules: List[*Rule*]

property variations: List[str]

property version: int

class featureprobe.**User**(*attrs: Optional[Dict[str, str]] = None*, *stable_rollout_key: Optional[str] = None*)

Bases: object

A collection of attributes that can affect toggle evaluation.

Usually corresponding to a user of your application.

property attrs: Dict[str, str]

Gets all attributes of a FeatureProbe User

has_attr(*attr: str*) → bool

Checks if an attribute exists.

Parameters
attr – Attribute name / key.

Returns
bool

property key: str

Gets FeatureProbe User unique identifier

stable_rollout(*key*)

to_dict() → dict

with_attr(*key: str*, *value: str*) → *User*

Adds an attribute to the user.

Parameters

- **key** – Attribute key / name.
- **value** – Attribute value.

Returns
User

Usage:

```
>>> import featureprobe as fp
>>> user = fp.User('unique id').with_attr('key1', 'value1').with_attr('key2',
↪'value2')
```

1.3 Contributing

We are working on continue evolving FeatureProbe core, making it flexible and easier to use. Development of FeatureProbe happens in the open on GitHub, and we are grateful to the community for contributing bugfixes and improvements.

Please read CONTRIBUTING for details on our code of conduct, and the process for taking part in improving FeatureProbe.

- Bug Report
- Feature Request

PYTHON MODULE INDEX

f

INDEX

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W