surveydata

Release 0.1.12

Orange Chair Labs LLC

Dec 21, 2022

REFERENCE

The `surveydata` Python package offers flexible access to survey data and support for multiple local and cloud storage options.

CHAPTER

ONE

INSTALLATION

Installing the latest version with pip:

```
pip install surveydata
```

CHAPTER

TWO

OVERVIEW

To use the `surveydata` package, you access data from specific survey platforms via an appropriate `SurveyPlatform` object:

- `SurveyCTOPlatform` provides support for SurveyCTO data, including methods to process text audits and submit submission updates via the review and correction workflow (in support of SurveyCTO's machine learning roadmap, with the ml4qc project)
- *Support for more survey platforms is coming!* Reach out if you have a particular need or are willing to contribute.

All survey data must be stored somewhere, and storage is handled via an appropriate `StorageSystem` object:

- `FileStorage` provides support for local file storage
- `S3Storage` provides support for AWS S3 storage
- `DynamoDBStorage` provides support for AWS DynamoDB storage
- `GoogleCloudStorage` provides support for Google Cloud Storage
- `AzureBlobStorage` provides support for Azure Blob Storage
- `SurveyCTOExportStorage` provides support for local exports from SurveyCTO Desktop

In general, the workflow goes like this:

1. Initialize the survey platform
2. Initialize one or more storage systems
3. Synchronize data between the survey platform and the storage system(s) to ensure that data in storage is fully up-to-date (except for static export storage, via a class like `SurveyCTOExportStorage`, which doesn't support synchronization)
4. Load data and/or attachments via the survey platform and storage API's
5. Optionally: Save processed data and then, later, load it back again, for cases where ingestion and processing tasks are separated from actual analysis or use

CHAPTER

THREE

EXAMPLES

See this example notebook for a series of usage examples.

3.1 surveydata

3.1.1 surveydata package

3.1.1.1 Submodules

3.1.1.1.1 surveydata.azureblobstorage module

Support for Azure Blob Storage survey data storage.

class surveydata.azureblobstorage.**AzureBlobStorage**(*container_name: str*, *blob_name_prefix: str*, *connection_string: Optional[str] = None*, *account_url: Optional[str] = None*)

Bases: *StorageSystem*

Azure Blob Storage survey data storage implementation.

__init__(*container_name: str*, *blob_name_prefix: str*, *connection_string: Optional[str] = None*, *account_url: Optional[str] = None*)

Initialize Azure Blob Storage for survey data.

Parameters

- **container_name** (*str*) – Azure Storage container name (must already exist)
- **blob_name_prefix** (*str*) – Prefix to use for all blob names (e.g., "Surveys/Form123/")
- **connection_string** (*str*) – If connecting via connection string, the connection string to use
- **account_url** (*str*) – If connecting via manual (prior) authentication, account URL to use, like https://<storageaccountname>.blob.core.windows.net

attachment_object_name(*submission_id: str*, *attachment_name: str*) → str

Get attachment object name for specific attachment.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **attachment_name** (*str*) – Attachment filename

Returns
Object name for submission

Return type
str

`attachments_supported`() → bool

Query whether storage system supports attachments.

Returns
True if attachments supported, otherwise False

Return type
bool

`get_attachment`(*attachment_location: str = ''*, *submission_id: str = ''*, *attachment_name: str = ''*) → BinaryIO

Get submission attachment from storage.

Parameters

- **attachment_location** (*str*) – Attachment location string (as returned when attachment stored)
- **submission_id** (*str*) – Unique submission ID (in lieu of attachment_location)
- **attachment_name** (*str*) – Attachment filename (in lieu of attachment_location)

Returns
Attachment as file-like object (though, note: it doesn't support seeking)

Return type
BinaryIO

Must pass either attachment_location or both submission_id and attachment_name.

`get_metadata`(*metadata_id: str*) → str

Get metadata string from storage.

Parameters
metadata_id (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)

Returns
Metadata string from storage, or empty string if no such metadata exists

Return type
str

`get_metadata_binary`(*metadata_id: str*) → bytes

Get metadata bytes from storage.

Parameters
metadata_id (*str*) – Unique metadata ID (should not conflict with any submission ID)

Returns
Metadata bytes from storage, or empty bytes array if no such metadata exists

Return type
bytes

get_submission(*submission_id: str*) → dict

Get submission data from storage.

Parameters
submission_id (*str*) – Unique submission ID

Returns
Submission data (or empty dictionary if submission not found)

Return type
dict

list_attachments(*submission_id: str = ''*) → list

List all attachments currently in storage.

Parameters
submission_id (*str*) – Optional submission ID, to list only attachments for specific submission

Returns
List of attachments, each as dict with name, submission_id, and location_string

Return type
list

list_submissions() → list

List all submissions currently in storage.

Returns
List of submission IDs

Return type
list

query_attachment(*attachment_location: str = ''*, *submission_id: str = ''*, *attachment_name: str = ''*) → bool

Query whether specific submission attachment exists in storage.

Parameters

- **attachment_location** (*str*) – Attachment location string (as returned when attachment stored)
- **submission_id** (*str*) – Unique submission ID (in lieu of attachment_location)
- **attachment_name** (*str*) – Attachment filename (in lieu of attachment_location)

Returns
True if submission exists in storage; otherwise False

Return type
bool

Must pass either attachment_location or both submission_id and attachment_name.

query_submission(*submission_id: str*) → bool

Query whether specific submission exists in storage.

Parameters
submission_id (*str*) – Unique submission ID

Returns
True if submission exists in storage; otherwise False

Return type
bool

store_attachment(*submission_id: str*, *attachment_name: str*, *attachment_data: BinaryIO*) → str

Store submission attachment in storage.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **attachment_name** (*str*) – Attachment filename
- **attachment_data** (*BinaryIO*) – File-type object containing the attachment data

Returns
Location string for stored attachment

Return type
str

store_metadata(*metadata_id: str*, *metadata: str*)

Store metadata string in storage.

Parameters

- **metadata_id** (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)
- **metadata** (*str*) – Metadata string to store

store_metadata_binary(*metadata_id: str*, *metadata: bytes*)

Store metadata bytes in storage.

Parameters

- **metadata_id** (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)
- **metadata** (*bytes*) – Metadata bytes to store

store_submission(*submission_id: str*, *submission_data: dict*)

Store submission data in storage.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **submission_data** (*dict*) – Submission data to store

submission_id(*object_name: str*) → str

Get submission ID from object name.

Parameters
object_name (*str*) – Object name (e.g., from submission_object_name())

Returns
Submission ID

Return type
str

submission_id_and_attachment_name(*object_name: str) -> (<class 'str'>*, *<class 'str'>*)

Get submission ID and attachment name from object name.

Parameters
object_name (*str*) – Object name (e.g., from submission_object_name())

Returns
Submission ID and attachment name

Return type
(str, str)

submission_object_name(*submission_id: str*) → str

Get submission object name for specific submission.

Parameters
submission_id (*str*) – Unique submission ID

Returns
Object name for submission

Return type
str

3.1.1.1.2 surveydata.dynamodbstorage module

Support for AWS DynamoDB survey data storage.

class surveydata.dynamodbstorage.**DynamoDBStorage**(*aws_region: str*, *table_name: str*, *id_field_name: str*, *partition_key_name: str = ''*, *partition_key_value: str = ''*, *aws_access_key_id: Optional[str] = None*, *aws_secret_access_key: Optional[str] = None*, *aws_session_token: Optional[str] = None*)

Bases: *StorageSystem*

AWS DynamoDB survey data storage implementation.

__init__(*aws_region: str*, *table_name: str*, *id_field_name: str*, *partition_key_name: str = ''*, *partition_key_value: str = ''*, *aws_access_key_id: Optional[str] = None*, *aws_secret_access_key: Optional[str] = None*, *aws_session_token: Optional[str] = None*)

Initialize DynamoDB storage for survey data.

Parameters

- **aws_region** (*str*) – AWS region to use
- **table_name** (*str*) – DynamoDB table name (must already exist)
- **id_field_name** (*str*) – Field name for unique submission ID (e.g., "KEY")
- **partition_key_name** (*str*) – Partition key name for optional fixed partition (e.g., "FormID")
- **partition_key_value** (*str*) – Partition value for optional fixed partition (e.g., form ID)
- **aws_access_key_id** (*str*) – AWS access key ID; if None, will use local config file and/or environment vars
- **aws_secret_access_key** (*str*) – AWS access key secret; if None, will use local config file and/or environment vars
- **aws_session_token** (*str*) – AWS session token to use, only if using temporary credentials

The DynamoDB table should already exist with the primary key configured in one of two ways:

1. a fixed partition key with the name passed as partition_key_name, and the sort key with the name passed as id_field_name; or
2. a partition key with the name passed as id_field_name (and no sort key).

`attachments_supported`() → bool

Query whether storage system supports attachments.

Returns
True if attachments supported, otherwise False

Return type
bool

`get_attachment`(*attachment_location: str = ''*, *submission_id: str = ''*, *attachment_name: str = ''*) → BinaryIO

Get submission attachment from storage.

Parameters

- **attachment_location** (*str*) – Attachment location string (as returned when attachment stored)
- **submission_id** (*str*) – Unique submission ID (in lieu of attachment_location)
- **attachment_name** (*str*) – Attachment filename (in lieu of attachment_location)

Returns
Attachment as file-like object (though, note: it doesn't support seeking)

Return type
BinaryIO

Must pass either attachment_location or both submission_id and attachment_name.

`get_metadata`(*metadata_id: str*) → str

Get metadata string from storage.

Parameters
metadata_id (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)

Returns
Metadata string from storage, or empty string if no such metadata exists

Return type
str

`get_metadata_binary`(*metadata_id: str*) → bytes

Get metadata bytes from storage.

Parameters
metadata_id (*str*) – Unique metadata ID (should not conflict with any submission ID)

Returns
Metadata bytes from storage, or empty bytes array if no such metadata exists

Return type
bytes

get_submission(*submission_id: str*) → dict

Get submission data from storage.

Parameters
submission_id (*str*) – Unique submission ID

Returns
Submission data (or empty dictionary if submission not found)

Return type
dict

list_attachments(*submission_id: str = ''*) → list

List all attachments currently in storage.

Parameters
submission_id (*str*) – Optional submission ID, to list only attachments for specific submission

Returns
List of attachments, each as dict with name, submission_id, and location_string

Return type
list

list_submissions() → list

List all submissions currently in storage.

Returns
List of submission IDs

Return type
list

query_attachment(*attachment_location: str = ''*, *submission_id: str = ''*, *attachment_name: str = ''*) → bool

Query whether specific submission attachment exists in storage.

Parameters

- **attachment_location** (*str*) – Attachment location string (as returned when attachment stored)
- **submission_id** (*str*) – Unique submission ID (in lieu of attachment_location)
- **attachment_name** (*str*) – Attachment filename (in lieu of attachment_location)

Returns
True if submission exists in storage; otherwise False

Return type
bool

Must pass either attachment_location or both submission_id and attachment_name.

query_submission(*submission_id: str*) → bool

Query whether specific submission exists in storage.

Parameters
submission_id (*str*) – Unique submission ID

Returns
True if submission exists in storage; otherwise False

Return type
bool

store_attachment(*submission_id: str*, *attachment_name: str*, *attachment_data: BinaryIO*) → str

Store submission attachment in storage.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **attachment_name** (*str*) – Attachment filename
- **attachment_data** (*BinaryIO*) – File-type object containing the attachment data

Returns
Location string for stored attachment

Return type
str

store_metadata(*metadata_id: str*, *metadata: str*)

Store metadata string in storage.

Parameters

- **metadata_id** (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)
- **metadata** (*str*) – Metadata string to store

store_metadata_binary(*metadata_id: str*, *metadata: bytes*)

Store metadata bytes in storage.

Parameters

- **metadata_id** (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)
- **metadata** (*bytes*) – Metadata bytes to store

store_submission(*submission_id: str*, *submission_data: dict*)

Store submission data in storage.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **submission_data** (*dict*) – Submission data to store

submission_primary_key(*submission_id: str*) → dict

Get submission primary key for specific submission.

Parameters
submission_id (*str*) – Unique submission ID

Returns
Primary key for submission

Return type
dict

3.1.1.1.3 surveydata.filestorage module

Support for local file system survey data storage.

class surveydata.filestorage.**FileStorage**(*submission_path: str*)

Bases: *StorageSystem*

Local file system survey data storage implementation.

__init__(*submission_path: str*)

Initialize local file system storage for survey data.

Parameters
submission_path (*str*) – Globally-unique S3 bucket name (must already exist)

attachment_path(*submission_id: str*, *attachment_name: str*) → str

Get attachment path for specific attachment.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **attachment_name** (*str*) – Attachment filename

Returns
Path for submission

Return type
str

attachments_supported() → bool

Query whether storage system supports attachments.

Returns
True if attachments supported, otherwise False

Return type
bool

get_attachment(*attachment_location: str = ''*, *submission_id: str = ''*, *attachment_name: str = ''*) → BinaryIO

Get submission attachment from storage.

Parameters

- **attachment_location** (*str*) – Attachment location string (as returned when attachment stored)
- **submission_id** (*str*) – Unique submission ID (in lieu of attachment_location)
- **attachment_name** (*str*) – Attachment filename (in lieu of attachment_location)

Returns
Attachment as file-like object (though, note: it doesn't support seeking)

Return type
BinaryIO

Must pass either attachment_location or both submission_id and attachment_name.

get_metadata(*metadata_id: str*) → str

Get metadata string from storage.

Parameters
metadata_id (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)

Returns
Metadata string from storage, or empty string if no such metadata exists

Return type
str

get_metadata_binary(*metadata_id: str*) → bytes

Get metadata bytes from storage.

Parameters
metadata_id (*str*) – Unique metadata ID (should not conflict with any submission ID)

Returns
Metadata bytes from storage, or empty bytes array if no such metadata exists

Return type
bytes

get_submission(*submission_id: str*) → dict

Get submission data from storage.

Parameters
submission_id (*str*) – Unique submission ID

Returns
Submission data (or empty dictionary if submission not found)

Return type
dict

list_attachments(*submission_id: str = ''*) → list

List all attachments currently in storage.

Parameters
submission_id (*str*) – Optional submission ID, to list only attachments for specific submission

Returns
List of attachments, each as dict with name, submission_id, and location_string

Return type
list

list_submissions() → list

List all submissions currently in storage.

Returns
List of submission IDs

Return type
list

query_attachment(*attachment_location: str = ''*, *submission_id: str = ''*, *attachment_name: str = ''*) → bool

Query whether specific submission attachment exists in storage.

Parameters

- **attachment_location** (*str*) – Attachment location string (as returned when attachment stored)

- **submission_id** (*str*) – Unique submission ID (in lieu of attachment_location)
- **attachment_name** (*str*) – Attachment filename (in lieu of attachment_location)

Returns
True if submission exists in storage; otherwise False

Return type
bool

Must pass either attachment_location or both submission_id and attachment_name.

query_submission(*submission_id: str*) → bool

Query whether specific submission exists in storage.

Parameters
submission_id (*str*) – Unique submission ID

Returns
True if submission exists in storage; otherwise False

Return type
bool

store_attachment(*submission_id: str*, *attachment_name: str*, *attachment_data: BinaryIO*) → str

Store submission attachment in storage.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **attachment_name** (*str*) – Attachment filename
- **attachment_data** (*BinaryIO*) – File-type object containing the attachment data

Returns
Location string for stored attachment

Return type
str

store_metadata(*metadata_id: str*, *metadata: str*)

Store metadata string in storage.

Parameters

- **metadata_id** (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)
- **metadata** (*str*) – Metadata string to store

store_metadata_binary(*metadata_id: str*, *metadata: bytes*)

Store metadata bytes in storage.

Parameters

- **metadata_id** (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)
- **metadata** (*bytes*) – Metadata bytes to store

store_submission(*submission_id: str*, *submission_data: dict*)

Store submission data in storage.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **submission_data** (*dict*) – Submission data to store

submission_file_name(*submission_id: str*) → str

Get submission filename for specific submission.

Parameters
submission_id (*str*) – Unique submission ID

Returns
Filename for submission

Return type
str

submission_id(*filename: str*) → str

Get submission ID from filename.

Parameters
filename (*str*) – Filename (e.g., from submission_file_name())

Returns
Submission ID

Return type
str

3.1.1.1.4 surveydata.googlecloudstorage module

Support for Google Cloud Storage survey data storage.

class surveydata.googlecloudstorage.**GoogleCloudStorage**(*project_id: str*, *bucket_name: str*, *blob_name_prefix: str*, *credentials: Optional[Credentials] = None*)

Bases: *StorageSystem*

Google Cloud Storage survey data storage implementation.

__init__(*project_id: str*, *bucket_name: str*, *blob_name_prefix: str*, *credentials: Optional[Credentials] = None*)

Initialize Google Cloud Storage for survey data.

Parameters

- **project_id** (*str*) – Google Cloud Storage project ID
- **bucket_name** (*str*) – Cloud Storage bucket name (must already exist)
- **blob_name_prefix** (*str*) – Prefix to use for all blob names (e.g., "Surveys/Form123/")
- **credentials** (*credentials.Credentials*) – Explicit service account credentials to use (e.g., loaded from service_account.Credentials.from_service_account_file())

attachment_object_name(*submission_id: str*, *attachment_name: str*) → str

Get attachment object name for specific attachment.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **attachment_name** (*str*) – Attachment filename

Returns
Object name for submission

Return type
str

attachments_supported() → bool

Query whether storage system supports attachments.

Returns
True if attachments supported, otherwise False

Return type
bool

get_attachment(*attachment_location: str = ''*, *submission_id: str = ''*, *attachment_name: str = ''*) → BinaryIO

Get submission attachment from storage.

Parameters

- **attachment_location** (*str*) – Attachment location string (as returned when attachment stored)
- **submission_id** (*str*) – Unique submission ID (in lieu of attachment_location)
- **attachment_name** (*str*) – Attachment filename (in lieu of attachment_location)

Returns
Attachment as file-like object (though, note: it doesn't support seeking)

Return type
BinaryIO

Must pass either attachment_location or both submission_id and attachment_name.

get_metadata(*metadata_id: str*) → str

Get metadata string from storage.

Parameters
metadata_id (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)

Returns
Metadata string from storage, or empty string if no such metadata exists

Return type
str

get_metadata_binary(*metadata_id: str*) → bytes

Get metadata bytes from storage.

Parameters
metadata_id (*str*) – Unique metadata ID (should not conflict with any submission ID)

Returns
Metadata bytes from storage, or empty bytes array if no such metadata exists

Return type
bytes

get_submission(*submission_id: str*) → dict

Get submission data from storage.

Parameters
submission_id (*str*) – Unique submission ID

Returns
Submission data (or empty dictionary if submission not found)

Return type
dict

list_attachments(*submission_id: str = ''*) → list

List all attachments currently in storage.

Parameters
submission_id (*str*) – Optional submission ID, to list only attachments for specific submission

Returns
List of attachments, each as dict with name, submission_id, and location_string

Return type
list

list_submissions() → list

List all submissions currently in storage.

Returns
List of submission IDs

Return type
list

query_attachment(*attachment_location: str = ''*, *submission_id: str = ''*, *attachment_name: str = ''*) → bool

Query whether specific submission attachment exists in storage.

Parameters

- **attachment_location** (*str*) – Attachment location string (as returned when attachment stored)
- **submission_id** (*str*) – Unique submission ID (in lieu of attachment_location)
- **attachment_name** (*str*) – Attachment filename (in lieu of attachment_location)

Returns
True if submission exists in storage; otherwise False

Return type
bool

Must pass either attachment_location or both submission_id and attachment_name.

query_submission(*submission_id: str*) → bool

Query whether specific submission exists in storage.

Parameters
submission_id (*str*) – Unique submission ID

Returns
True if submission exists in storage; otherwise False

Return type
bool

store_attachment(*submission_id: str*, *attachment_name: str*, *attachment_data: BinaryIO*) → str

Store submission attachment in storage.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **attachment_name** (*str*) – Attachment filename
- **attachment_data** (*BinaryIO*) – File-type object containing the attachment data

Returns
Location string for stored attachment

Return type
str

store_metadata(*metadata_id: str*, *metadata: str*)

Store metadata string in storage.

Parameters

- **metadata_id** (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)
- **metadata** (*str*) – Metadata string to store

store_metadata_binary(*metadata_id: str*, *metadata: bytes*)

Store metadata bytes in storage.

Parameters

- **metadata_id** (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)
- **metadata** (*bytes*) – Metadata bytes to store

store_submission(*submission_id: str*, *submission_data: dict*)

Store submission data in storage.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **submission_data** (*dict*) – Submission data to store

submission_id(*object_name: str*) → str

Get submission ID from object name.

Parameters
object_name (*str*) – Object name (e.g., from submission_object_name())

Returns
Submission ID

Return type
str

submission_id_and_attachment_name(*object_name: str) -> (<class 'str'>*, *<class 'str'>*)

Get submission ID and attachment name from object name.

Parameters
object_name (*str*) – Object name (e.g., from submission_object_name())

Returns
Submission ID and attachment name

Return type
(str, str)

submission_object_name(*submission_id: str*) → str

Get submission object name for specific submission.

Parameters
submission_id (*str*) – Unique submission ID

Returns
Object name for submission

Return type
str

3.1.1.1.5 surveydata.s3storage module

Support for AWS S3 survey data storage.

class surveydata.s3storage.**S3Storage**(*bucket_name: str*, *key_name_prefix: str*, *aws_access_key_id: Optional[str] = None*, *aws_secret_access_key: Optional[str] = None*, *aws_session_token: Optional[str] = None*)

Bases: *StorageSystem*

AWS S3 survey data storage implementation.

__init__(*bucket_name: str*, *key_name_prefix: str*, *aws_access_key_id: Optional[str] = None*, *aws_secret_access_key: Optional[str] = None*, *aws_session_token: Optional[str] = None*)

Initialize S3 storage for survey data.

Parameters

- **bucket_name** (*str*) – Globally-unique S3 bucket name (must already exist)
- **key_name_prefix** (*str*) – Prefix to use for all key names (e.g., "Surveys/Form123/")
- **aws_access_key_id** (*str*) – AWS access key ID; if None, will use local config file and/or environment vars
- **aws_secret_access_key** (*str*) – AWS access key secret; if None, will use local config file and/or environment vars
- **aws_session_token** (*str*) – AWS session token to use, only if using temporary credentials

attachment_object_name(*submission_id: str*, *attachment_name: str*) → str

Get attachment object name for specific attachment.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **attachment_name** (*str*) – Attachment filename

Returns
Object name for submission

Return type
str

attachments_supported() → bool

Query whether storage system supports attachments.

Returns
True if attachments supported, otherwise False

Return type
bool

get_attachment(*attachment_location: str = ''*, *submission_id: str = ''*, *attachment_name: str = ''*) → BinaryIO

Get submission attachment from storage.

Parameters

- **attachment_location** (*str*) – Attachment location string (as returned when attachment stored)
- **submission_id** (*str*) – Unique submission ID (in lieu of attachment_location)
- **attachment_name** (*str*) – Attachment filename (in lieu of attachment_location)

Returns
Attachment as file-like object (though, note: it doesn't support seeking)

Return type
BinaryIO

Must pass either attachment_location or both submission_id and attachment_name.

get_metadata(*metadata_id: str*) → str

Get metadata string from storage.

Parameters
metadata_id (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)

Returns
Metadata string from storage, or empty string if no such metadata exists

Return type
str

get_metadata_binary(*metadata_id: str*) → bytes

Get metadata bytes from storage.

Parameters
metadata_id (*str*) – Unique metadata ID (should not conflict with any submission ID)

Returns
Metadata bytes from storage, or empty bytes array if no such metadata exists

Return type
bytes

get_submission(*submission_id: str*) → dict

Get submission data from storage.

Parameters
submission_id (*str*) – Unique submission ID

Returns
Submission data (or empty dictionary if submission not found)

Return type
dict

`list_attachments`(*submission_id: str = ''*) → list

List all attachments currently in storage.

Parameters
submission_id (*str*) – Optional submission ID, to list only attachments for specific submission

Returns
List of attachments, each as dict with name, submission_id, and location_string

Return type
list

`list_submissions`() → list

List all submissions currently in storage.

Returns
List of submission IDs

Return type
list

`query_attachment`(*attachment_location: str = ''*, *submission_id: str = ''*, *attachment_name: str = ''*) → bool

Query whether specific submission attachment exists in storage.

Parameters

- **attachment_location** (*str*) – Attachment location string (as returned when attachment stored)
- **submission_id** (*str*) – Unique submission ID (in lieu of attachment_location)
- **attachment_name** (*str*) – Attachment filename (in lieu of attachment_location)

Returns
True if submission exists in storage; otherwise False

Return type
bool

Must pass either attachment_location or both submission_id and attachment_name.

`query_submission`(*submission_id: str*) → bool

Query whether specific submission exists in storage.

Parameters
submission_id (*str*) – Unique submission ID

Returns
True if submission exists in storage; otherwise False

Return type
bool

`store_attachment`(*submission_id: str*, *attachment_name: str*, *attachment_data: BinaryIO*) → str

Store submission attachment in storage.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **attachment_name** (*str*) – Attachment filename
- **attachment_data** (*BinaryIO*) – File-type object containing the attachment data

Returns
Location string for stored attachment

Return type
str

store_metadata(*metadata_id: str*, *metadata: str*)

Store metadata string in storage.

Parameters

- **metadata_id** (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)
- **metadata** (*str*) – Metadata string to store

store_metadata_binary(*metadata_id: str*, *metadata: bytes*)

Store metadata bytes in storage.

Parameters

- **metadata_id** (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)
- **metadata** (*bytes*) – Metadata bytes to store

store_submission(*submission_id: str*, *submission_data: dict*)

Store submission data in storage.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **submission_data** (*dict*) – Submission data to store

submission_id(*object_name: str*) → str

Get submission ID from object name.

Parameters
object_name (*str*) – Object name (e.g., from submission_object_name())

Returns
Submission ID

Return type
str

submission_id_and_attachment_name(*object_name: str) -> (<class 'str'>*, *<class 'str'>*)

Get submission ID and attachment name from object name.

Parameters
object_name (*str*) – Object name (e.g., from submission_object_name())

Returns
Submission ID and attachment name

Return type
(str, str)

submission_object_name(*submission_id: str*) → str

Get submission object name for specific submission.

Parameters
submission_id (*str*) – Unique submission ID

Returns
Object name for submission

Return type
str

3.1.1.1.6 surveydata.storagesystem module

Core interface (informal) for survey data storage systems.

class surveydata.storagesystem.**StorageSystem**

Bases: object

Largely-abstract base class for survey data storage systems.

__init__()

Initialize storage system.

attachments_supported() → bool

Query whether storage system supports attachments.

Returns
True if attachments supported, otherwise False

Return type
bool

get_attachment(*attachment_location: str = ''*, *submission_id: str = ''*, *attachment_name: str = ''*) → BinaryIO

Get submission attachment from storage.

Parameters

- **attachment_location** (*str*) – Attachment location string (as returned when attachment stored)
- **submission_id** (*str*) – Unique submission ID (in lieu of attachment_location)
- **attachment_name** (*str*) – Attachment filename (in lieu of attachment_location)

Returns
Attachment as file-like object (though, note: it doesn't support seeking)

Return type
BinaryIO

Must pass either attachment_location or both submission_id and attachment_name.

get_data_timezone() → timezone

Get the timezone for timestamps in the data.

Returns
Timezone for timestamps in the data (defaults to datetime.timezone.utc if unknown)

Return type
datetime.timezone

get_dataframe(*metadata_id: str*) → DataFrame

Get Pandas DataFrame from a binary file in storage.

Parameters
metadata_id (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)

Returns
Metadata string from storage, or empty string if no such metadata exists

Return type
pd.DataFrame

get_dataframe_csv(*metadata_id: str*) → DataFrame

Get Pandas DataFrame from a .csv file in storage.

Parameters
metadata_id (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)

Returns
Metadata string from storage, or empty string if no such metadata exists

Return type
pd.DataFrame

get_metadata(*metadata_id: str*) → str

Get metadata string from storage.

Parameters
metadata_id (*str*) – Unique metadata ID (should not conflict with any submission ID)

Returns
Metadata string from storage, or empty string if no such metadata exists

Return type
str

get_metadata_binary(*metadata_id: str*) → bytes

Get metadata bytes from storage.

Parameters
metadata_id (*str*) – Unique metadata ID (should not conflict with any submission ID)

Returns
Metadata bytes from storage, or empty bytes array if no such metadata exists

Return type
bytes

get_submission(*submission_id: str*) → dict

Get submission data from storage.

Parameters
submission_id (*str*) – Unique submission ID

Returns
Submission data (or empty dictionary if submission not found)

Return type
dict

get_submissions() → list

Get all submission data from storage.

Returns
List of dictionaries, one for each submission

Return type
list

get_submissions_df() → DataFrame

Get all submission data from storage, organized into a Pandas DataFrame.

Returns
Pandas DataFrame containing all submissions currently in storage

Return type
pandas.DataFrame

list_attachments(*submission_id: str = ''*) → list

List all attachments currently in storage.

Parameters
submission_id (*str*) – Optional submission ID, to list only attachments for specific submission

Returns
List of attachments, each as dict with name, submission_id, and location_string

Return type
list

list_submissions() → list

List all submissions currently in storage.

Returns
List of submission IDs

Return type
list

query_attachment(*attachment_location: str = ''*, *submission_id: str = ''*, *attachment_name: str = ''*) → bool

Query whether specific submission attachment exists in storage.

Parameters

- **attachment_location** (*str*) – Attachment location string (as returned when attachment stored)
- **submission_id** (*str*) – Unique submission ID (in lieu of attachment_location)
- **attachment_name** (*str*) – Attachment filename (in lieu of attachment_location)

Returns
True if submission exists in storage; otherwise False

Return type
bool

query_submission(*submission_id: str*) → bool

Query whether specific submission exists in storage.

Parameters
submission_id (*str*) – Unique submission ID

Returns
True if submission exists in storage; otherwise False

Return type
bool

`set_data_timezone`(*tz: timezone*)

Set the timezone for timestamps in the data.

Parameters
tz (*datetime.timezone*) – Timezone for timestamps in the data

`store_attachment`(*submission_id: str*, *attachment_name: str*, *attachment_data: BinaryIO*) → str

Store submission attachment in storage.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **attachment_name** (*str*) – Attachment filename
- **attachment_data** (*BinaryIO*) – File-type object containing the attachment data

Returns
Location string for stored attachment

Return type
str

`store_dataframe`(*metadata_id: str*, *df: DataFrame*)

Store Pandas DataFrame as binary file in storage.

Parameters

- **metadata_id** (*str*) – Unique metadata ID to save as (should begin and end with __ and not conflict with any submission ID)
- **df** (*pd.DataFrame*) – Pandas DataFrame to store as binary file

`store_dataframe_csv`(*metadata_id: str*, *df: DataFrame*)

Store Pandas DataFrame as .csv file in storage.

Parameters

- **metadata_id** (*str*) – Unique metadata ID to save as (should begin and end with __ and not conflict with any submission ID)
- **df** (*pd.DataFrame*) – Pandas DataFrame to store as .csv file

`store_metadata`(*metadata_id: str*, *metadata: str*)

Store metadata string in storage.

Parameters

- **metadata_id** (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)
- **metadata** (*str*) – Metadata string to store

`store_metadata_binary`(*metadata_id: str*, *metadata: bytes*)

Store metadata bytes in storage.

Parameters

- **metadata_id** (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)
- **metadata** (*bytes*) – Metadata bytes to store

store_submission(*submission_id: str*, *submission_data: dict*)

Store submission data in storage.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **submission_data** (*dict*) – Submission data to store

3.1.1.1.7 surveydata.surveyctoexportstorage module

Read-only support for SurveyCTO survey data exports.

class surveydata.surveyctoexportstorage.**SurveyCTOExportStorage**(*export_file: str*, *attachments_available: bool*, *data_timezone: Optional[timezone] = None*)

Bases: *StorageSystem*

Implementation of storage interface for read-only access to SurveyCTO survey data exports.

__init__(*export_file: str*, *attachments_available: bool*, *data_timezone: Optional[timezone] = None*)

Initialize SurveyCTO export data.

Parameters

- **export_file** (*str*) – Path to the export file
- **attachments_available** (*bool*) – True if attachments available from SurveyCTO Desktop (in media subfolder)
- **data_timezone** (*datetime.timezone*) – Timezone for timestamps in the data (defaults to current timezone if not specified)

attachments_supported() → bool

Query whether storage system supports attachments.

Returns
True if attachments supported, otherwise False

Return type
bool

get_attachment(*attachment_location: str = ''*, *submission_id: str = ''*, *attachment_name: str = ''*) → BinaryIO

Get submission attachment from storage.

Parameters

- **attachment_location** (*str*) – Attachment location string (as exported by SurveyCTO Desktop)
- **submission_id** (*str*) – Unique submission ID (in lieu of attachment_location)
- **attachment_name** (*str*) – Attachment filename (in lieu of attachment_location)

Returns
Attachment as file-like object

Return type
BinaryIO

Must pass either attachment_location or both submission_id and attachment_name.

`get_data_timezone()` → timezone

Get the timezone for timestamps in the data.

Returns
Timezone for timestamps in the data (defaults to datetime.timezone.utc if unknown)

Return type
datetime.timezone

`get_metadata(metadata_id: str)` → str

Get metadata string from storage.

Parameters
metadata_id (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)

Returns
Metadata string from storage, or empty string if no such metadata exists

Return type
str

`get_metadata_binary(metadata_id: str)` → bytes

Get metadata bytes from storage.

Parameters
metadata_id (*str*) – Unique metadata ID (should not conflict with any submission ID)

Returns
Metadata bytes from storage, or empty bytes array if no such metadata exists

Return type
bytes

`get_submission(submission_id: str)` → dict

Get submission data from storage.

Parameters
submission_id (*str*) – Unique submission ID

Returns
Submission data (or empty dictionary if submission not found)

Return type
dict

`get_submissions()` → list

Get all submission data from storage.

Returns
List of dictionaries, one for each submission

Return type
list

list_attachments(*submission_id: str = ''*) → list

List all attachments currently in storage.

Parameters
submission_id (*str*) – Optional submission ID, to list only attachments for specific submission

Returns
List of attachments, each as dict with name, submission_id, and location_string

Return type
list

list_submissions() → list

List all submissions currently in storage.

Returns
List of submission IDs

Return type
list

query_attachment(*attachment_location: str = ''*, *submission_id: str = ''*, *attachment_name: str = ''*) → bool

Query whether specific submission attachment exists in storage.

Parameters

- **attachment_location** (*str*) – Attachment location string (as exported by SurveyCTO Desktop)
- **submission_id** (*str*) – Unique submission ID (in lieu of attachment_location)
- **attachment_name** (*str*) – Attachment filename (in lieu of attachment_location)

Returns
True if submission exists in storage; otherwise False

Return type
bool

Must pass either attachment_location or both submission_id and attachment_name.

query_submission(*submission_id: str*) → bool

Query whether specific submission exists in storage.

Parameters
submission_id (*str*) – Unique submission ID

Returns
True if submission exists in storage; otherwise False

Return type
bool

set_data_timezone(*tz: timezone*)

Set the timezone for timestamps in the data.

Parameters
tz (*datetime.timezone*) – Timezone for timestamps in the data

store_attachment(*submission_id: str*, *attachment_name: str*, *attachment_data: BinaryIO*) → str

Store submission attachment in storage.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **attachment_name** (*str*) – Attachment filename
- **attachment_data** (*BinaryIO*) – File-type object containing the attachment data

Returns
Location string for stored attachment

Return type
str

store_metadata(*metadata_id: str*, *metadata: str*)

Store metadata string in storage.

Parameters

- **metadata_id** (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)
- **metadata** (*str*) – Metadata string to store

store_metadata_binary(*metadata_id: str*, *metadata: bytes*)

Store metadata bytes in storage.

Parameters

- **metadata_id** (*str*) – Unique metadata ID (should begin and end with __ and not conflict with any submission ID)
- **metadata** (*bytes*) – Metadata bytes to store

store_submission(*submission_id: str*, *submission_data: dict*)

Store submission data in storage.

Parameters

- **submission_id** (*str*) – Unique submission ID
- **submission_data** (*dict*) – Submission data to store

3.1.1.1.8 surveydata.surveyctoplatform module

Support for SurveyCTO as a survey data platform.

class surveydata.surveyctoplatform.**SurveyCTOPlatform**(*server: str = ''*, *username: str = ''*, *password: str = ''*, *formid: str = ''*, *private_key: str = ''*)

Bases: *SurveyPlatform*

SurveyCTO survey data platform implementation.

__init__(*server: str = ''*, *username: str = ''*, *password: str = ''*, *formid: str = ''*, *private_key: str = ''*)

Initialize SurveyCTO for access to survey data.

Parameters

- **server** (*str*) – SurveyCTO server name (like "use", without the https prefix or .surveycto.com suffix)
- **username** (*str*) – Email address for API access
- **password** (*str*) – Password for API access
- **formid** (*str*) – SurveyCTO form ID

- **private_key** (*str*) – Full text of private key, if using encryption

If you're not going to call sync_data(), you don't need to supply any of the parameters to this constructor.

static get_submissions_df(*storage:* StorageSystem) → DataFrame

Get all submission data from storage, organized into a Pandas DataFrame and optimized based on the platform.

Parameters
storage (StorageSystem) – Storage system for submissions

Returns
Pandas DataFrame containing all submissions currently in storage

Return type
pandas.DataFrame

static get_text_audit_df(*storage:* StorageSystem, *location_string: str = ''*, *location_strings: Optional[Series] = None*) → DataFrame

Get one or more text audits from storage, organized into a Pandas DataFrame.

Parameters

- **storage** (StorageSystem) – Storage system for attachments
- **location_string** (*str*) – Location string of single text audit to load
- **location_strings** (*pandas.Series*) – Series of location strings of text audits to load

Returns
DataFrame with either the single text audit contents or all text audit contents indexed by Series index

Return type
pandas.DataFrame

Pass either a single location_string or a Series of location_strings.

static process_text_audits(*ta_df: DataFrame*, *start_times: Optional[Series] = None*, *end_times: Optional[Series] = None*, *data_tz: Optional[timezone] = None*, *collection_tz: Optional[timezone] = None*) → DataFrame

Process text audits by summarizing, transforming, and reshaping into a single row per submission.

Parameters

- **ta_df** (*pd.DataFrame*) – DataFrame with raw text audit data, typically from get_text_audit_df()
- **start_times** (*pd.Series*) – Pandas Series with a starting date and time for each submission (indexed by submission ID)
- **end_times** (*pd.Series*) – Pandas Series with an ending date and time for each submission (indexed by submission ID)
- **data_tz** (*datetime.timezone*) – Timezone of timestamps in start_times and end_times
- **collection_tz** (*datetime.timezone*) – Timezone of data collection

Returns
Pandas DataFrame, indexed by submission ID, with summary details as well as field-by-field visit summaries

Return type
pd.DataFrame

The returned DataFrame is indexed by submission ID and includes the following columns:

- **ta_duration_total** - Total duration spent in form fields (ms); feature engineering recommendation: divide by max to rescale to 0-1
- **ta_duration_mean** - Mean duration spent in form fields (ms); feature engineering recommendation: divide by max to rescale to 0-1
- **ta_duration_sd** - Standard deviation of duration spent in form fields (ms); feature engineering recommendation: divide by max to rescale to 0-1
- **ta_duration_min** - Min duration spent in form field (ms); feature engineering recommendation: divide by max to rescale to 0-1
- **ta_duration_max** - Max duration spent in form field (ms); feature engineering recommendation: divide by max to rescale to 0-1
- **ta_fields** - Number of fields visited; feature engineering recommendation: divide by max to rescale to 0-1
- **ta_time_in_fields** - Percent of overall calendar time spent in fields; feature engineering recommendation: leave as 0-1 scale (but note that rounding errors and device clock issues can result in values outside (0, 1))
- **ta_sessions** - Number of form-filling sessions (always 1 unless eventlog-level text audit data); feature engineering recommendation: divide by max to rescale to 0-1
- **ta_pct_revisits** - Percent of field visits that are revisits (always 0 unless eventlog-level text audit data); feature engineering recommendation: leave as 0-1 scale
- **ta_start_dayofweek** - Day of week submission started (0 for Sunday, only available if eventlog text audit data or timezone information supplied); feature engineering recommendation: one-hot encode
- **ta_start_hourofday** - Hour of day submission started (only available if eventlog text audit data or timezone information supplied); feature engineering recommendation: one-hot encode
- **ta_field_x_visited** - 1 if field x visited, otherwise 0; feature engineering recommendation: leave as 0-1 scale, fill missing with 0
- **ta_field_x_visit_y_start** - When field x was visited the yth time divided by *ta_total_duration*, otherwise 0; feature engineering recommendation: leave as 0-1 scale, fill missing with 0
- **ta_field_x_visit_y_duration** - Time spent on field x the yth time it was visited divided by *ta_total_duration*, otherwise 0 (i.e., percentage of overall form time spent on the field visit); feature engineering recommendation: leave as 0-1 scale, fill missing with 0 (or divide by max to rescale to full 0-1 range)

sync_data(*storage: StorageSystem*, *attachment_storage: Optional[StorageSystem] = None*, *no_attachments: bool = False*, *review_statuses: Optional[list] = None*) → list

Sync survey data to storage system.

Parameters

- **storage** (StorageSystem) – Storage system for submissions (and attachments, if supported and other options don't override)
- **attachment_storage** (StorageSystem) – Separate storage system for attachments (only if needed)
- **no_attachments** (*bool*) – True to not sync attachments
- **review_statuses** (*list*) – List of review statuses to include (any combo of "approved", "pending", "rejected"; if not specified, syncs only approved submissions)

Returns
List of new submissions stored (submission ID strings)

Return type
list

`update_submissions`(*submission_updates: list*)

Submit one or more submission updates, including reviews, classifications, and/or comments.

Parameters
submission_updates (*list*) – List of dictionaries with one per update; each should include values for "submissionID"; "reviewStatus" ("none", "approved", or "rejected"); "qualityClassification" ("good", "okay", "poor", or "fake"); and/or "comment" (custom text)

Warning: this method uses an undocumented SurveyCTO API that may break in future SurveyCTO releases.

3.1.1.1.9 surveydata.surveyplatform module

Core interface (informal) for survey data platforms.

`class surveydata.surveyplatform.SurveyPlatform`

Bases: `object`

Abstract base class (informal) for survey data platforms.

`__init__`()

Initialize survey platform for access to survey data.

`static get_submissions_df`(*storage:* StorageSystem) → DataFrame

Get all submission data from storage, organized into a Pandas DataFrame and optimized based on the platform.

Parameters
storage (StorageSystem) – Storage system for submissions

Returns
Pandas DataFrame containing all submissions currently in storage

Return type
pandas.DataFrame

`sync_data`(*storage:* StorageSystem, *attachment_storage: Optional[*StorageSystem*] = None*, *no_attachments: bool = False*) → list

Sync survey data to storage system.

Parameters

- **storage** (StorageSystem) – Storage system for submissions (and attachments, if supported and other options don't override)
- **attachment_storage** (StorageSystem) – Separate storage system for attachments (only if needed)
- **no_attachments** (*bool*) – True to not sync attachments

Returns
List of new submissions stored (submission ID strings)

Return type
list

3.1.1.2 Module contents

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

Symbols

A

D

F

G

L

M

P

Q

S

U