tdxlib

Jul 07, 2023

Contents:

TDXLib is a suite of Python libraries that can interact with the TeamDynamix Web API.

CHAPTER 1

Quick Links

- GitHub Repository (includes quick-start guide)
- Teamdynamix Web API Documentation

1.1 TDX Integration

This class contains all the methods that work with the following TDX API endpoints:

- Authentication
- Locations
- Rooms
- People
- Groups
- Accounts
- Custom attributes

1.1.1 Class & Methods

This class is inherited by the ticket & asset integrations

class tdxlib.tdx_integration.**TDXIntegration**(*filename: str = None*, *config: dict = None*)

auth() → bool
Internal method to authenticate to the TDX api using the selected method Stores a token in the token property, used for future calls. Returns true for success, false for failure.

clean_cache()
Internal method to refresh the cache in a tdxlib object.

create_account(*name: str*, *manager: str*, *additional_info: dict = None*, *custom_attributes: dict = None*) → dict
Creates an account in TeamDynamix.

Parameters

- **name** – Name of account to create.
- **manager** – email address of the TDX Person who will be the manager of the group
- **additional_info** – dict of other attributes to set on account. Retrieved from: https://api.teamdynamix.com/TDWebApi/Home/type/TeamDynamix.Api.Accounts.Account
- **custom_attributes** – dict of names of custom attributes and corresponding names of the choices to set on each attribute. These names must match the names in TDX, not IDs.

Returns a dict with information about the created account

Return type dict

create_room(*location*, *name: str*, *external_id: str = None*, *description: str = None*, *floor: str = None*, *capacity: int = None*, *attributes: dict = None*) → dict
Creates a room in a location in TDX.

Parameters

- **location** – Dict of location information (or ID), possibly from get_location_by_name()
- **name** – Name of new room
- **external_id** – External ID of new room as a string (optional)
- **description** – Description of new room as a string (optional)
- **floor** – Floor for new room as a string (optional)
- **capacity** – Capacity of the room, as an integer (optional)
- **attributes** – Dict of Custom Attributes (optional)

Returns Dict with newly created room information

edit_account(*name: str*, *changed_attributes: dict*) → dict
Edits an account in TeamDynamix

Parameters

- **name** – Name of account to edit.
- **changed_attributes** – dict of names of attributes and corresponding data to set on each attribute.

Returns a dict with information about the edited account

Return type dict

get_account_by_id(*account_id: int*) → dict
Gets an account by the account ID.

Parameters **account_id** – ID number of account to get information about

Returns dict of account data

Return type dict

get_account_by_name(*key: str*, *additional_params: dict = None*) → dict
Gets an account with by searching on its name.

Parameters

- **key** – a partial or full name of an account to search for
- **additional_params** – other search items, as a python dict, as described in TDX Api Docs

Returns dict of account data (not complete, but including the ID)

Return type dict

get_all_accounts() → list
Gets a list of all accounts in TDX

Returns list of dicts containing account data

Return type list

get_all_custom_attributes(*object_type: int*, *associated_type: int = 0*, *app_id: int = 0*) → list
Gets all custom attributes for the component type in TDX. See https://solutions.teamdynamix.com/TDClient/KB/ArticleDet?ID=22203 for possible values.

Parameters

- **object_type** – the object type to get attributes for (tickets = 9, assets = 27, CI's = 63)
- **associated_type** – the associated type of object to get attributes for, default: 0
- **app_id** – the application number to get attributes from, default: 0

Returns list of dicts containing custom attributes, including choices and choice ID's

Return type list

get_all_groups() → list
Gets a list of all groups in TDX

Returns list of dicts containing group data

Return type list

get_all_locations() → list
Gets all locations in TDX.

Returns a list of dicts containing location information

Return type list

get_custom_attribute_by_name_id(*key: str*, *object_type: int*) → dict
Gets a custom attribute for the component type. See https://solutions.teamdynamix.com/TDClient/KB/ArticleDet?ID=22203 for possible values for component_type.

NOTE: The best way to assign CA's is to test for an existing value (for choice-based CA's) using get_custom_attribute_value_by_name, and then if it returns false, directly assign the desired value to the CA. Because of this, date-type and other format-specific attributes need to be in a TDX-acceptable format, this means that a field designated to hold person objects needs to be set to a UID.

Parameters

- **key** – a partial or full name of the custom attribute to search for
- **object_type** – the object type ID to get attributes for

Returns the attribute as a dict, with all choice items included

Return type dict

get_custom_attribute_choice_by_name_id(*attribute*, *key*)
Gets the choice item from a custom attribute, maybe from get_custom_attribute_by_name()

NOTE: The best way to assign CA's is to test for an existing value (for choice-based CA's), and then if this method returns false, directly assign the desired value to the CA. Because of this, date-type and other format specific attributes need to be in a TDX-acceptable format, this means that a field designated to hold person objects needs to be set to a UID.

Parameters

- **key** – a partial or full name of the choice to look for
- **attribute** – a dict of custom attribute data (as retrieved from get_attribute_by_name())

Returns the the choice object from this attribute whose name matches 'key', or False if none matches.

Return type dict

get_group_by_id(*group_id: int*) → dict
Gets a group by the group ID.

Parameters **group_id** – ID number of group to get information about

Returns dict of group data, including members

Return type dict

get_group_by_name(*key: str*, *additional_params=None*) → dict
Gets a group by searching on its name.

Parameters

- **key** – a partial or full name of Group to search for
- **additional_params** – other search items, as a dict, as described in TDX Api Docs

Returns a dict of group data (not complete, but including the ID)

Return type dict

get_group_members_by_id(*group_id: int*) → list
Gets a list of group members by the group ID.

Parameters **group_id** – ID number of group to get members of

Returns list of person data for people in the group

Return type list

get_group_members_by_name(*key: str*) → list
Gets all the members of a group as person objects by searching on the group's name.

Parameters **key** – a partial or full name of a group

Returns list of group members

Return type list

get_location_by_id(*location_id: int*) → dict
Gets a location by the location ID.

Parameters **location_id** – ID number of location to get information about

Returns dict of location data

Return type dict

get_location_by_name(*key: str*, *additional_params: dict = None*) → dict
Gets a location by searching its name.

Parameters

- **key** – a partial or full name of the location to search for
- **additional_params** – other search items, as a dict, as described in TDX Api Docs

Returns a dict of location data

Return type dict

get_person_by_name_email(*key: str*) → dict
Gets the top match of people with based on a simple text search, such as: - Name - Email - Username - Organizational ID

Parameters **key** – string with search text of person to search with

Returns dict of person data

Return type dict

get_person_by_uid(*uid: str*) → dict
Gets a person by their UID.

Parameters **uid** – UID string corresponding to a person

Returns dict of person data

Return type dict

static get_room_by_name(*location: dict*, *room: str*) → dict
Gets a room by searching its name in location information, maybe from get_location_by_name().

Parameters

- **location** – dict of location info
- **room** – partial or full name of a room to search for

Returns a dict with all the information regarding the room. Use this to retrieve the ID attribute.

Return type dict

get_tdx_item_by_id(*obj_type: str*, *key*)
A generic function to get something from the TDX API using its ID/UID.

Since the TDX API endpoints are almost all in the form /<object type>/id, this method gives an easy way to template all the different get_<object>_by_id methods.

Parameters

- **obj_type** – the type of object to get.
- **key** – the ID number of an object to get, as a string

Returns list of person data

make_delete(*request_url: str*)
Makes an HTTP DELETE request to the TDX Api.

Parameters **request_url** – the path (everything after /TDWebAPI/api/) to call

Returns None

make_file_post(*request_url: str*, *file: BinaryIO*, *filename: str = None*)
Makes an HTTP POST request to the TDX Api with a Multipart-Encoded File

Parameters

- **request_url** – the path (everything after /TDWebApi/api/) to call
- **file** – BinaryIO object opened in read mode to upload as attachment.

(read documentation at requests.readthedocs.io/en/master/user/quickstart/#post-a-multipart-encoded-file) :param filename: (optional), allows to explicitly specify filename header. If None, requests will determine from passed-in file object. This is useful for if you want to upload a file in memory without a filename, which is required for uploading to TeamDynamix.

Returns the API's response as a python dict

make_get(*request_url: str*, *retries: int = 3*)
Makes an HTTP GET request to the TDX Api.

Parameters

- **request_url** – the path (everything after /TDWebAPI/api/) to call
- **retries** – the number of times to retry a failed request (defaults to 3)

Returns the API's response as a python dict or list

make_patch(*request_url: str*, *body: dict*)
Makes an HTTP PATCH request to the TDX API.

The TeamDynamix API supports limited PATCH functionality. Since TDX data is highly structured, items are referenced explicitly by their TDX ID, and not by their order in the object. Likewise, since the fields in a TDX object are all predefined, a PATCH call cannot add or remove any fields in the object.

Parameters

- **request_url** – the path (everything after /TDWebAPI/api/) to call
- **body** – a list of PATCH operations as dictionaries, each including the keys "op", "path", and "value"

Returns the API's response, as a python dict or list

make_post(*request_url: str*, *body: dict*)
Makes an HTTP POST request to the TDX Api

Parameters

- **request_url** – the path (everything after /TDWebAPI/api/) to call
- **body** – dumped JSON data to send with the POST

Returns the API's response as a python dict or list

make_put(*request_url: str*, *body: dict*)
Makes an HTTP PUT request to the TDX API.

Parameters

- **request_url** – the path (everything after /TDWebAPI/api/) to call
- **body** – dumped JSON data to send with the PUT

Returns the API's response as a python dict or list

search_people(*key: str*, *max_results: int = 20*) → list
Gets a list of people, based on a simple text search, which may match Name, Email, Username or ID

Parameters

- **key** – string with search text of person to search with

- **max_results** – maximum number of matches to return (Default: 20)

Returns list of dicts of person data

Return type list

1.2 TDX Ticket Integration

This class contains all the methods that work with the TDX Ticket API endpoints.

This class inherits the base TDX integration class.

1.2.1 Class & Methods

class tdxlib.tdx_ticket_integration.**TDXTicketIntegration**(*filename: str = None*, *config=None*)

add_asset_to_ticket(*ticket_id: int*, *asset_id: int*) → dict
Attaches an asset to a ticket.

Parameters

- **ticket_id** – The Ticket ID to update
- **asset_id** – The ID of the Asset to associate with the Ticket

Returns dict of update info

Return type dict

build_ticket_custom_attribute_value(*custom_attribute: Union[str, dict]*, *value: Union[str, int]*) → dict
Builds a custom attribute for a ticket from the name of the attribute and value.

Parameters

- **custom_attribute** – name of custom attribute (or dict of info from)
- **value** – name of value to set, or value to set to

Returns list of updated assets in dict format (for use in change_custom_attribute_value())

change_ticket_custom_attribute_value(*ticket: Union[dict, str, int, list]*, *custom_attributes: list*) → Union[tdxlib.tdx_ticket.TDXTicket, list]
Takes a correctly formatted list of CA's (from build_ticket_custom_attribute_value, for instance) and updates one or more assets with the new values.

Parameters

- **ticket** – ticket/Ticket ID to update (doesn't have to be full record), or list of same
- **custom_attributes** – List of ID/Value dicts (from build_ticket_custom_attribute_value())

Returns list of updated ticket in dict format

clean_cache()
Clears the tdx_ticket_integration cache.

Returns None

create_custom_ticket_status(*name: str*, *order: float*, *status_class: str*, *description: str = None*, *active: bool = True*) → dict
Creates a custom ticket status.

Parameters

- **name** – A string containing the name of the new status. (Required)
- **order** – A float containing the order number for sorting purposes.
- **status_class** – A name of a status class. These values are hard-coded into the TDWebApi, and stored in this class as a class variable.
- **description** – A string containing the description of the new status. (Default: Empty String)
- **active** – A bool indicating whether this new status should be active. (Default: True)

Returns The new ticket status as a dict

Return type dict

create_ticket(*ticket: tdxlib.tdx_ticket.TDXTicket*, *silent: bool = True*) → tdxlib.tdx_ticket.TDXTicket
Creates a ticket in TeamDynamix using a TdxTicket object

Parameters

- **ticket** – TDXTicket Object
- **silent** – Boolean – if False, notifications are sent to requestor and responsible, default: True

Returns Created ticket, if successful

Return type tdxlib.tdx_ticket.TDXTicket

create_ticket_task(*ticket_id: Union[str, int], task: dict*) → dict
Adds a ticket task to a ticket.

Parameters

- **ticket_id** – The ticket ID on which to create the ticket task.
- **task** – dict of task to create, possibly generated from generate_ticket_task

Returns dict of created ticket task information

Return type dict

delete_ticket_task(*ticket_id: str*, *task_id: str*) → None
Deletes a ticket task by ID

Parameters

- **ticket_id** – The ticket ID on which the ticket task exists.
- **task_id** – The task ID of the task you want to delete.

Returns none

edit_custom_ticket_status(*name: str*, *changed_attributes: dict*) → dict
Edits a custom ticket status

Parameters

- **name** – The name of the status (for finding the object to edit)

- **changed_attributes** – A dict containing values to substitute for the status's current values.

Returns The edited status information

Return type dict

edit_ticket(*ticket: Union[tdxlib.tdx_ticket.TDXTicket, str, int], changed_attributes: dict, notify: bool = False*) → tdxlib.tdx_ticket.TDXTicket

Edits one ticket, based on a dict of parameters to change.

Parameters

- **ticket** – a TDXTicket object or a Ticket ID
- **changed_attributes** – Attributes to alter in the ticket
- **notify** – If true, will notify newly-responsible resource if changed because of edit (default: false)

Returns edited ticket as TDXTicket

Return type tdxlib.tdx_ticket.TDXTicket

edit_ticket_task(*ticket_id: int, task: Union[str, int, dict], changed_attributes: dict*) → dict

Edits a ticket task with a set of new values.

Parameters

- **ticket_id** – The ticket ID on which the ticket task exists.
- **task** – a single ticket task in dict (maybe from get_ticket_task_by_id), or a task ID
- **changed_attributes** – The new values ot set on the ticket task.

Returns The modified ticket task as a dict, if the operation was successful

Return type dict

edit_tickets(*ticket_list: list*, *changed_attributes: dict*, *notify: bool = False*, *visual: bool = False*) → list

Edits one or more tickets, based on a dict of parameters to change

Parameters

- **ticket_list** – list of TDXTicket objects, maybe from search_tickets
- **changed_attributes** – Attributes to alter in selected tickets
- **notify** – If true, will notify newly-responsible resource(s) if changed because of edit
- **visual** – If true, print a . for each successful ticket that is edited

Returns list of edited TDXTicket objects, with complete data in json format

Return type list

generate_ticket(*title_template: str*, *ticket_type: str*, *account: str*, *responsible: str*, *template_values: dict = None*, *body_template: str = None*, *attrib_prefix: str = None*, *due_date: Union[datetime.datetime, str] = None*, *location: str = None*, *room: str = None*, *active_days: int = 5*, *priority: str = 'Low'*, *status: str = 'New'*, *requestor: str = None*, *classification: str = 'Incident'*, *form: str = None*, *responsible_is_group: str = False*, *custom_attributes: dict = None*) → tdxlib.tdx_ticket.TDXTicket

Makes a TdxTicket object based on templates.

Parameters

- **title_template** – a string with {placeholders} that correspond to keys in template_values dict (REQUIRED)
- **ticket_type** – name of ticket Type (REQUIRED)
- **account** – name of requesting Account for Ticket (REQUIRED)
- **responsible** – group or email address to set as responsible for ticket (REQUIRED)
- **template_values** – a dictionary with substitutions for title/body, using the {placeholders} as keys
- **body_template** – a string with {placeholders} that correspond to keys in template_values parameter
- **attrib_prefix** – [DEPRECATED] the string that prefixes all the custom attribute column names in the template_values dict
- **due_date** – due date for ticket, default None
- **location** – Building name of location (optional)
- **room** – Room name of location (optional) Will not set room if location not included.
- **active_days** – number of days before due date to assign start date, default is 5
- **priority** – name of priority of ticket, default "Low"
- **status** – name of status for new ticket, default "New"
- **requestor** – name or email for requester of the ticket, defaults to username of integration (optional)
- **classification** – name of classification name for new ticket, default "Incident" (optional)
- **form** – name or ID of a form to use for the new ticket
- **responsible_is_group** – Boolean indicating whether 'responsible' refers to a group. (Default: False)
- **custom_attributes** – dict of attribute names and values

Returns TdxTicket object ready to be created via create_ticket()

Return type tdxlib.tdx_ticket.TDXTicket

generate_ticket_task(*title: str*, *est_minutes: int = 30*, *description: str = None*, *start: datetime.datetime = None*, *end: datetime.datetime = None*, *completion_minutes: int = None*, *responsible: str = None*, *group: bool = False*, *predecessor: int = None*) → dict

Generates a dict with the information in the proper format for creating at ticket task.

Parameters

- **title** – A string indicating the title of the ticket (Required)
- **est_minutes** – Estimation of minutes required to complete this task (Default: 30) This is used for comparison to actual hours when using time tracking
- **description** – A string containing a description of the task (Default: Empty String)
- **start** – Datetime object indicating the start date of the ticket task (Default: date of creation)
- **end** – Datetime object indicating the end date of the ticket task. Sets the due date/time. (Default: one hour after start)

- **completion_minutes** – This parameter is used for tasks with predecessors. (Default: 0) They set the due date/time based on the activation date and time.
- **responsible** – String containing the name or partial name of a group or individual to assign the task to. (Default: None)
- **group** – Boolean indicating whether the responsible parameter references a group (Default:false)
- **predecessor** – Task ID of another task in the destination ticket to set as the predecessor. (Default: None)

Returns Dict of task information fit for creating a task on a ticket using create_task()

get_all_tasks_by_ticket_id(*ticket_id: Union[str, int], is_eligible_predecessor: bool = None*) → list

Gets a list of tasks currently on an open ticket. If the ticket is closed, no tasks will be returned.

Parameters

- **ticket_id** – The ticket ID to retrieve ticket tasks for.
- **is_eligible_predecessor** – (optional) If true, will only retrieve tasks that can be assigned as a predecessor for other tasks.

Returns list of ticket tasks as dicts

Return type list

get_all_ticket_assets(*ticket_id: int*) → list

Gets all asset attached to a ticket.

Parameters **ticket_id** – The Ticket ID to update

Returns list of asset info

Return type list

get_all_ticket_forms() → list

Gets a list of all ticket forms from TDX.

Returns list of ticket forms as python dicts

Return type list

get_all_ticket_impacts() → list

Gets a list of all ticket impacts from TDX.

Returns list of impacts in as python dicts

Return type dict

get_all_ticket_priorities() → list

Gets a list of all ticket priorities from TDX.

Returns list of priorities in python dict

Return type list

get_all_ticket_sources() → list

Gets a list of all ticket sources from TDX

Returns list of sources in python dict

Return type list

get_all_ticket_statuses() → list

Gets a list of all ticket statuses from TDX

Returns list of status data in python dicts

Return type list

get_all_ticket_types() → list
Gets a list of all ticket types from TDX.

Returns list of type data in python dicts

Return type list

get_all_ticket_urgencies() → list
Gets all ticket urgencies from the Tickets app.

Returns list of priorities in python dict

Return type dict

get_ticket_by_id(*ticket_id: int*) → tdxlib.tdx_ticket.TDXTicket
Gets a ticket, based on its ID

Parameters **ticket_id** – ticket ID of required ticket

Returns ticket info as python dict

Return type dict

classmethod get_ticket_classification_id_by_name(*name: str*)
Gets ticket classification data by searching by the name of the classification.

Parameters **name** – the name of the classification to search for

Returns dict of ticket classification info

Return type dict

get_ticket_custom_attribute_by_name(*key: str*) → dict
Gets a ticket custom attribute based on its name or ID. This includes hard-coded the component ID for tickets.

Parameters **key** – A full or partial name of the CA to get.

Returns a dict of custom attribute information

Return type dict

get_ticket_custom_attribute_by_name_id(*key: str*) → dict
Gets a ticket custom attribute based on its name or ID. This includes hard-coded the component ID for tickets.

Parameters **key** – A full or partial name of the CA to get.

Returns a dict of custom attribute information

Return type dict

get_ticket_feed(*ticket_id: Union[str, int]*) → list
Gets the feed entries from a ticket.

Parameters **ticket_id** – The ticket ID on which the ticket task exists.

Returns list of feed entries from the task as python dicts, if any exist

Return type list

get_ticket_form_by_name_id(*key: Union[str, int]*) → dict
Gets ticket form based on ID or Name.

Parameters **key** – Name or ID of form to search for

Returns form data in python dict

Return type dict

get_ticket_impact_by_name_id(*key: Union[str, int]*) → dict
Gets ticket impact based on ID or Name

Parameters **key** – Name or ID of impact to search for

Returns impact data in python dict

Return type dict

get_ticket_priority_by_name_id(*key: Union[str, int]*) → dict
Gets ticket priority based on ID or Name.

Parameters **key** – ID or Name of priority to search for

Returns priority data in python dict

Return type dict

get_ticket_source_by_name_id(*key: Union[str, int]*) → dict
Gets ticket source based on ID or Name

Parameters **key** – Name or ID of source to search for.

Supports search for exact name ('1. Phone'), or part of name ('Phone').

Returns source data in python dict

Return type dict

get_ticket_status_by_id(*key: Union[str, int]*) → dict
Gets ticket status based on ID or Name.

Parameters **key** – ID of ticket status to search for

Returns status data in python dict

Return type dict

get_ticket_task_by_id(*ticket_id: Union[str, int], task_id: Union[str, int]*) → dict
Gets ticket task by ID.

Parameters

- **ticket_id** – The ticket ID on which the ticket task exists.
- **task_id** – The ticket task ID.

Returns Task data in dict

Return type dict

get_ticket_task_feed(*ticket_id: Union[str, int], task_id: Union[str, int]*) → list
Gets all the feed entries from a ticket task.

Parameters

- **ticket_id** – The ticket ID on which the ticket task exists.
- **task_id** – The ticket task ID.

Returns list of feed entries from the task, if any exist

Return type list

get_ticket_type_by_name_id(*key: Union[str, int]*) → dict
Gets ticket type based on ID or Name.

Parameters **key** – Name or ID of attribute to search for

Returns type data in python dict

Return type dict

get_ticket_urgency_by_name_id(*key: Union[str, int]*) → dict
Gets ticket urgency based on ID or Name.

Parameters **key** – ID or Name of urgency to search for

Returns urgency data in python dict

Return type dict

make_call(*url: str*, *action: str*, *post_body: dict = None*)
Makes an HTTP call using the Tickets API information.

Parameters

- **url** – The URL (everything after tickets/) to call
- **action** – The HTTP action (get, put, post, delete, patch) to perform.
- **post_body** – A dict of the information to post, put, or patch. Not used for get/delete.

Returns the API response as a python dict or list

reassign_ticket(*ticket_id: Union[str, int], responsible: str, group: bool = False*) → tdxlib.tdx_ticket.TDXTicket
Reassigns a ticket to a person or group

Parameters

- **ticket_id** – The ticket of the ticket you want to edit.
- **responsible** – a username, email, Full Name, or ID number to use to search for a person
- **group** – If this parameter is True, assign to group instead of individual

Returns Edited TDXTicket object, if the operation was successful

Return type tdxlib.tdx_ticket.TDXTicket

reassign_ticket_task(*ticket_id: int, task: Union[str, dict, int], responsible: str, group=False*) → dict
Reassigns a ticket task to a person or group

Parameters

- **ticket_id** – The ticket ID on which the ticket task exists.
- **task** – a single ticket task in dict (maybe from get_ticket_task_by_id), or a task ID
- **responsible** – a username, email, Full Name, or ID number to use to search for a person, or a group name.
- **group** – If this parameter is True, assign to group instead of individual

Returns The modified ticket task as a dict, if the operation was successful

Return type dict

reschedule_ticket(*ticket_id: Union[str, int], start_date: datetime.datetime = False, end_date: datetime.datetime = False*) → tdxlib.tdx_ticket.TDXTicket

Reschedules the start and end dates of a ticket. This is impossible if the ticket has a task.

Parameters

- **ticket_id** – The ticket of the ticket you want to edit.
- **start_date** – datetime.datetime object for the start date of the ticket (defaults to now)
- **end_date** – datetime.datetime object for the end date of the ticket (defaults to now + 1 day)

Returns Edited TDXTicket object, if the operation was successful

Return type tdxlib.tdx_ticket.TDXTicket

reschedule_ticket_task(*ticket_id*, *task*, *start_date: datetime.datetime = None*, *end_date: datetime.datetime = None*) → dict

Sets the start date and end date for a ticket task. This will affect the start & end dates of the parent ticket.

Parameters

- **ticket_id** – The ticket ID on which the ticket task exists.
- **task** – a single ticket task in dict (maybe from get_ticket_task_by_id), or a task ID
- **start_date** – datetime.datetime object to use as the starting date for a task, defaults to now.
- **end_date** – datetime.datetime object to use as the starting date for a task, defaults to now + 1 day.

Returns The modified ticket task as a dict, if the operation was successful

Return type dict

search_ticket_status(*key: str*) → dict

Gets ticket status based on name.

Parameters **key** – Name of ticket status to search for

Returns status data in python dict

Return type dict

search_tickets(*criteria: dict*, *max_results: int = 25*, *closed: bool = False*, *cancelled: bool = False*, *other_status: bool = False*) → list

Gets a ticket, based on a variety of criteria:

```
{'TicketClassification': [List of Int],
'SearchText': [String],
'Status IDs': [List of Int],
'ResponsibilityUids': [List of String (GUID)],
'ResponsibilityGroupIDs': [List of String (ID)],
'RequestorEmailSearch': [String],
'LocationIDs': [List of Int],
'LocationRoomIds': [List of Int],
'CreatedDateFrom': [DateTime],
'CreatedDateTo': [DateTime],
'SlaViolationStatus': [Boolean -- true = SLA Violated]}
```

(https://api.teamdynamix.com/TDWebApi/Home/type/TeamDynamix.Api.Tickets.TicketSearch)

Parameters

- **max_results** – maximum number of results to return
- **criteria** – a string, list or dict to search for tickets with
- **cancelled** – include cancelled tickets in search if true
- **closed** – include closed tickets in search if true
- **other_status** – Status ID of a custom status

Returns list of TDXTicket objects

Return type list

update_ticket(*ticket_id: Union[str, int], comments: str, new_status: str = None, notify: list = None, private: bool = True*) → dict

Sends an update to a ticket feed.

Parameters

- **ticket_id** – the ticket ID whose task to update
- **comments** – a string to provide as a comment to the update.
- **new_status** – The name of the new status to set for the ticket (Default: The status whose ID is 0)
- **notify** – a list of strings containing email addresses to notify regarding this ticket. Default: None
- **private** – boolean indicating whether the update to the task should be private. Default: True

Returns python dict containing created ticket update information

Return type dict

update_ticket_task(*ticket_id: int, task_id: int, percent: int, comments: str = '', notify: list = None, private: bool = True*) → dict

Sends an update to a ticket task.

Parameters

- **ticket_id** – the ticket ID whose task to update
- **task_id** – the ID of the task to update
- **percent** – the percent complete to set the task to after update
- **comments** – a string to provide as a comment to the update. Defaults to empty string.
- **notify** – a list of strings containing email addresses to notify regarding this ticket. Default: None
- **private** – boolean indicating whether the update to the task should be private. Default: True

Returns dict of update info

Return type dict

upload_attachment(*ticket_id: Union[str, int], file: BinaryIO, filename: str = None*)

Uploads an attachment to a ticket.

Parameters

- **ticket_id** – the ticket ID to upload the attachment
- **file** – Python file object opened in binary read mode to upload as attachment

- **filename** – (optional), explicitly specify filename header. If None, requests will determine from

passed-in file object.

Returns python dict containing created attachment information

Return type dict

1.3 TDX Asset Integration

This class contains all the methods that work with the TDX Asset API endpoints.

This class inherits the base TDX integration class.

1.3.1 Searching for Assets

Special Characters

Using the method `search_assets()` exposes a few undocumented features in TDX's Web API. The primary one is some limited support for Regular Expressions in the SearchText parameter. This can be confusing because you may occasionally specify a Regex special character without realizing it.

For instance, you may use the following:

```
my_asset_integration_object.search_assets('K-')
```

This would be a pretty broad search anyway, but because of the way the TDX API interprets it, it actually includes all assets that have a "K" in their name or Serial number, or a number of other fields.

If to search for assets whose names or serial numbers begin with "K-", use the following:

```
my_asset_integration_object.search_assets('^K-')
```

This search returns more what you would expect.

Normal searches with only alphanumerical characters work as expected.

Searching with Attributes

The following may be useful if you need to filter by a specific product type:

```
prodmod = my_asset_integration_object.get_all_product_models()
```

Then, set up a list for the models you actually want:

```
models = []
```

Then run a quick loop, appending to the list:

```
for mod in prodmod:
    if mod['ProductTypeName'] == "Desktop" or mod['ProductTypeName'] == "Laptop":
        models.append(mod['ID'])
```

Then use this info in a filter in `search_assets()`

```
criteria = {'ProductModelIDs': models}
assets = my_asset_integration_object.search_assets(criteria, max_results=1000)
```

Searching with Custom Attributes

If you'd like to search for assets that have a specific value for a custom attribute, you'll need to do something similar:

```
ca = t.build_asset_custom_attribute_value("My CA Name", "My Value Name")
criteria = {'CustomAttributes': [ca]}
assets = my_asset_integration_object.search_assets(criteria, max_results=1000)
```

This will find all assets with the value "My Value Name" for the Custom Attribute "My CA Name".

For more information check out the documentation on the AssetSearch object in TDX's API docs

1.3.2 Custom Attributes

Custom Attributes are so useful in TeamDynamix and so difficult to understand in the API that it's worth a section in the documentation dealing directly with them.

The simplest use case is to set a custom attribute to a certain value on an asset.

1. Set up your TDX Integration:

```
import tdxlib
t = tdxlib.tdxassetintegration.TdxAssetIntegration()
```

2. Set up your dict of updated values with CA and value that you would like to set. This command retrieves the CA ID and the ID of the choice you selected (if a choice variable) and formats it correctly for ingestion by TDX. If it is a non-choice field, it will simply include the second attribute as the value of the CA.

```
update = {'Attributes': [t.build_asset_custom_attribute_value("My CA Name", "My Value␣
↪Name")]}
```

3. To add more attributes:

```
update['Attributes'].append(t.build_asset_custom_attribute_value("My Other CA Name",
↪"My Other Value Name"))
```

4. Get the assets you want to update:

```
assets = t.search_assets("KIOSK")
```

5. Update the assets:

```
updated_assets = t.update_assets(assets, update)
```

1.3.3 Class & Methods

class tdxlib.tdx_asset_integration.**TDXAssetIntegration**(*filename: str = None*)

add_asset_user(*asset_id: str*, *user_uid: str*) → dict
Adds a users to an asset

Parameters

- **asset_id** – the ID of the asset to get users
- **user_uid** – the UID of the person to add

Returns the API response (success/failure only)

build_asset(*asset_name: str*, *serial_number: str*, *status_name: str*, *location_name: str = None*, *room_name: str = None*, *asset_tag: str = None*, *acquisition_date: datetime.datetime = None*, *asset_lifespan_years: int = None*, *requester: str = None*, *requesting_dept: str = None*, *owner: str = None*, *owning_dept: str = None*, *parent: Union[int, str] = None*, *external_id: str = None*, *product_model: str = None*, *form: str = None*, *asset_custom_attributes: Union[list, dict] = None*, *supplier: str = None*, *replacement_date: datetime.datetime = None*) → dict

Makes a correctly-formatted dict of asset attributes for inputting into create_asset() function

Parameters

- **asset_name** – a string containing the name for the asset
- **serial_number** – String with serial number of new asset
- **status_name** – String with name of status of new asset
- **location_name** – String with name of location for new asset (optional)
- **room_name** – String with name of room for new asset (optional, requires location_name)
- **asset_tag** – String with asset tag value for new asset (optional)
- **acquisition_date** – Datetime for Acquisition date (Default: date of execution)
- **asset_lifespan_years** – Number of years you expect this device to be in service (Default: None)
- **requester** – String with email of requester for new asset (Default: integration username)
- **requesting_dept** – Account Name of requesting department for new asset
- **owner** – String with Email of owner of new asset
- **owning_dept** – String with Account name of owning department
- **parent** – Int with ID or String with serial number of a parent asset. Parent Asset must already exist.
- **external_id** – String with external id for new asset (Default: serial Number)
- **product_model** – String with name of product model
- **form** – Name of the Asset form to use
- **asset_custom_attributes** – a dictionary of asset custom attribute values (or list from asset['Attributes'])
- **supplier** – name of a TDX vendor to set as supplier (Default: manufacturer from product model)
- **replacement_date** – datetime object for expected replacement of asset (Default: set by asset_lifespan)

Returns dict usable in create_asset()

Return type dict

build_asset_custom_attribute_value(*custom_attribute: Union[dict, str, int], value: Union[datetime.datetime, str, int]*) → dict
Builds a custom attribute for an asset from the name of the attribute and value.

Parameters

- **custom_attribute** – name of custom attribute (or dict of info)
- **value** – name of value to set, or value to set to

Returns list of updated assets in dict format (for use in change_custom_attribute_value())

change_asset_custom_attribute_value(*asset: Union[dict, str, int, list], custom_attributes: list*) → list
Takes a correctly formatted list of CA's (from build_asset_custom_attribute_value, for instance) and updates one or more assets with the new values.

Parameters

- **asset** – asset to update (doesn't have to be full record), or list of same
- **custom_attributes** – List of ID/Value dicts (from build_asset_custom_attribute_value())

Returns list of updated assets in dict format

change_asset_location(*asset: Union[dict, str, int, list], new_location, new_room=None*)
Updates Location data in a list of assets

Parameters

- **asset** – asset to update (doesn't have to be full record), or list of same
- **new_location** – name of new location, or dict of location data
- **new_room** – name of new room, or dict of room data

Returns list of the updated assets

change_asset_owner(*asset: Union[dict, str, int, list], new_owner, new_dept=None*) → list
Updates owner data in a list of assets

Parameters

- **asset** – asset to update (doesn't have to be full record), or list of same
- **new_owner** – email or name of new owner, or dict of their information
- **new_dept** – name of new department, or dict of information

Returns list of the updated assets

change_asset_requesting_dept(*asset: Union[dict, str, int, list], new_dept*) → list
Updates Requesting Department data in a list of assets

Parameters

- **asset** – asset to update (doesn't have to be full record), or list of same
- **new_dept** – name of new department

Returns list of the updated assets

clean_cache() → None
Internal method to refresh the cache in a tdxlib object.

clear_asset_custom_attributes(*asset: Union[dict, str, int], attributes_to_clear: Union[str, list]*) → dict
Takes a list of CA names and removes those custom attributes from the provided asset

Parameters

- **asset** – asset to update (doesn't have to be full record)
- **attributes_to_clear** – List of names of custom attributes to remove

Returns the updated asset in dict format

copy_asset_attributes(*source_asset: dict*, *target_asset: dict*, *copy_name: bool = False*, *exclude: list = None*, *new_status: str = None*, *new_name: str = None*, *is_full_source: bool = False*)

Copies asset attributes from one asset to another. Does not include attributes like Serial Number, Asset Tag, and other hardware-specific fields.

Parameters

- **source_asset** – asset to copy attributes from (doesn't have to be full record)
- **target_asset** – asset to copy attributes to (doesn't have to full record) This asset will be OVERWRITTEN!
- **copy_name** – Set to true to copy the name of the source asset to the target asset
- **exclude** – List of attributes to be excluded, in addition to defaults
- **new_status** – Name or ID of new status for source asset
- **new_name** – New name for source asset (usually used with copy_name=True). Default: False
- **is_full_source** – Boolean indicating whether source_asset is a full asset record or not. Default: False

Returns list of the target and source asset data

create_asset(*asset: dict*, *check_duplicate: bool = True*) → dict

Creates an asset

Parameters

- **asset** – a dict of asset info (maybe from make_asset_json()) to use in creation
- **check_duplicate** – boolean of whether we should check to see if this is a duplicate asset

Returns dict of created asset details

create_product_model(*name: str, product_type: Union[str, dict], source: str, description: str = None, part_number: str = None, active: bool = True, attributes: dict = None*) → dict

Creates a new Product Model with the information provided.

Parameters

- **name** – The name of the new product model
- **product_type** – A Type (dict) or Type ID to set as the product type of this model
- **source** – The manufacturer or vendor the model is sourced from
- **description** – A description of the model (optional)
- **part_number** – Part number for this model (optional)
- **active** – Boolean indicating whether the new model should be active (optional, default True)

- **attributes** – Dict of custom attributes to set on the new model (no validation yet – build this by hand)

Returns dict of created product type

create_product_type(*name: str*, *description: str = None*, *parent=None*, *order: int = 1*, *active: bool = True*) → dict

Creates a new Product Type with the information provided.

Parameters

- **name** – The name of the new product type
- **description** – A description of the type (optional)
- **parent** – A Type (dict) or Type ID to set as the parent type of this type (creates a subtype)(optional)
- **order** – Sort order for this type (optional, defaults to 1)
- **active** – Boolean indicating whether the new type should be active (optional, default True)

Returns dict of created product type

create_vendor(*name: str*, *email: str = None*, *description: str = None*, *account_number: str = None*, *additional_info: dict = None*, *active=True*) → dict

Creates a new Vendor with the information provided.

Parameters

- **name** – The name of the new product model
- **email** – An email contact for the new vendor (optional)
- **description** – A description of the model (optional)
- **account_number** – An account number with the vendor (optional)
- **active** – Boolean indicating whether the new vendor should be active (optional, default True)
- **additional_info** – Dict of other info for the vendor (including CAs, no validation yet – build by hand)

Returns dict of created product type

delete_asset_users(*asset_id: str*, *users: list*)

Deletes specified users of an asset

Parameters

- **asset_id** – the ID of the asset to delete users from
- **users** – a list of the users (maybe from get_asset_users()) or user UIDs to delete

Returns list of this asset's users, each represented by a dict

find_asset_by_sn(*sn: str*, *full_record: bool = False*, *all_statuses: bool = True*) → dict

Gets an asset based on its serial number

Parameters

- **sn** – serial number as a string
- **full_record** – boolean indicating whether to fetch the full Asset record, or just summary info

- **all_statuses** – gets assets, regardless of what their status is (default: True)

Returns the single asset with the corresponding serial number

find_asset_by_tag(*tag: str*, *full_record: bool = False*, *all_statuses: bool = True*) → dict
Gets an asset based on its asset tag

Parameters

- **tag** – asset tag as a string
- **full_record** – boolean indicating whether to fetch the full Asset record, or just summary info
- **all_statuses** – gets assets, regardless of what their status is (default: True)

Returns the single asset with the corresponding tag

get_all_asset_forms() → list
Gets a list asset forms

Returns list of form data

get_all_asset_statuses() → list
Gets a list asset statuses

Returns list of status data

get_all_product_models() → list
Gets a list asset models

Returns list of model data

get_all_product_models_of_type(*product_type: Union[str, dict]*) → list
Get all product models of a specific type

Parameters **product_type** – dict, name, or ID of a product type

Returns list of product models of that type

get_all_product_types() → list
Gets a list of all product types

Returns list of product type data

get_all_vendors() → list
Gets a list vendors

Returns list of vendor data

get_asset_by_id(*asset_id: Union[str, int]*) → dict
Gets a specfic asset object, including the full list of attributes.

Parameters **asset_id** – asset ID from TDX

Returns dict of asset data

get_asset_custom_attribute_by_name_id(*key: str*) → dict
Gets a specific Asset Custom Attribute object

Parameters **key** – name or id of the Custom Attribute. This must be the exact name, no partial searching.

Returns dict of custom attribute data

get_asset_custom_attribute_value_by_name(*asset: Union[dict, str, int], key: str, id_only: bool = False*) → str
Returns the current value of a specific CA in the specified asset

Parameters

- **asset** – asset to get CA value for, in dict or ID form
- **key** – Name or ID of CA to find in the asset
- **id_only** – (Default: False) Return the ID of the value, instead of ValueText (only for choice-based CA's)

Returns a string representation of the value or ID

get_asset_form_by_name_id(*key: str*) → dict
Gets a specific asset form object

Parameters **key** – name of AssetForm to search for

Returns list of form data

get_asset_status_by_name_id(*key: str*) → dict
Gets a specific asset status object

Parameters **key** – name of an asset status to search for

Returns dict of status data

get_asset_users(*asset_id: str*) → list
Gets users of an asset

Parameters **asset_id** – the ID of the asset to get users

Returns list of this asset's users, each represented by a dict

get_assets_by_location(*location: Union[str, dict, list], max_results: int = 5000, full_record: bool = False, retired: bool = False, disposed: bool = False, all_statuses: bool = False*) → list
Gets all assets in a location

Parameters

- **location** – a single location (from get_location_by_name()) or list of same
- **max_results** – an integer indicating the maximum number of results that should be returned (default: 25)
- **full_record** – boolean indicating whether to fetch the full Asset record, or just summary info
- **retired** – include retired assets in search if true
- **disposed** – include disposed assets in search if true
- **all_statuses** – gets assets, regardless of what their status is (default: False)

Returns a list of assets in the location(s)

get_assets_by_owner(*person: str*, *max_results: int = 25*, *full_record: bool = False*, *retired: bool = False*, *disposed: bool = False*, *all_statuses: bool = False*) → list
Gets all assets owned by a particular person in TDX

Parameters

- **person** – the name or email of a person in TDX, or a dict containing their information
- **max_results** – an integer indicating the maximum number of results that should be returned (default: 25)
- **full_record** – boolean indicating whether to fetch the full Asset record, or just summary info

- **retired** – include retired assets in search if true
- **disposed** – include disposed assets in search if true
- **all_statuses** – gets assets, regardless of what their status is (default: False)

Returns a list of assets owned by that person

get_assets_by_product_model(*model: Union[dict, str, int], max_results: int = 25, full_record: bool = False, retired: bool = False, disposed: bool = False, all_statuses: bool = False*) → list

Gets all assets of a certain product model

Parameters

- **model** – the name or ID of a product model, or a dict of same
- **max_results** – an integer indicating the maximum number of results that should be returned (default: 25)
- **full_record** – boolean indicating whether to fetch the full Asset record, or just summary info
- **retired** – include retired assets in search if true
- **disposed** – include disposed assets in search if true
- **all_statuses** – gets assets, regardless of what their status is (default: False)

Returns a list of assets of the specified model

get_assets_by_product_type(*product_type: Union[dict, str, int], max_results: int = 25, full_record: bool = False, retired: bool = False, disposed: bool = False, all_statuses: bool = False*) → list

Gets all assets of a certain product type

Parameters

- **product_type** – the name or ID of a product type, or a dict of same
- **max_results** – an integer indicating the maximum number of results that should be returned (default: 25)
- **full_record** – boolean indicating whether to fetch the full Asset record, or just summary info
- **retired** – include retired assets in search if true
- **disposed** – include disposed assets in search if true
- **all_statuses** – gets assets, regardless of what their status is (default: False)

Returns a list of assets of the specified type

get_assets_by_requesting_department(*dept: Union[dict, str], max_results: int = 25, full_record: bool = False, retired: bool = False, disposed: bool = False, all_statuses: bool = False*) → list

Gets all assets requested by a particular account/department in TDX

Parameters

- **dept** – the name or email of an account/department, or a dict containing its information
- **max_results** – an integer indicating the maximum number of results that should be returned (default: 25)

- **full_record** – boolean indicating whether to fetch the full Asset record, or just summary info
- **retired** – include retired assets in search if true
- **disposed** – include disposed assets in search if true
- **all_statuses** – gets assets, regardless of what their status is (default: False)

Returns a list of assets requested by that department

get_assets_by_room(*room: dict*, *max_results: int = 25*, *full_record: bool = False*, *retired: bool = False*, *disposed: bool = False*, *all_statuses: bool = False*) → list
Gets all assets in a specific room in a location

Parameters

- **room** – a single room (from get_room_by_name())
- **max_results** – an integer indicating the maximum number of results that should be returned (default: 25)
- **full_record** – boolean indicating whether to fetch the full Asset record, or just summary info
- **retired** – include retired assets in search if true
- **disposed** – include disposed assets in search if true
- **all_statuses** – gets assets, regardless of what their status is (default: False)

Returns a list of assets in the room

get_product_model_by_name_id(*key: Union[str, int]*) → dict
Gets a specific product model object

Parameters **key** – name of product model to search for

Returns dict of model data

get_product_type_by_name_id(*key: str*) → dict
Gets a specific product type object

Parameters **key** – name of product type to search for

Returns dict of product type data

get_vendor_by_name_id(*key: str*) → dict
Gets a specific vendor object

Parameters **key** – name or ID of vendor to search for

Returns dict of vendor data

make_call(*url: str*, *action: str*, *post_body: dict = None*) → Union[list, dict]
Makes an HTTP call using the Assets API information.

Parameters

- **url** – The URL (everything after assets/) to call
- **action** – The HTTP action (get, put, post, delete, patch) to perform.
- **post_body** – A python dict of the information to post, put, or patch. Not used for get/delete.

Returns the API response as a python dict or list

move_child_assets(*source_asset: Union[dict, str, int], target_asset: Union[dict, str, int]*) → list
Moves child assets from one parent asset to another

Parameters

- **source_asset** – asset (or asset ID) to move children from (doesn't have to be full record)
- **target_asset** – asset (or asset ID) to move children to

Returns list of the updated assets

search_assets(*criteria: Union[str, dict], max_results=25, retired=False, disposed=False, full_record=False, all_statuses: bool = False*) → list
Searches for assets, based on criteria

Common criteria to put in dict: {'SerialLike': [List of Int], 'SearchText': [String], 'StatusIDs': [List of Int], 'CustomAttributes': [Dict of CA], 'ParentIDs': [List of Int]} (https://api.teamdynamix.com/TDWebApi/Home/type/TeamDynamix.Api.Assets.AssetSearch)

Parameters

- **max_results** – maximum number of results to return
- **criteria** – a string or dict to search for tickets with. If a string, use as 'SearchString'
- **retired** – include retired assets in search if true (overridden if "StatusIDs" in criteria or all_statuses is set)
- **disposed** – include disposed assets in search if true (overridden if "StatusIDs" in criteria or all_statuses is set)
- **full_record** – get full asset record (Default: False). Takes more time, but returns full asset record(s)
- **all_statuses** – gets assets, regardless of what their status is (default: False) (overridden if "StatusIDs" in criteria)

Returns list of asset info, or None if no assets found matching criteria. (by default, NOT FULL ASSET RECORDS, pass full_record=True to get full record)

search_product_types(*search_string: str = '*', active: bool = True, root_only: bool = False, parent=None*) → list
Searches product types by parent, text, or parent.

Parameters

- **search_string** – String to search by name/description
- **active** – Boolean, when true, searches only active product types
- **root_only** – Boolean, when true, limits search to only top-level product types
- **parent** – Name or ID of parent type. Limits search to children of that parent

Returns list of product types as dicts

update_assets(*assets: Union[dict, str, int, list], changed_attributes: dict, clear_custom_attributes: bool = False*) → list
Updates data in a list of assets

Parameters

- **assets** – a list of assets (maybe from search_assets()) or a single asset (only ID required)
- **changed_attributes** – a dict of attributes in the ticket to be changed

- **clear_custom_attributes** – (default: False) Indicates whether custom attributes not specified in the changed_attributes argument should be cleared

Returns list of updated assets

update_product_type(*product_type: Union[str, dict], updated_values: dict*) → dict
Updates an existing product type

Parameters

- **product_type** – Type (dict) or Type ID to edit
- **updated_values** – dict of values that should be changed

Returns dict of edited product type

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Index

E

F

G

M

R

S

T

U