AZENTA
LIFE SCIENCES

FreezerPro

API Guide

Title:	FreezerPro 7.4.3 API Guide
Doc No:	**DHF-FreezerPro-API Guide-01**
Rev. No:	**.4**

Revision Record

Revision Number	Date	Author	Description of Change
.1	25-Oct-2016	Louis Riley	Revision 1
.2	21-Mar-2018	Clay Angelly	Changed branding from RuRo to Brooks Life Sciences and URL to publicly accessible link to this API document.
.3	04-Apr-2019	Jennifer Bell	Add delete_box method
.4	29-Aug-2020	Jennifer Bell	Add set_sensor method, correct discrepancies found in testing

1. Introduction

This document describes the client-side API used to access data objects stored in the FreezerPro database. By using this API, users can create custom reports and extensions and integrate FreezerPro with other software.

Note: The user account used to access the API should have a sufficient privileges to access certain methods).

In Ruby, for easy parsing of JSON data return from FreezerPro, install JSON gem (gem install json) or XML gem (gem install libxml-ruby).

- FreezerPro API is compatible with both JSON (JavaScript Object Notation) and XML (Extensible Markup Language) standards for data exchange.
- JSON and XML are well-documented text formats that are completely programing-language independent and can be used from many scripting languages like JavaScript, Ruby, Python, Perl and many others.
- For the purposes of this document Ruby language will be used for all code samples and JSON as a data format.

- To access the API functions a valid user account within FreezerPro is needed. All access is provided via an http connection to the URL in the following format:
    - http://{FREEZERPRO_ADDR}:{FREEZERPRO_PORT}/api
      (Example: http://freezerpro.com/api)
- A web browser can be used to test API calls.

2. List of Objects Supported by FreezerPro API

Following is a list of Objects and their identifying values that are called by supported methods. See the 'List of API Functions' section for corresponding methods.

2.1. AuditRec:

Uniquely identifies an action in the system:

:**obj_id** = Unique ID of a record.

:id = Same as above.

:obj_name = Name of FreezerPro object that was affected (Sample, User, etc.).

:user_name = User who performed the action that created the audit.

:created_at = The date at which the record was created.

:created_time = The time at which the record was created.

:message = The action performed that created the record.

:comments = Any user-generated comment added when the audited action was performed.

2.2. Auth_token:

An authorization token used in place of a user's password for API calls. Using the auth_token will omit the "API Session Created" and "API Session Removed" entries in the audit log. A generated token will only be valid for ten minutes after it was created or since it was last used.

2.3. Box:

Uniquely identifies a storage box in the system. Each box belongs to a particular box type:

:id = Unique ID of a box.

:obj_id = Same as above.

:location = Full box path(location) in the storage hierarchy.

:name = Box name.

:description = Description of the box.

:width = Box width(number of horizontal cells).

:height = Box height(number of vertical cells).

:samples = Number of samples(vials) in the box.

:barcode_tag = Unique barcode ID of the box.

:rfid_tag = Unique RFID tag of the box.

:box_id_type = Box type.

:created_at = Date when the box was created in the system.

:updated_at = Date when the box was last updated.

2.4. BoxType:

Defines a what type a storage box is:

:obj_id = Unique ID of a box type.

:id = Same as above.

:name = Name of a box type.

:width = Box type horizontal dimensions(number of cells).

:height = Box type vertical dimensions(number of cells).

:coords = Box type coordinate system(Alpha/Numeric, Numeric/Alpha, etc.).

:inuse = The number of boxes of this type within the system.

2.5. Freezer:

Uniquely identifies a freezer storage:

:id = Unique ID of a freezer.

:name = Internal name of the freezer.

:description = Description of the freezer.

:subdivisions = Number of first-level subdivisions in the freezer.

:boxes = Number of first-level boxes in the freezer.

:barcode_tag = Unique barcode ID of the freezer.

:rfid_tag = Unique RFID tag of the freezer.

2.6. Location:

A location(single vial or aliquot is the lowest storage unit to contain a sample) of a sample. Each location identifies a vial location in a virtual box. Virtual samples can contain more than one vial therefore more than one location can be associated with a sample:

:id = Unique ID of the sample being located.

:custom_id = Custom vial identifier of a sample.

:barcode_tag = Unique barcode ID of the sample.

:rfid_tag = Unique RFID tag of the sample.

:location = Full vial path(location) in the storage hierarchy.

:position = The specific position of the vial within the box it is stored in.

:freeze_thaw = How many times the sample has been removed from a freezer

:out_by_user_id = The Unique ID of the user who last removed the sample from a freezer.

:out_at = The date the sample was last removed from a freezer.

:sample_id = ID of the sample associated with the vial.

:volume = The volume of the sample within the vial.

:box_id= The unique identifier of the box containing the vial.

:box_name = The name of the box containing the vial.

:Freezer = The name of the freezer containing the vial.

:Level1 = The name of the subdivision containing the vial.

2.7. Role:

A role that defines the access permissions in the system:

:id = Unique ID of a role.

:name = Internal name of the role.

:rights = The rights associated with this role.

:updated_at = Date when the role was last updated.

2.8. Sample:

A frozen laboratory sample. Sample can have multiple location(aliquots) with each aliquot represented by a vial in a virtual box:

:id = Unique ID of a sample.

:name = Sample name.

:description = Description of the sample.

:icon = Sample type icon URL.

:rfid_tag = The unique RFID number assigned to the subject.

:sample_type = Sample type name (Example: Bacteria or DNA).

:source = Name of the sample source.

:source_id = Unique ID of the sample source.

:group_ids = The unique IDs of all groups this sample is associated with.

:scount = The number of sample locations(vials or aliquots).

:volume = The total volume of the sample.

:volume_in_freezers = The total volume of the sample currently in freezers.

:owner = The current owner of the sample.

:created_at = The date on which the sample was created.

:updated_at = The date on which the sample was last updated.

:location = Full vial path(location) in the storage hierarchy.

2.9. SampleGroup:

A group of samples. Multiple samples can be organized into sample groups:

:id = Unique ID of a sample group.

:obj_id = Same as above.

:inuse = Total number of samples in the group.

:name = Sample group name.

:vials = The number of vials in the group.

:description = Description of the sample group.

:created_at = The date on which the sample group was created.

:updated_at = The date on which the sample group was last updated.

2.10. SampleSource:

Identifies a source or origin of a sample. Each sample source belongs to a particular sample source type:

:id = Unique ID of a sample source.

:obj_id = Same as above.

:sample_source_type = The name of the sample source type(Patient).

:scount = The number of sample locations(vials or aliquots) associated with this sample source.

:name = Sample source name.

:description = Description of the sample source.

:created_at = The date on which the sample source was created.

:updated_at = The date on which the sample source was last updated.

:enabled = Whether the sample source is enabled or disabled.

:userfields = The user defined fields in use for this sample source.

2.11. SampleSourceType:

Defines a type of a sample sources to be tracked in the system:

:id = Unique ID of a sample source type.

:name = Sample source type name.

:descr = Description of the sample source type.

:fields_count = The number of user defined fields associated with this sample source type.

:fields = The name of each user defined field associated with this sample source type.

:created_at = The date on which the sample source type was created.

:updated_at = The date on which the sample source type was last updated.

:inuse = Total number of sample sources of this type in the

system. **:enabled** = Whether the sample source type is

enabled or disabled.

2.12. SampleType:

Uniquely defines a type of a sample in the system:

:id = Unique ID of a sample type.

:icon = Sample type icon URL.

:name = Sample type name.

:enabled = Whether the sample type is enabled or disabled.

:units = Unit of volume used by the sample type.

:descr = Description of the sample type.

:fields_count = The number of user defined fields associated with this sample type.

:fields = The name of each user defined field associated with this sample type.

:created_at = The date on which the sample type was created.

:updated_at = The date on which the sample type was last updated.

:inuse = Total number of samples of this type in the system.

2.13. Subdivision:

Defines a level in a freezer hierarchy (such as racks, shelves, etc.):

:id = Unique ID of a subdivision.

:obj_id = Same as above.

:name = Subdivision name.

:description = Description of the subdivision.

:subdivisions = Number of the next-level subdivisions within this subdivision.

:boxes = Number of boxes within this subdivision.

:barcode_tag = Unique barcode ID of the subdivision.

:rfid_tag = Unique RFID tag of the subdivision.

2.14. User:

Uniquely identifies a user in the system:

:id = Unique ID of a User.

:obj_id = Same as above.

:username =

Login name.

:fullname = User’s full name.

:email = User’s email.

:created_at = The date on which the sample type was created.

:disabled = Defines whether the user is disabled or not.

:active = Defines whether the user is currently active or not.

:role = What role is currently assigned to the user.

:samples = How many samples the user is currently the owner of.

2.15. UserField:

A user defined field within the system:

:id = Unique ID of a sample type.

:obj_id = Same as above.

:display_name = Name of the user defined field.

:name = Same as above.

:type = User defined field type (Date, Text Field, List, Choice, etc.).

:values = The values used within a List or Choice UDF type.

:created_at = The date on which the sample type was created.

:updated_at = The date on which the sample type was last updated.

:inuse = Total number of samples of this type in the system.

:inuse_by = The names of the items currently using this userfield.

2.16. SensorReading

A temperature sensor reading

:id = Unique id of the reading

:value = The value of the sensor reading

:created_at = The date the sensor reading was created

:container_id = The id of the freezer the reading is associated with

3. List of API Functions

Following is a list of Methods along with the returned objects they call along with any required, query, or control parameters to use with them. See Section 3. List of API Examples for list of corresponding script examples. An explanation of each is as followed:

Returned Objects: FreezerPro objects returned by API function.

Required Parameters: Necessary parameter or parameters for the method to run correctly. Without the required parameters an error message from the server should be expected. These will be written in the format required for the script along with placeholder text within the ‘ ’ marks.

Optional Query parameters: Optional parameters to control the results. These will be written in the format required for the script along with placeholder text within the ‘ ’ marks.

Optional Control parameters: Optional parameters to control the number and order of records in the output, and to implement paging of the results. These will be written in the format required for the script along with placeholder text within the ‘ ’ marks.

3.1. “advanced_search”:

Retrieves a list of samples which match the search parameters.

Search Parameters:

:subject_type = The subject type to be searched. Currently can only be “sample”.

:query = Array of search conditions. Search condition is a Hash with the following keys:

:type = Search field type. “sdf” (system defined field/standard field) or “udf” (user defined field). Default: “sdf”

:field = Name of field to be searched. Following is a list of SDFs for “Sample” and the corresponding value type:

- **id** = Number.
- **name** = Text string.
- **description** = Text string.
- **sample_type_id** = Number.
- **sample_type_name** = Text string.
- **sample_source_id** = Number.
- **sample_source_name** = Text string.
- **sample_group_id** = Number.
- **sample_group_name** = Text string.
- **owner_id** = Number.
- **owner_username** = Text string.
- **owner_fullname** = Text string.
- **created_at** = Formatted date.

- **updated_at** = Formatted date.
- **expiration** = Formatted date.
- **locations_count** = Number.

:op = Use one of the following conditional operators to define what search is being performed:

- **"contains"**
- **"equals" ("eq" for short)**
- **"not equals" ("ne" for short)**
- **"greater than" ("gt" for short)**
- **"less than" ("lt" for short)**

:value = The value or array of values being searched for

Optional Control Parameters:

:sdfs = Array of SDF names to return in the result.

:udfs = Array of UDF names to return in the result.

:limit = Maximum number of subjects to include in the result. There is no limit by default.

:start = Index of the first result to return.

:sort = Sort by field, using one of the following:

- **id (default)**
- **sample_type_id**
- **owner_id**

:dir = Sort direction, using one of the following:

- **ASC (default)**
- **DESC**

3.2. "alerts":

Retrieves a list of all active sample alerts within the system.

Returned Objects:

Samples **Required**

Parameters: None

Optional Query Parameters:

:query=>'text' = Limits the list to items containing a specific text string.

Optional Control Parameters:

:start=>'value' = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

3.3. "audits":

Retrieves a list of audit records within the system.

Returned Objects:

AuditRec **Required**

Parameters: None

Optional Query Parameters:

:date_flag=>'all/today/yesterday/week/month' = Displays audit records made today, yesterday, this week, or this month.

:date_range=>'date_from,date_to' = Displays audit records made within a specific date range(format: mm/dd/yyyy).

Optional Control Parameters:

:start=>'value' = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

3.4. "box_types":

Retrieves a list of box types within the system.

Returned Objects: BoxTypes

Required Parameters: None

Optional Query

Parameters: None **Optional**

Control Parameters: None

3.5. “box_userfields”:

Retrieves a list of user-defined fields within the system.

Returned Objects: A list of key/value pairs

Required Parameters: None

:id=>’value’ = ID of the box in question.

Optional Query

Parameters: None **Optional**

Control Parameters: None

3.6. “boxes”:

Retrieves a list of boxes within the system.

Returned Objects: Boxes

Required Parameters: None

Optional Query Parameters:

:user_id=>'value' = Option to search boxes made by specific users

:show_empty=>'true/false' = Option to hide/show empty boxes from the result

:query=>’text’ = Limits the list to items containing a specific text string.

Optional Control Parameters:

:start=>’value’ = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

3.7. “delete_box”:

Deletes a box with the corresponding RFID tag, Barcode tag, or ID.

Returned Objects: None

Required Parameters: Use one of the following

:rfid_tag=>”value” = The RFID tag of the box in question.

:barcode_tag=>”value” = The barcode of the box in question.

:id=>”value” = The ID of the box in question.

Optional Query Parameters: None

Optional Control Parameters: None

:comment => “value” = A comment for the audit log :force

=> “Yes” = Delete this box even if it contains samples.

3.8. “delete_vials”:

Deletes a vial with the corresponding RFID tag, Barcode tag, custom ID or vial ID.

Returned Objects: None

Required Parameters: Use one of the following

:rfid_tags=>”value” = The RFID tag of the Vials in question. Multiple RFIDs may be used, separated by comma.

:barcode_tags=>”value” = The barcode of the Vials in question. Multiple barcodes may be used, separated by comma.

:custom_ids=>”value” = The custom ID of the Vials in question. Multiple RFIDs may be used, separated by comma.

:vial_ids=>”value” = The ID of the Vials in question. Multiple IDs may be used, separated by comma.

Optional Query Parameters: None

Optional Control Parameters: None

3.9. “freezer_samples”:

Retrieves a list of samples from a freezer or subdivision within the system.

Returned Objects: Samples

Required Parameters:

:id=>’value’ = Freezer or subdivision id to be

searched. **Optional Query Parameters:** None

Optional Control Parameters:

:start=>’value’ = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific identifier such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

3.10. “freezers”:

Retrieves a list of all freezers within the system.

Returned Objects: Freezers

Required Parameters: None

Optional Query

Parameters: None **Optional**

Control Parameters: None

3.11. “gen_token”:

Creates an “auth_token” to be used instead of a user’s password. Using the auth_token will omit the “API Session Created” and “API Session Removed” entries in the audit log.

Note: Generated Authorization Tokens are only valid for 10 minutes after they are either generated or last used. If an expired or invalid token is used, the following error message should be displayed: “auth_token is not valid”. **Returned Objects:** Auth_token

Required Parameters: None

Optional Query

Parameters: None **Optional**

Control Parameters: None

3.12. “get_perfect_box”:

Retrieves the ID and full location path for the “perfect” box with the desired empty cells so the import methods can use it.

Returned Objects: JSON object with Box ID and full location.

(**Example:** {"success"=>true, "box_id"=>259, "location"=>"Freezer->Level 6->Test Box 1"}).

Required Parameters:

:freezer_name=>'text' = The name of the freezer where to find the

"perfect box" **OR**

:container_id=>'value' = The unique ID of the freezer or subdivision where to find the "perfect box".

:space =>value = The desired empty space inside the "perfect box"(Example: 10x10).

Optional Query

Parameters: None **Optional**

Control Parameters: None

3.13. "get_job_status":

Retrieves the status of a background import/update job given the specific job ID. The following Import/Update methods after this method will return the required job IDs. NOTE: If this method is used on a quick job (less than 50-100 items) it will not output a result.

All import/update operations have the following "Status" constants:

ERROR = 1

DONE = 2

PROGRESS = 3

CLIENT_ALIVE = 4

CANCEL_JOB = 5

Returned Objects: JSON object job status

(**Example:** {"status":x,"total":NNN,"msg":MMM}

OR

{"status":x,"msg":"msg":"NNN Records successfully imported/updated."}).

Required Parameters:

:job_id=>'text' = The unique job ID of the import/update process being pulled.

(**Example:**

sample_group_updaters:update:e777973230cf0ef4f46d9cccfec7787f). **Optional**

Query Parameters: None

Optional Control Parameters:

:cancel=>'text' = Cancel the job using the job_id and rolls back the transaction.

3.14. “import_sources”:

Imports a sample source or set of sample sources via CSV.file. UDFs may be added by adding them as a column in the CSV file. **Note:** All UDFs must be present in the CSV file.

Returned Objects: JSON object with import results.

(**Example:** {"status":"DONE","msg":"NNN Records successfully

imported.","success":true} **OR**

If "background_job" parameter is “true”:

{"job_id":"importers:import_sources:xxx", "success":true}).

Required Parameters:

:sample_source_type=>’text’ = The name of the sample source type to be imported. (**Example:** Patient).

:file=>’.CSV file’ = File to upload(Must be

CSV format). **OR**

‘json’ = JSON

formatted string. **Optional**

Query Parameters: None

Optional Control Parameters:

:background_job=>’true/false’ = This parameter is recommended for large amounts of data (more than ~50-100 records).

Separator = a CSV file separator (comma by default).

2.14 “update_sources”:

Updates a sample source or set of sample sources via CSV.file. UDFs may be added by adding them as a column in the CSV file. **Note:** All UDFs must be present in the CSV file and the file must contain the sample source UID, Barcode, or RFID tag.

Returned Objects: JSON object with update results.

Required Parameters:

:sample_source_type=>’text’ = The name of the sample source type to be updated.

:file=>’.CSV file’ = File to upload(Must be

CSV format). **OR**

‘json’ = JSON

formatted string. **Optional**

Query Parameters: None

Optional Control Parameters:

:background_job=>'true/false' = This parameter is recommended for large amounts of data (more than ~50-100 records).

Separator = a CSV file separator (comma by default).

2.15 "import_samples":

Imports a sample or set of samples via CSV.file. UDFs may be added by adding them as a column in the CSV file. **Note:** All UDFs must be present in the CSV file.

Returned Objects: JSON object with import results.

Required Parameters:

:box_path=>'text' = Full, comma separated path of a box to import samples to. (**Example:** Freezer,Level 6,Box 1).

:file=>'.CSV file' = File to upload(Must be

CSV format). **OR**

'json' = JSON

formatted string. **Optional**

Query Parameters: None

Optional Control Parameters:

:background_job=>'true/false' = This parameter is recommended for large amounts of data (more than ~50-100 records).

:next_box=>'true/false' = Indicates to import to the next box down the freezer's hierarchy.

:subdivision_barcode=>'value' = Imports the sample/s into the first available location in the subdivision as specified by subdivision barcode.

:sample_type=>'text' = Name of the sample type to import, otherwise the CSV file should contain the "Sample Type" column.

:create_storage=>'box path' = Indicates to create freezers/racks/shelves and boxes.

:box_type=>'text' = Is required when "create_storage" is used. Indicates the box size to use (**Example:** 10x10).

Separator = a CSV file separator (comma by default).

2.16 "update_samples":

Updates a sample or set of samples via CSV.file. UDFs may be added by adding them as a column in the CSV file. **Note:** All UDFs must be present in the CSV file and the file must contain the sample unique ID, Barcode, or RFID tag.

Returned Objects: JSON object with update results.

Required Parameters:

:file=>'.CSV file' = File to upload(Must be

CSV format). **OR**

'json' = JSON

formatted string. **Optional**

Query Parameters: None

Optional Control Parameters:

:background_job=>'true/false' = (This parameter is recommended for large amounts of data (more than ~50-100 records).

Separator = a CSV file separator (comma by default).

2.17 "import_sample_groups":

Imports a sample group or set of sample groups via CSV.file. UDFs may be added by adding them as a column in the CSV file. **Note:** All UDFs must be present in the CSV file.

Returned Objects: JSON object with import results.

Required Parameters:

:file=>'.CSV file' = File to upload(Must be

CSV format). **OR**

'json' = JSON

formatted string. **Optional**

Query Parameters: None

Optional Control Parameters:

:background_job=>'true/false' = This parameter is recommended for large amounts of data (more than ~50-100 records).

Separator = a CSV file separator (comma by default).

2.18 "update_sample_groups":

Updates a sample group or set of sample groups via CSV.file. UDFs may be added by adding them as a column in the CSV file. **Note:** All UDFs must be present in the CSV file and the file must contain the sample group unique ID, Barcode, or RFID tag.

Returned Objects: JSON object with update results.

Required Parameters:

:file=>'.CSV file' = File to upload(Must be

CSV format). **OR**

'json' = JSON

formatted string. **Optional Query**

Parameters: None

Optional Control Parameters:

:background_job=>'true/false' = This parameter is recommended for large amounts of data (more than ~50-100 records).

Separator = a CSV file separator (comma by default).

2.19 "update_boxes":

Updates a box or set of boxes via CSV.file. UDFs may be added by adding them as a column in the CSV file. **Note:** All UDFs must be present in the CSV file and the file must contain the box unique ID, Barcode, or RFID tag.

Returned Objects: JSON object with update results.

Required Parameters:

:id=>'value' = Unique ID of the box to be updated.

:file=>'.CSV file' = File to upload(Must be

CSV format). **OR**

'json' = JSON formatted string.

Standard fields within the CSV file:

Name = Name of the box.

Description = Box description.

Barcode = The box's

unique barcode. **Optional Query**

Parameters: None **Optional Control**

Parameters:

:background_job=>'true/false' = This parameter is recommended for large amounts of data (more than ~50-100 records).

Separator = a CSV file separator (comma by default).

2.20 “location_info”:

Provides the location of a sample along with relevant information about that sample as specified by ID.

Returned Objects: A series of Key/Value pairs of relevant fields associated with the sample.

Required Parameters:

:id=>’ value = Unique ID of the sample to be looked up.

OR

:barcode=>value = The barcode of the sample to be looked up.

Optional Query

Parameters: None **Optional**

Control Parameters: None

2.21 “roles”:

Retrieves a list of roles within the system along with all relevant information.

Returned Objects: Roles

Required Parameters: None

Optional Query

Parameters: None **Optional**

Control Parameters: None

2.22 “sample_groups”:

Retrieves a list of all sample groups within the system along with all relevant information.

Returned Objects: SampleGroups

Required Parameters: None

Optional Query Parameters:

:query=>’text’ = Limits the list to items containing a specific text string.

Optional Control Parameters:

:start=>’value’ = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

2.23 “sample_info”:

Retrieves a list of relevant information associated with a sample as specified by ID.

Returned Objects: A series of Key/Value pairs of relevant fields associated with the sample.

Required Parameters:

:id=>’value’ = Unique ID of the sample to be looked up.

Optional Query

Parameters: None **Optional**

Control Parameters: None

2.24 “sample_source_info”:

Retrieves a list of relevant information associated with a sample as specified by ID.

Returned Objects: A series of Key/Value pairs of relevant fields associated with the sample source.

Required Parameters:

:id=>’value’ = Unique ID of the sample source to be looked up. To look up multiple sources enter multiple IDs separated by commas.

Optional Query

Parameters: None **Optional**

Control Parameters: None

2.25 “sample_source_userfields”:

Retrieves a list of relevant information associated with a sample source as specified by ID.

Returned Objects: A series of Key/Value pairs of user defined fields associated with the sample source.

Required Parameters:

:id=>’value’ = Unique ID of the sample source to be looked up.

Optional Query

Parameters: None **Optional**

Control Parameters: None

2.26 “sample_source_types”:

Retrieves a list of all sample source types within the system along with any relevant information

Returned Objects: SampleSourceTypes

Required Parameters: None

Optional Query Parameters:

:query=>’text’ = Limits the list to items containing a specific text string.

Optional Control Parameters:

:start=>’value’ = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

2.27 “upload_file_udf”:

Uploads a file to a UDF within a sample as specified by ID.

Returned Objects: Success or error message.

Required Parameters:

:file=>’.CSV file’ = File to import(Must be CSV format).

:udf_name=>’text’ = The name of the UDF to upload the file to.

Optional Query

Parameters: None **Optional**

Control Parameters: None

2.28 “put_samples_in”:

Puts samples as specified by ID back in a freezer.

Returned Objects: Success or error message.

Required Parameters:

:barcode_tags=>’value’ = The barcode tag of the sample or samples in question.

:custom_ids=>’text’ = The custom vial identifier(if set) of the sample or samples in question.

:rfid_tags=>’value’ = The RFID tag for the sample or samples in question.

Optional Query

Parameters: None **Optional**

Control Parameters: None

2.29 “take_samples_out”:

Takes samples as specified by ID out of their current freezer.

Returned Objects: Success or error message.

Required Parameters:

:barcode_tags=>'value' = The barcode tag of the sample or samples in question.

:custom_ids=>'text' = The custom vial identifier(if set) of the sample or samples in question.

:rfid_tags=>'value' = The RFID tag for the sample or samples in question.

Optional Query

Parameters: None **Optional**

Control Parameters: None

2.30 "sample_sources":

Retrieves a list of all sample sources within the system along with any relevant information.

Returned Objects: SampleSources

Required Parameters: None

Optional Query Parameters:

:id=>'value' = Limit the list to a particular sample source type

:query=>'text' = Limits the list to items containing a specific text string.

Optional Control Parameters:

:start=>'value' = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

2.31 "sample_types":

Retrieves a list of all sample types within the system along with any relevant information.

Returned Objects: SampleTypes

Required Parameters: None

Optional Query Parameters: None

Optional Control Parameters:

:start=>'value' = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

2.32 “sample_userfields”:

Retrieves a list of all user-defined fields within a sample as specified by ID.

Returned Objects: A series of Key/Value pairs of relevant fields associated with the sample.

Required Parameters:

:id=>’value’ = The unique ID of the sample in question.

Optional Query

Parameters: None **Optional**

Control Parameters: None

2.33 “samplegroup_samples”:

Retrieves a list of all samples along with relevant information within a sample group as specified by

ID. **Returned Objects:** Samples

Required Parameters:

:id=>’value’ = The unique ID of the sample group in question.

Optional Query Parameters:

:sampletype_id=>’value’ = Limit the list to a specific sample type as specified by ID.

:query=>’text’ = Limits the list to items containing a specific text string.

Optional Control Parameters:

:start=>’value’ = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

2.34 “samples_by_date”:

Retrieves a list of samples created on a specific date or within a specific date range along with any

relevant information. **Returned Objects:** Samples **Required Parameters:**

:date=>’today/yesterday/week/month/specific date’ = The specific date or date range in question.

Optional Query Parameters:

:source_id=>’value’ = Limits the list to a particular sample source.

:group_id=>’value’ = Limits the list to a particular sample group.

:query=>’text’ = Limits the list to items containing a specific text string.

Optional Control Parameters:

:start=>’value’ = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

2.35 “samples_out”:

Retrieves a list of samples that are currently out of any freezers along with any relevant information.

Returned Objects: Samples

Required Parameters: None

Optional Query Parameters: None

Optional Control Parameters: None

:start=>’value’ = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

2.36 “samples_trashbin”:

Retrieves a list of all samples currently in the trashbin along with any relevant information.

Returned Objects: Locations

Required Parameters: None

Optional Query Parameters:

:query=>’text’ = Limits the list to items containing a specific text string.

Optional Control Parameters:

:start=>’value’ = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

2.37 “samplesource_samples”:

Retrieves a list of all samples along with relevant information within a sample source as specified by ID. **Returned Objects:** Samples

Required Parameters:

:id=>’value’ = The unique ID of the sample source in question.

Optional Query Parameters:

:query=>’text’ = Limits the list to items containing a specific text string.

Optional Control Parameters:

:start=>’value’ = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

2.38 “sampletype_samples”:

Retrieves a list of all samples along with relevant information within a sample type as specified by ID. **Returned Objects:** Samples

Required Parameters:

:id=>’value’ = The unique ID of the sample type in question.

Optional Query Parameters:

:query=>’text’ = Limits the list to items containing a specific text string.

Optional Control Parameters:

:start=>’value’ = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

2.39 "search_samples":

Searches through and Retrieves all samples in the system with a specific text string or value.

Returned Objects: Samples

Required Parameters: None

Optional Query Parameters: None

:query=>'text' = Limits the list to items containing a specific text string.

Optional Control Parameters:

:start=>'value' = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

2.40 "subdivisions":

Retrieves a list of all freezers or subdivisions within a freezer as specified

by ID. **Returned Objects:** Subdivisions

Required Parameters:

:id=>'value' = The unique ID of the freezer in question.

Optional Query Parameters: None

Optional Control Parameters: None

2.41 "user_samples":

Retrieves a list of all samples currently owned by a user as specified

by ID. **Returned Objects:** Samples

Required Parameters:

:id=>'value' = The unique ID of the user in question.

Optional Query Parameters:

:query=>'text' = Limits the list to items containing a specific text string.

Optional Control Parameters:

:start=>'value' = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

2.42 "userfields":

Retrieves a list of all user defined fields in the system along with any relevant information.

Returned Objects: UserFields

Required Parameters: None

Optional Query Parameters: None

:query=>'text' = Limits the list to items containing a specific text string.

Optional Control Parameters:

:start=>'value' = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

2.43 "users":

Retrieves a list of all user in the system along with any relevant information.

Returned Objects: Users

Required Parameters: None

Optional Query Parameters: None

Optional Control Parameters: None

2.44 "vials_box":

Retrieves a list of all vials in a box as specified by ID along with any relevant

information. **Returned Objects:** Locations

Required Parameters:

:id=>'value' = The unique ID of the box in question.

Optional Query Parameters:

:query=>'text' = Limits the list to items containing a specific text string.

Optional Control Parameters:

:start=>'value' = The item specific ID to start the list with.

2.45 "vials_sample":

Retrieves a list of all vials from a sample as specified by ID along with any relevant

information. **Returned Objects:** Locations

Required Parameters:

:sample_id=>'value' = The unique ID of the sample in question.

Optional Query Parameters:

:query=>'text' = Limits the list to items containing a specific text string.

Optional Control Parameters:

:start=>'value' = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

2.46 "vials_import":

Retrieves a list of all vials imported by a job as specified by ID along with any relevant

information. **Returned Objects:** Locations

Required Parameters:

:id=>'value' = The unique ID of the job in question.

Optional Query Parameters:

:query=>'text' = Limits the list to items containing a specific text string.

Optional Control Parameters:

:start=>'value' = The item specific ID to start the list with.

:limit=>'value' = The total number of results to retrieve.

:sort=>'text' = Sort the displayed results by a specific field such as subject_type, subject_name, etc.

:dir=>'ASC/DESC' = Sort the displayed results in ascending or descending order.

2.47 "add_to_picklist":

Places Vials specified via RFID, Barcode, Custom ID or Vial ID into a Picklist specified via name or ID.

Returned Objects: Picklist ID

Required Parameters: Pick one of the following Picklist methods then one of the following Vial ID methods

:picklist_name=>'text' = The name of the Picklist to move Vials to.

:picklist_id=>'value' = The ID of the Picklist to move Vials to.

:rfid_tags="value,value" = The RFID tag/s of the Vial/s to move to a Picklist. Multiple RFID tags are separated by commas.

:barcode_tags="value,value" = The barcode tag/s of the Vial/s to move to a Picklist. Multiple barcode tags are separated by commas.

:custom_ids="value,value" = The custom ID/s of the Vial/s to move to a Picklist. Multiple IDs are separated by commas.

:vial_ids="value,value" = The vial ID/s of the Vial/s to move to a Picklist. Multiple IDs are separated by commas. **Optional Query Parameters:** None

Optional Control Parameters:

:create_picklist=>true = Specifies whether to create the Picklist if it is not found by "picklist_name".

2.48 "set_sensors":

Adds a temperature sensor reading for a freezer.

Returned Objects: Sensor Reading

Required Parameters:

:freezer_name=>'text' = The name of the freezer **OR**

:freezer_id=>'value' = The id of the freezer

:reading=>'value' The temperature reading

4. Section 3. List of API Examples

Following is a list of Example API scripts written in ruby. Each example references the corresponding API method in Section 2. List of API Functions.

3.1 Example 1 Advanced Search.rb:

```ruby
#Example #1 - This API will run an avanced subject for a Sample by variables such as Subject Type, Type(S
DF/UDF), Field, and texts strings.
#add any of the optional control parameters by copy and pasting <, function: 'value'> after <method: ''>:

    #, sdfs: 'value,value': specifies a range of sdfs
    #, udfs: 'value,value': specifies a range of udfs
    #, start: 'value': specifies what record to start listing from
    #, limit: 'value': limit number of records to retrieve
    #, sort: 'value': sort the records by a specific value
    #, dir: 'ASC/DESC': sort the records in ascending or descending order
require 'net/http'
require 'json'

url = URI.parse('http://your-freezerpro-url/api')
header = {'Content-Type'=> 'application/json'}
params = {
  username: 'admin',
  password: 'admin',
  method: 'advanced_search',
  subject_type: 'Sample',
  query: [
    {type:'sdf',
    field:'description',
    op:'contains',
    value:'test'}],
    #sdfs: ['id', 'locations_count', 'sample_source_name', 'owner_username'],
}

http = Net::HTTP.new(url.host, url.port)

if url.scheme == "https"
    http.use_ssl = true
    http.verify_mode = OpenSSL::SSL::VERIFY_NONE
end

request = Net::HTTP::Post.new(url.request_uri, header)
request.body = params.to_json
response = http.request(request)

# Option 1
#puts JSON.pretty_generate(JSON.parse(response.body))

# Option 2
data = JSON.load(response.body)
total = data['Total']
puts "Total: #{total}"
data.each do |sample_type,samples_array|
    if sample_type == 'Samples'
        samples_array.each do |show_sample|
            show_sample.each do |udf, value|
                puts (udf.to_s + ": " + value.to_s)
            end

        end
    end
end
```

3.2

Example 2 Alerts.rb:

```ruby
#Example #2 - This API will print the total number of active sample alerts
#add any of the optional control parameters by copy and pasting <, :function=>'value'> after <:method=>''
>:
    #, :start=>'value': specifies what record to start listing from
    #, :limit=>'value': limit number of records to retrieve
    #, :sort=>'value': sort the records by a specific value
    #, :dir=>'ASC/DESC': sort the records in ascending or descending order
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'alerts'#Copy and paste optional control parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
data.each do |key,value|
    if key == "Samples"
        value.each do |types|
            puts "--------------"
            types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)
            end
        end
```

3.3

```
                      will print the total number of



    end
end
```

Example 3 Audit.rb:

```
#Example #3 - This API                                Audit Records and list all related information.
    #add any of the Optional query parameters by copy and pasting <, :function=>'value'> after
    <:method=>''>
```

3.4

```
    #, :date_flag=>'all/today/yesterday/week/month': Allows user to search for a specific date
    #, :date_flag=>'date from,date to': Allows user to search a specific date range
    #add any of the optional control parameters by copy and pasting <, :function=>'value'> after
    <:method=>'' >:
    #, :start=>'value': specifies what record to start listing from
    #, :limit=>'value': limit number of records to retrieve
    #, :sort=>'value': sort the records by a specific value
    #, :dir=>'ASC/DESC': sort the records in ascending or descending order
    require 'rubygems'
    require 'json'
    require 'net/http'
    require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'audit'#Copy and paste optional query and/or control par
ameters here

res = Net::HTTP.start(url.host, url.port) do |http|
http.request(req)   22. end

            data = JSON.load(res.body)
            total = data['Total']
            puts total
```

```
            data.each do |key,value|
```

```
 1.              will  print  the  total  number  of
            if key == "AuditRec"
```

3.5

```
                  value.each do |types|
                  puts "--------------"
                  types.each do |k,v|
                  puts (k.to_s + ": " + v.to_s)
                  end           end          end
                  end
```

Example 4 Box Types.rb:

```
#Example #4 - This API
require 'rubygems'
```

3.6

will print the total number of

```ruby
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'box_types'

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "BoxTypes"
        value.each do |types|
            puts "--------------"
            types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)
            end
        end
    end
end
```

```ruby
require 'json'
```

Box Types and the ID and name of each.

3.5 Example 5 Box Userfields.rb:

will print the total number of

```ruby
#Example #5 - This API will print the total number of User Fields in a box as specified by ID.
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)s

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'box_userfields', :id=>'259'#Substitute number for box ID.

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
puts data
data.each do |key,value|
    puts (key.to_s + ": " + value.to_s)
end
```

3.6 Example 6 Boxes.rb:

will print the total number of

```
#Example #6 - This API       Boxes
#add any of the Optional query parameters by copy and pasting <, :function=>'value'> after
<:method=>''>
#, :id=>'value': Allows user to limit list of boxes to a specific freezer or subdivision
#, :user_id=>'value': Option to search boxes made by specific users
#, :show_empty=>'true/false': option to hide/show empty boxes from the result
#, :query=>'text': optional search string to filter the results.
#add any of the optional control parameters by copy and pasting <, :function=>'value'> after
<:method=>'' >:
#, :start=>'value': specifies what record to start listing from
#, :limit=>'value': limit number of records to retrieve
#, :sort=>'value': sort the records by a specific value
#, :dir=>'ASC/DESC': sort the records in ascending or descending order

require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'boxes', :show_empty=>'true'#Copy and paste optional que
ry and/or control parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
http.request(req)
end

                data = JSON.load(res.body)
                total = data['Total']
                puts total
                data.each do |key,value|
                if key == "Boxes"
                value.each do |types|
                puts "-------------"
                types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)



                end
                end    end
                end
```

3.7 Example 7 Delete Vials.rb:

3.8

```
require 'rubygems'

require 'json'     # For JSON   require 'net/http'
url = URI.parse('http://your-freezerpro-url/api')

   net = Net::HTTP.post_form(url,
   {:username=>"admin", :password=>"admin", :method=>'delete_vials',
# caller should provided at least one rfid_tags, barcode_tags, custom_ids or vial_ids parameters

   #:rfid_tags=>"355AB1CBC0000010000067D6,355AB1CBC000001000006292", # (optional) comma separated list of rfid tags
   #:barcode_tags=>"1026556,1026590", # (optional) comma separated list of barcodes
   #:custom_ids=>"Cell Line_26576,Bacteria_26554", # (optional) comma seraprted list of vial custom ids
   :vial_ids=>"25243,26539", # (optional) comma seraprted list of internal vial ids
   :comment=>"Reason for deletion" # (optional) comment for the audit log
  }
)

str = net.body
puts str
```

3.8 Example 8 Freezer Samples.rb:

.rb:

```
        #Example #8 - This API will print the total number of Samples in a freezer or subdivision as
        specified by  id and list all related information.
        #add any of the optional control parameters by copy and pasting <, :function=>'value'> after
        <:method=>'' >:
        #, :start=>'value': specifies what record to start listing from
        #, :limit=>'value': limit number of records to retrieve
        #, :sort=>'value': sort the records by a specific value
        #, :dir=>'ASC/DESC': sort the records in ascending or descending order
        require 'rubygems'
        require 'json'
        require 'net/http'
        require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'freezer_samples', :id=>'9'#Substitute number for Freeze
r ID#Copy and paste optional control parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
http.request(req)    end

                    data = JSON.load(res.body)
                    total = data['Total']
                    puts total
                    data.each do |key,value|
                    if key == "Samples"
                    value.each do |types|
                    puts "--------------"
                    types.each do |k,v|
                    puts (k.to_s + ": " + v.to_s)



                    end
                     end
                     end
                     end
```

3.9 Example 9 Freezers.rb:

```ruby
#Example #9 - This API will print the total number of freezers and the id, name and description of each.
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'freezers'

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "Freezers"
        value.each do |types|
            puts "--------------"
            types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)
            end
        end
    end
end
```

3.10 Example 10 Gen Token.rb:

```ruby
#Example #10 - This API will generate an 'auth_token' to be used for continuous imports, in order to prev
ent audit log flooding of "session started" and "session stopped"
require 'rubygems'
require 'json'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'gen_token'

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
puts data
```

#Example

3.11 Example 11 Get Perfect Box

```ruby
#11 - This API returns the ID and full location path for the "perfect" box with desired empty ce lls,
so the import method can use it
#, :freezer_name=>'value': Name of Freezer where to find "perfect" box OR
#, :container_id=>'value': ID of containing Freezer or shelf
```

.rb:

```ruby
    #, :space=>'value': Size of Box
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'get_perfect_box', :space=>'8x8', :container_id=>'16'#Copy and paste <:container_id=>'value'> over <:freezer_name=>'text'>

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
puts data
```

3.12 Example 12 Get Job Status.rb:

```ruby
    #Example #12 - This API returns the status of the import/update job for given job ID
    #add any of the optional control parameters by copy and pasting <, :function=>'value'> after
    <:method=>'' >:
    #, :cancel=>'job_id': Stops the specific Job and rollsback the transaction

require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'get_job_status',
:job_id=>'sample_group_updaters:update:e777973230cf0ef4f46d9cccfec7787f'#Copy and paste optional control parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
http.request(req)
end
```

3.13 .rb:

```
19.
                                                     as specified within a CSV file
2.

#Example
data = JSON.load(res.body)
puts data
```

Example 13 Import Sources #13 -

```
     This API imports sources

#NOTE:<, :box_path=>'Test Freezer,Level 1,empty box'> will override any path specified in the CSV
#add any of the optional control parameters by copy and pasting <, :function=>'value'> after <:method=>''
>:
    #, :background_job=>'true': this parameter is recommended for a large amount of data (more than ~50-
100 records)

require 'rubygems'
require 'json'
require 'net/http'
require 'csv'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)
user = 'admin'
password = 'admin'

job_id = nil
data = nil

File.open("./Import_sources.csv") do |csv|
    req = Net::HTTP::Post::Multipart.new url.path, :file=>UploadIO.new(csv, "text", "Import_sources.csv")
,
    :username=>'admin', :password=>'admin',
    :method=>'import_sources', :sample_source_type=>'Patient'#Copy and paste optional control parameters
here
    res = Net::HTTP.start(url.host, url.port) do |http|
        http.request(req)
    end
    data = JSON.load(res.body)
    job_id = data["job_id"]

    puts data.inspect
end
```

Example 14 Update Sources

```
     #14 - This API updates sources
#NOTE:<, :box_path=>'Test Freezer,Level 1,empty box'> will override any path specified in the CSV
#Example                                    as specified within a CSV file
2.

#add any of the optional control parameters by copy and pasting <, :function=>'value'> after <:method=>''
```

3.14 .rb:

```
1.
```

3.15 .rb:

```ruby
#Example                                        as specified within a CSV file

>
    #, :background_job=>'true': this parameter is recommended for a large amount of data (more than ~50-100 records)
require 'rubygems'
require 'json'
require 'net/http'
require 'csv'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)
user = 'admin'
password = 'admin'

job_id = nil
data = nil

File.open("./Update_sources.csv") do |csv|
    req = Net::HTTP::Post::Multipart.new url.path, :file=>UploadIO.new(csv, "text", "Update_sources.csv"),
    :username=>'admin', :password=>'admin', :method=>'update_sources', :sample_source_type=>'Patient'#Copy and paste optional control parameters here
    res = Net::HTTP.start(url.host, url.port) do |http|
        http.request(req)
    end
    data = JSON.load(res.body)
    job_id = data["job_id"]

    puts data.inspect
end
```

3.16 .rb:

Example 15 Import Samples

```
#Example                                        as specified within a CSV file

        #15 - This API imports samples
     #NOTE:<:box_path=>'Test Freezer,Level 1,empty box'> will override any path specified in the CSV
     #add any of the optional control parameters by copy and pasting <, :function=>'value'> after
     <:method=>'' >
     #:background_job=>'true' This parameter is recommended for a large amount of data (more than ~50100
     records)
     #:next_box=>'true' Indicates to import to the next box down the freezers tree
     #:subdivision_barcode=>'value' Imports the samples into the first available location in the subdivisi
     on
     #:sample_type=>'text' Name of the sample type to Import (Example: Bacteria). Otherwise the Import fil
     e should contain the sample type column.
     #:create_storage=>'true' Option to create freezers/racks/shelfs and boxes (see User Guide).
     #:box_type=>'value' Required when create_storage is used. Example: box_type=>"10 x 10". A default box
     size to use
     require 'rubygems'
     require 'json'
     require 'net/http'
     require 'csv'
     require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and
paste actual URL within the '' marks)
user = 'admin'
password = 'admin'

job_id = nil
data = nil

File.open("./Import1000.csv") do |csv|
req = Net::HTTP::Post::Multipart.new url.path, :file=>UploadIO.new(csv, "text", "Import1000.csv"),
:username=>'admin', :password=>'admin',
        :method=>'import_samples',
        :box_path=>'Freezer,Level 7,Box 1', #Enter the desired Freezer path here
        :next_box=>'true',
        :background_job=>'true' #Copy and paste optional control parameters here
        res = Net::HTTP.start(url.host, url.port) do |http|
        http.request(req)
        end
```

3.17 .rb:

```
```

3.18

```
#Example
data = JSON.load(res.body)    job_id = data["job_id"]
puts data.inspect
end
```

Example 16 Update Samples

```
        #16 - This API updates samples
#add any of the optional control parameters by copy and pasting <, :function=>'value'> after <:method=>''
```

```
>
   #, :background_job=>'true': this parameter is recommended for a large amount of data (more than ~50-
100 records)
require 'rubygems'
require 'json'
require 'net/http'
require 'csv'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)
user = 'admin'
```

3.19

```
#NOTE:<, :box_path=>'Test Freezer,Level 1,empty box'> will override any path specified in the CSV
#add any of the optional control parameters by copy and pasting <, :function=>'value'> after

<:method=>'' >:
#, :background_job=>'true': this parameter is recommended for a large amount of data (more than
~50100 records)
#separator: a CSV file separator (comma by default)


require 'rubygems'
require 'json'
```

3.20

```
    require 'net/http'
```

.rb:

```
#Example                                        as specified within a CSV file

password = 'admin'

job_id = nil
data = nil

File.open("./Update_sample.csv") do |csv|
    req = Net::HTTP::Post::Multipart.new url.path, :file=>UploadIO.new(csv, "text", "Update_sample.csv"),

     :username=>'admin', :password=>'admin', :method=>'update_samples'#Copy and paste optional control parameters here
    res = Net::HTTP.start(url.host, url.port) do |http|
        http.request(req)
    end
    data = JSON.load(res.body)
    job_id = data["job_id"]

    puts data.inspect
end
```

Example 17 Import Sample Groups.rb:

```
#17 - This API imports Sample Groups as specified within a CSV file

#Example
require 'csv'
```

3.21

```
require 'net/http/post/multipart'
url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)
user = 'admin'
password = 'admin'

job_id = nil
data = nil

        File.open("./samplegroups_import.csv") do |csv|
        req = Net::HTTP::Post::Multipart.new url.path, :file=>UploadIO.new(csv, "text",
        "samplegroups_import. csv"),
        :username=>'admin', :password=>'admin', :method=>'import_sample_groups'#Copy and paste optional
        contr ol parameters here
        res = Net::HTTP.start(url.host, url.port) do |http|
        http.request(req)
        end
        data = JSON.load(res.body)
        job_id = data["job_id"]

    puts data.inspect
    end
```

3.22 Example

```
#Example
```

18 Update Sample Groups.rb:

```
#18 - This API updates Sample Groups as specified within a CSV file
```

```
add any of the optional control parameters by copy and pasting <, :function=>'value'> after  <:method=>''>:

    #, :background_job=>'true': this parameter is recommended for a large amount of data (more than ~50-100 records)
require 'rubygems'
require 'json'
require 'net/http'
require 'csv'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)
user = 'admin'
password = 'admin'

job_id = nil
data = nil

File.open("./samplegroups_update.csv") do |csv|
    req = Net::HTTP::Post::Multipart.new url.path, :file=>UploadIO.new(csv, "text", "samplegroups_update.csv"),
    :username=>'admin', :password=>'admin', :method=>'update_sample_groups'#Copy and paste optional control parameters here
    res = Net::HTTP.start(url.host, url.port) do |http|
        http.request(req)
```

3.23 Example

```
#Example
```

```
    end
    data = JSON.load(res.body)
    job_id = data["job_id"]

    puts data.inspect
end
```

19 Update Boxes.rb:

```
    #add any of the optional control parameters by copy and pasting <, :function=>'value'> after
    <:method=>'' >:
```

```
        #19 - This API updates Boxes as specified within a CSV file
```

```
    #, :background_job=>'true': this parameter is recommended for a large amount of data (more than ~50-
100 records)
require 'rubygems'
require 'json'
require 'net/http'
require 'csv'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)
user = 'admin'
password = 'admin'

job_id = nil
data = nil

File.open("./update_boxes.csv") do |csv|
    req = Net::HTTP::Post::Multipart.new url.path, :file=>UploadIO.new(csv, "text", "update_boxes.csv"),

    :username=>'admin', :password=>'admin', :method=>'update_boxes', :id=>'259'#Copy and paste optional c
ontrol parameters here
    res = Net::HTTP.start(url.host, url.port) do |http|
        http.request(req)
    end
    data = JSON.load(res.body)
    job_id = data["job_id"]

    puts data.inspect
end
```

3.20 Example 20 Location Info.rb:

Example

```
#Example
```

```ruby
#Example #20 - This API will print the location of a Vial and all identifying information as specified by
 ID or barcode.
#   ,:id=>'value': locate a sample via ID
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'location_info',
:barcode=>'1021259'#Copy and paste sample barcode within value or replace method with ":id" from above

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    puts (key.to_s + ": " + value.to_s)
end
```

3.21 21 Roles.rb:

```
        #21 - This API will print the total number of Roles in FreezerPro and list all identifying infor
```

Example

```
#Example
```

3.23

```
mation.
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'roles'

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "Roles"
        value.each do |types|
            puts "--------------"
            types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)
            end
        end
    end
end
```

3.22 22 Sample Groups.rb:

```
#22 - This API will print a list of all Sample Groups and all related infomation


#, :query=>'text': optional search string to filter the results.
```

Example

```
#Example
        #add any of the Optional query parameters by copy and pasting <, :function=>'value'> after
        <:method=>''>

         #Optional control parameters:
         #, :start=>'value': specifies what record to start listing from
         #, :limit=>'value': limit number of records to retrieve
         #, :sort=>'value': sort the records by a specific value
         #, :dir=>'ASC/DESC': sort the records in ascending or descending order
         require 'rubygems'
         require 'json'
         require 'net/http'
         require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'sample_groups'#Copy and paste optional query and/or con
trol parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
http.request(req)    end

                data = JSON.load(res.body)
                total = data['Total']
                puts total
                data.each do |key,value|
                if key == "SampleGroups"
                value.each do |types|
                puts "--------------"
                types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)



                end
                end    end
                end
```

3.23 Example 23 Sample Info.rb:

Example

```
#Example
```

```
#Example #23 - This API will print all identifying information for the sample as specified by ID
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'sample_info', :id=>'21529'#subsitute number for Sample ID

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end


data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    puts (key.to_s + ": " + value.to_s)
end
```

3.24 24 Sample Source Info.rb:

```
#24 - This API will print all identifying information for the sample source as specified by ID
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'sample_source_info', :id=>'5'#subsitute number for Sample Source ID

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end


data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    puts (key.to_s + ": " + value.to_s)
end
```

3.25 Example 25 Sample Source Userfields.rb:

Example

```
#Example
```

```
#Example #25 - This API will print the total number of User Fields in a sample source as specified by ID.
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'sample_source_userfields', :id=>'5'#subsitute number fo
r Sample Source ID

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    puts (key.to_s + ": " + value.to_s)
end
```

3.26 26 Sample Source Types.rb:

Example

```
#Example
```

```
        #26 - This API will print a list of all Sample Source Types and all related infomation
#add any of the Optional query parameters by copy and pasting <, :function=>'value'> after <:method=>''>

    #, :query=>'text': optional search string to filter the results.
#Optional control parameters:
    #, :start=>'value': specifies what record to start listing from
    #, :limit=>'value': limit number of records to retrieve
    #, :sort=>'value': sort the records by a specific value
    #, :dir=>'ASC/DESC': sort the records in ascending or descending order
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin',
:method=>'sample_source_types'#Copy and paste optional query and/or control parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total

data.each do |key,value|
    if key == "SampleSourceTypes"
        value.each do |types|
            puts "--------------"
            types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)
            end
        end
    end
end
```

3.27 Example 27 Upload File UDF.rb:

Example

```
#Example
```

```
#Example #27 - This API will upload a file to a Sample as specified by ID
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

fname = 'Your file path and name'

File.open("#{fname}") do |f|

req = Net::HTTP::Post::Multipart.new url.path, :username=>'admin',
:password=>'admin', :method=>'upload_file_udf', :file=> UploadIO.new(f, "text", fname), :id=>'21259', :udf_name=>'File'

res = Net::HTTP.start(url.host, url.port) do |http|
http.request(req)
end

str = res.body
puts str
end
```

```
#Example
```

3.28 Example 28 Put Samples In.rb:

```
28 - This API puts samples as specified by Barcode, RFID, or Custom Vial Identifier back in the freezer they were removed from.
#Add or remove tags to put in more or fewer samples.
# ,:barcode_tags=>'value': Identifies a sample by its Barcode ID.
# ,:custom_ids=>'text: Identifies a sample by its Custom Vial Identifier.
# ,:rfid_tags=>'value': Identifies a sample by its RFID tag.

   require 'rubygems'
   require 'json'
require 'net/http'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)
json = {:tags=>['1026547,1026548'],:type=>:barcode_tags}#Copy and paste the required parameter in place of <:type> and the corresponding value in <:tags>
opts = {:username=>'admin', :password=>'admin', :method=>'put_samples_in', :json=>json.to_json}
net = Net::HTTP.post_form(url+'/api', opts)

str = net.body
puts str
```

3.29 Example 29 Take Samples Out.rb:

```
#Example 29 - This API removes samples as specified by Barcode, RFID, or Custom Vial Identifier from the freezer they are currently in.
#Add or remove tags to put in more or fewer samples.
# ,:barcode_tags=>'value': Identifies a sample by its Barcode ID.
# ,:custom_ids=>'text: Identifies a sample by its Custom Vial Identifier.
# ,:rfid_tags=>'value': Identifies a sample by its RFID tag.

   require 'rubygems'
   require 'json'
require 'net/http'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)
json = {:tags=>['1026547,1026548'],:type=>:barcode_tags}#Copy and paste the required parameter in place of <:type> and the corresponding value in <:tags>
opts = {:username=>'admin', :password=>'admin', :method=>'take_samples_out',:json=>json.to_json}
net = Net::HTTP.post_form(url+'/api', opts)

str = net.body
puts str
```

3.30 30 Sample Sources.rb:

```
#30 - This API returns the status of the import/update job for given job ID
#add any of the optional query parameters by copy and pasting <, :function=>'value'> after <:method=>''>:
#, :id=>'value': limit sample source to a particular type
#, :query=>'text': optional search string to filter the results
```

```
        #add any of the optional control parameters by copy and pasting <, :function=>'value'> after
        <:method=>'' >:
        #, :start=>'value': specifies what record to start listing from
        #, :limit=>'value': limit number of records to retrieve
        #, :sort=>'value': sort the records by a specific value
        #, :dir=>'ASC/DESC': sort the records in ascending or descending order
        require 'rubygems'
        require 'json'
        require 'net/http'
        require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'sample_sources'#add control and/or query parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
http.request(req)
end

                data = JSON.load(res.body)
                total = data['Total']
                puts total
                puts data
                data.each do |key,value|
                if key == "SampleSources"
                value.each do |types|
                puts "--------------"
                types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)



                end
                 end
                 end
                 end
```

3.31 Example 31 Sample Types.rb:

Example

```
#Example


          #31 - This API returns a list
#Optional control parameters:
    #start = <staring record>
    #limit = <limit number of records to retrieve>
    #sort = <sort_field>
    #dir = <ASC / DESC>
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'sample_types'#add control parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "SampleTypes"
        value.each do |types|
            puts "--------------"
            types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)
            end
        end
    end
end
                                                               sample types and relevant information
```

3.32 Example 32 Sample Userfields.rb:

```
#Example
                                                                  of all
```

```ruby
#Example #32 - This API will print all User Fields in a sample as specified by ID.
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'sample_userfields', :id=>'26546'#subsitute number for Sample ID

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
puts data
data.each do |key,value|
    puts (key.to_s + ": " + value.to_s)
end
```

3.35 Example

```
#Example                                    of all samples
```

SampleGroup Samples.rb:

3.36

```
#Example                              of all samples

        #33 - This API returns a list                in a group as specified by ID
#add any of the Optional query parameters by copy and pasting <, :function=>'value'> after <:method=>''>:
    #, :sampletype_id=>'value': limit sample to a particular sample type
    #, :query=>'text': optional search string to filter the results
#Optional control parameters:
    #start = <staring record>
    #limit = <limit number of records to retrieve>
    #sort = <sort_field>
    #dir = <ASC / DESC>
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'samplegroup_samples', :id=>'6'#subsitute number for Sample Group ID, Copy and paste Query and/or Control parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "Samples"
        value.each do |types|
            puts "-------------"
            types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)
            end
        end
    end
end
```

3.37 Example

```
#Example                                    of all samples
```

Example 17 Import Sample Groups.rb:

```


```

3.38

```
#Example                              of all samples

          #34 - This API returns a list                specified by date creation
#add any of the Optional query parameters by copy and pasting <, :function=>'value'> after <:method=>''>:

    #, :source_id=>'value': limit sample to a particular sample source
    #, :group_id='value': limit sample to a particular sample group
    #, :query=>'text': optional search string to filter the results
#Optional control parameters:
    #start = <staring record>
    #limit = <limit number of records to retrieve>
    #sort = <sort_field>
    #dir = <ASC / DESC>
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'samples_by_date', :date=>'today'#<today|yesterday|week|month|"date">#Copy and paste optional query and/or control parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "Samples"
        value.each do |types|
            puts "-------------"
            types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)
            end
        end
    end
end
```

3.39 Example

```
#Example                                              of all samples
```

35 Samples Out.rb:

3.40

```
#Example                                of all samples

            #35 - This API returns a list                   currently not in a freezer and all related informat
ion
#Optional control parameters:
    #start = <staring record>
    #limit = <limit number of records to retrieve>
    #sort = <sort_field>
    #dir = <ASC / DESC>
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'samples_out'#Copy and paste optional control parameters
 here

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "Samples"
        value.each do |types|
            puts "-------------"
            types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)

        end
    end
end
               end
```

3.41 Example

```
#Example
                                        of all samples
                                                                  36 Samples Trashbin.rb:
```

3.42 Example

```
#Example                              of all samples
```

37 SampleSource Samples.rb:

```
#37 - This API returns a list                  from a source as specified by sample source ID
```

```
                end
            end
        end
    end
        #36 - This API returns a list                  currently in the trashbin and all related informati
    on
#add any of the Optional query parameters by copy and pasting <, :function=>'value'> after <:method=>''>:
    #, :query=>'text': optional search string to filter the results
#Optional control parameters:
    #start = <staring record>
    #limit = <limit number of records to retrieve>
    #sort = <sort_field>
    #dir = <ASC / DESC>
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'samples_trashbin'#Copy and paste optional query and/or
control parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "Locations"
        value.each do |types|
            puts "--------------"
```

3.43 Example

```
#Example                              of all samples
  #add any of the Optional query parameters by copy and pasting <, :function=>'value'> after

     #, :query=>'text': optional search string to filter the results
     #Optional control parameters:
     #start = <staring record>
     #limit = <limit number of records to retrieve>
     #sort = <sort_field>
     #dir = <ASC / DESC>
     require 'rubygems'
     require 'json'
     require 'net/http'
     require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'samplesource_samples', :id=>'2'#Substitute number for sample source ID.#Copy and paste optional query and/or control parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
http.request(req)
end

                data = JSON.load(res.body)
                total = data['Total']
end
                puts total
                data.each do |key,value|
                if key == "Samples"
<:method=>''>:
```

3.44 Example

```
#Example                                        of all samples
value.each do |types|
puts "--------------"
types.each do |k,v|        puts (k.to_s + ": " + v.to_s)
```

38 SampleType Samples.rb:

```
                end
            end

end
        end
```

3.45 Example

```
#Example                              of all samples
```

```
            end
        end
    end
end
         #38 - This API returns a list                from a type as specified by sample type ID
#add any of the Optional query parameters by copy and pasting <, :function=>'value'> after <:method=>''>:

    #, :query=>'text': optional search string to filter the results
#Optional control parameters:
    #start = <staring record>
    #limit = <limit number of records to retrieve>
    #sort = <sort_field>
    #dir = <ASC / DESC>
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'sampletype_samples', :id=>'20'#Substitute number for sample type ID.#Copy and paste optional query parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "Samples"
        value.each do |types|
            puts "-------------"
            types.each do |k,v|
```

3.46 Example

```
#Example                                        of all samples
```

39 Search Samples.rb:

```
end
```

```
                  end
            end

end
```

3.47 Example

```
#Example                              of all samples
```

```
            end
        end
    end
end
         #39 - This API returns a list                containing the queried text
#add any of the Optional query parameters by copy and pasting <, :function=>'value'> after <:method=>''>:

    #, :query=>'text': optional search string to filter the results
#Optional control parameters:
    #start = <staring record>
    #limit = <limit number of records to retrieve>
    #sort = <sort_field>
    #dir = <ASC / DESC>
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'search_samples', :query=>'asdf'#Copy and paste optional
 query and/or control parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "Samples"
        value.each do |types|
            puts "-------------"
            types.each do |k,v|
```

3.48 Example

```
#Example
```

40 Subdivisions.rb:

```
#40 - This API will print the total number subdivisions in a Specific Freezer list the user fields
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'subdivisions', :id=>'15'#Specific Freezer/parent division ID

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "Subdivisions"
        value.each do |types|
            puts "--------------"
            types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)
```

3.49 Example

```
#Example
                end
            end
        end
end
```

41 User Samples.rb:

3.50 Example

```
#Example                              a list of all

         #41 - This API returns a list of all samples for a user as specified by user ID
#add any of the Optional query parameters by copy and pasting <, :function=>'value'> after <:method=>''>:

    #, :query=>'text': optional search string to filter the results
#Optional control parameters:
    #start = <staring record>
    #limit = <limit number of records to retrieve>
    #sort = <sort_field>
    #dir = <ASC / DESC>
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'user_samples', :id=>'3'#Substitute number for user ID#C
opy and paste optional query and/or control parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "Samples"
        value.each do |types|
            puts "-------------"
            types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)
            end
        end
    end
end
```

42 Userfields.rb:

3.51 Example

```
#Example
            #42 - This API returns                User Fields and relevant information
#add any of the Optional query parameters by copy and pasting <, :function=>'value'> after <:method=>''>:

    #, :query=>'text': optional search string to filter the results
#Optional control parameters:
    #start = <staring record>
    #limit = <limit number of records to retrieve>
    #sort = <sort_field>
    #dir = <ASC / DESC>
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'userfields'#Copy and paste optional query and/or contro
l parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "UserFields"
        value.each do |types|
            puts "-------------"
            types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)
            end
        end
    end
end
```

3.52 Example

```
#Example
```

43

Users.rb:

```
        #43 - This API will print the total number of users in FreezerPro and list all identifying infor
    mation.
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'users'

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "Users"
        value.each do |types|
            puts "-------------"
            types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)
            end
        end
    end
end
```

44 Vials Box.rb:

```
#44 - This API returns with relevant information within a Box as specified b y box ID
                                   a list of all vials
```

3.53 Example

```
#Example


#add any of the Optional query parameters by copy and pasting <, :function=>'value'> after <:method=>''>:

    #, :query=>'text': optional search string to filter the results
#Optional control parameters:
    #start = <staring record>
    #limit = <limit number of records to retrieve>
    #sort = <sort_field>
    #dir = <ASC / DESC>
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'vials_box', :id=>'259'#Substitute number for box ID#Copy and paste optional query and control parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "Locations"
        value.each do |types|
            puts "--------------"
            types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)
            end
        end
    end
end
```

45 Vials Sample.rb:

```
#45 - This API returns th relevant            from a particular sample as specified by ID, along wi
information
                                a list of all vials
```

3.54 Example

```ruby
#Example
```

```ruby
#add any of the Optional query parameters by copy and pasting <, :function=>'value'> after <:method=>''>:
```

```ruby
    #, :query=>'text': optional search string to filter the results
#Optional control parameters:
    #start = <staring record>
    #limit = <limit number of records to retrieve>
    #sort = <sort_field>
    #dir = <ASC / DESC>
require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'vials_sample', :sample_id=>'26549'#Substitute number fo
r box ID#Copy and paste optional query and/or control parameters here

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "Locations"
        value.each do |types|
            puts "--------------"
            types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)
            end
        end
    end
end
```

46 Vials Import.rb:

a list of all vials

3.55 Example

```
#Example

            #46 - This API returns                     imported by a specified Job ID

require 'rubygems'
require 'json'
require 'net/http'
require 'net/http/post/multipart'

url = URI.parse('http://your-freezerpro-url/api')#Copy and paste actual URL within the '' marks)

req = Net::HTTP::Post::Multipart.new url.path,
:username=>'admin', :password=>'admin', :method=>'vials_import',
:job_id=>'importers:import:366a7f3e46575e77227b2acf6791a0f6'#Substitute vlaue for required job ID.

res = Net::HTTP.start(url.host, url.port) do |http|
    http.request(req)
end

data = JSON.load(res.body)
total = data['Total']
puts total
data.each do |key,value|
    if key == "Locations"
        value.each do |types|
            puts "--------------"
            types.each do |k,v|
                puts (k.to_s + ": " + v.to_s)
            end
        end
    end
end
```

3.47 Example 46 Add to Picklist.rb:

```
    require 'rubygems'
    require 'json'    # For JSON   3. require 'net/http'   4.
url = URI.parse('http://your-freezerpro-url/api')

    net = Net::HTTP.post_form(url,
    {:username=>"admin", :password=>"admin", :method=>'add_to_picklist',   9.
# caller should provide one of picklist_name of picklist_id parameters

   #:picklist_name=>"LR Picklist", # (optional) the name of a pick list
   :picklist_id=>"2",             # (optional) the interal ID of a pick list
   :create_picklist=>true,         # (optional) if true a pick list will be created if it is not found by
   the picklist_name

# caller should provided at least one rfid_tags, barcode_tags or vial_ids parameters

   #:rfid_tags=>"355AB1CBC0000010000067BF,355AB1CBC0000010000067AB", # (optional) comma separated list of
   rfid tags
   :barcode_tags=>"1026582,1026590", # (optional) comma separated list of barcodes
   #:custom_ids=>"Cell Line_26576", # (optional) comma separated list of vial custom ids   21.
   #:vial_ids=>"26556" # (optional) comma separated list of internal vial ids


                                    a list of all vials
```

3.56 Example

```
#Example
  }
)

str = net.body
puts str
```