How-To Guide:

Integrate with the Storage Commander API.

OVERVIEW

This guide provides step-by-step actions for integrating with the Storage Commander API Including but not limited to:

- Getting Started
- Unit API Process
- Reservation API Process
- Move-in API Process
- Payment API Process
- DocuSign API Process
- Storage Commander Pay- Tokenizing Plain Text Card Data

Please click on the desired topic above to take you directly to the process.

Note: If you require additional help or have questions related to the API, please reach out to the Storage Commander API team at apisupport@storagecommander.com.

Getting Started

To give Storage Commander customers the best opportunity to automate their business, we provide our partners with the ability to build directly onto our API. With that being said, the Storage Commander API follows standard REST architecture as well as accepts and returns **JSON** formatted data. Below are elements that are important to understand before you move forward with the integration of the Storage Commander API.

Common terms, global variables, and other items to note before you begin:

- ***usr*** – is the Guid issued when you signed up to integrate with Storage Commander's API. This will not change from facility to facility and may be stored as a static value.
- ***loc*** – is the unique identifier for the storage facility. It needs to be string and in the following Guid format: "00000000-0000-0000-0000-000000000000"
- **Every http request must include these two headers:**
    - Ocp-Apim-Subscription-Key
        - This is the unique API key that was given after registration was confirmed with the Storage Commander API in the Azure Developer Portal. This value will not change from facility to facility and may be stored as a static value.
    - ContentType
        - application/json

Unit API Process

Each unit in the Storage Commander API is comprised of several properties and three other objects: a Unit Features array, a Unit Type, and a Size.

Below is a list of JSON object definitions.

Unit Feature

```
{
 "Id": "00000000-0000-0000-0000-000000000000",
 "Description": null,
 "ImageId": "00000000-0000-0000-0000-000000000000",
 "Image": "" (this will be a byte array of the image)
}
```

Unit Type

```
{
 "Id": "00000000-0000-0000-0000-000000000000",
 "Description": null
}
```

Unit

```
{
 "Id": "00000000-0000-0000-0000-000000000000",
 "Name": null,
 "SortId": null,
 "Size": {
  "Id": "00000000-0000-0000-0000-000000000000",
  "Description": null,
  "Width": 0,
  "Length": 0,
  "DefaultMonthlyRate": 0,
  "ManagedMonthlyRate": 0,
  "AutoAppliedDiscounts": null
 },
 "Type": null,
 "Status": null,
 "Condition": null,
 "Features": [],
 "Floor": null
}
```

Note: If a unit's "ManagedMonthlyRate" field has a value greater than zero, that price should be displayed/used otherwise use the "DefaultMonthlyRate" in its place.

Note: If a unit has a discount or set of discounts in the "AutoAppliedDiscount" area, then it will be automatically applied during a move-in. See the Template Endpoint for Discount definition.

Issuing a GET request to the following Endpoint will return an array of units for a facility.

```
https://api.storagecommander.net/api/Units?usr={userId}&loc={locationId}
```

Reservation API Process

For the Storage Commander customers, placing a unit on *hold (Reserved)* is beneficial because it takes the unit out of available status. Reservations follow a few general steps in the Storage Commander API including:

1. Get a Cart for an available unit (the unit unique identifier must be known to retrieve a Cart. See “Units” section for details on retrieving UUIDs for units).
2. Optional – recalculate a Cart, if necessary, when adding/removing items such as discounts and insurance to update the total due at the time of reservation.
3. Submit the ReservationData object in the body of a Post to the Reservations endpoint.

To retrieve a Cart, issue a Get request to the following endpoint:

```
https://api.storagecommander.net/api/Reservations/{filter}/{identifier}?usr={userId}&loc={locationId}
```

Filter – acceptable values are “size” or “unit”. This will either return a Cart for an available unit within a size or a Cart for a specific available unit.

Identifier – this is the UUID for either the size or the unit depending on the {filter} value.

A **successful** response will return the Cart in the body. The Template endpoint can be used to retrieve the definition of the TransactionCart (Cart), TransactionCartItem for the “ItemsDueDetail” array, Entity (Tenant), and AssociatedListItem for the “RecurringItems” array.

Once the reservation is confirmed and is ready to process, the ReservationData object (definition available in the Template endpoint) must be submitted as JSON in the body of a Post request to the following endpoint

```
https://api.storagecommander.net/api/Reservations?usr={userId}&loc={locationId}
```

A successful response will contain a **UUID** as text which represents the Contract ID for that reservation.

An example of C# code detailing the reservation processor through the API may be downloaded here: https://empowerstorage.blob.core.windows.net/downloads/Reservation%20Example.cs

Payment API Process

Payments through the Storage Commander API consist of several steps:

1. Acquire a TransactionCart object, generally for a move-in or an existing tenant's payment, to acquire a total due to charge.
2. Create a tokenized representation of raw credit card data to charge through the Storage Commander API (for further details please see section "Storage Commander Pay – Tokenizing Plain Text Card Data").
3. Charge the required amount through the Processors endpoint.
4. Upon successful charging of the tokenized card, submit the payment along with a Cart to one of several endpoints including MoveIns or Payments.

Retrieving a Transaction Cart

To retrieve a transaction cart for a payment on an existing tenant, issue a Get request to the following URL:

```
https://api.storagecommander.net/api/Payments/{contractId}?usr={userId}&loc={locationId}
```

The above link will automatically pull the balance on the account or one month's worth of charges depending on the tenant's current outstanding balance or lack thereof.

To specify a particular number of billing plans, an optional billing plan specifier can be places in the URL as an integer value

```
https://api.storagecommander.net/api/Payments/{contractId}/{billingCycles}?usr={userId}&loc={locationId}
```

A successful response will return a TransactionCart object with a total due for the selected billing periods.

Charging a Previously Acquired Token

Once a credit card token has been created, the token string will be placed in the "Token" field of a PaymentInfo object (JSON definition available in the Template endpoint).

The amount will be required in the form of a decimal/double along with the PaymentType fields value set to "CreditCard". Any other available information should be placed in the appropriate fields.

NOTE: The format for the credit card expiration should be **MMYY** without any special characters or spaces.

Issue a Put request with the PaymentInfo serialized as JSON in the body to the Processor endpoint:

```
https://api.storagecommander.net/api/Processor?usr={userId}&loc={locationId}
```

A successful charging of the tokenized credit card will result in the ApprovalCode field having a string value in it (except for debit cards) and the ProcessorSuccessResult field having a value of "Approved". Other values include "Decline", "Error", and "Pending" which should be reflected to the user as an unsuccessful payment.

Submitting the Payment

At this point the credit card has been successfully charged. The PaymentInfo object can be placed along with the Cart and submitted to the Payments endpoint in the body of a Post request to complete a payment. The process is the same for move-ins as well. For JSON definitions of the PaymentData or MoveInData objects, please use the Template endpoint.

Move-In API Process

Move-Ins follow two to three general steps in the Storage Commander API:

1. Get a cart for an available unit (the unit unique identifier must be known to retrieve a Cart. See “Units” section for details on retrieving UUIDs for units).
2. Optional – recalculate a Cart, if necessary, when adding/removing items like discounts and insurance to update the total due at the time of move-in.
3. Submit the MoveInData object in the body of a Post to the MoveIns endpoint.

To retrieve a Cart, issue a Get request to the following endpoint:

```
https://api.storagecommander.net/api/MoveIns/{filter}/{identifier}?usr={userId}&loc={locationId}
```

Filter – acceptable values are “size” or “unit”. This will either return a Cart for an available unit within a size or a Cart for a specific available unit.

Identifier – this will be the UUID for either the size or the unit depending on the {filter} value.

A **successful** response will return the Cart in the body. The Template endpoint can be used to retrieve the definition of the TransactionCart (Cart), TransactionCartItem for the “ItemsDueDetail” array, Entity (Tenant), and AssociatedListItem for the “RecurringItems” array.

Once the move-in is confirmed and is ready to process, the MoveInData object (definition available in the Template endpoint) must be submitted as JSON in the body of a Post request to the following endpoint

```
https://api.storagecommander.net/api/MoveIns?usr={userId}&loc={locationId}
```

A **successful** response will contain a **UUID** as text which represents the Contract ID for that move-in.

An example of C# code detailing the move-in process through the API may be downloaded here:

https://empowerstorage.blob.core.windows.net/downloads/MoveInExampleCodeC%23.cs

DocuSign API Process

Many facilities will require that their rental contracts be signed when a tenant moves in online. Storage Commander has created an integration with DocuSign to take advantage of their signing features.

If your client needs electronic signatures, the first step will be to send an email to apisupport@storagecommander.com and let them know that you need to test the DocuSign integration. At this point, an engineer will set up the DocuSign test credentials and sample rental contract in your **sandbox environment**.

After a move-in has been successfully executed, the body of the return will contain a **UUID (Guid)** as text. The Guid is the new ContractID for the rental and is necessary for the DocuSign endpoint.

```
https://api.storagecommander.net/api/DocuSign/{contractId?usr={userId}&loc={locationId}&callbackUrl={callbackUrl}
```

IMPORTANT NOTE: *during development an additional parameter is required in order to trigger the testing credentials through DocuSign. Add "dbg=true" for it to complete successfully.*

Issue a Get request to the above URL. The new contract identifier is placed in the {contractId} area. The callbackUrl parameter is the **fully qualified** URL to return to after the renter has signed the DocuSign contract. For example, www.storagecommander.com will not work. DocuSign will reject the endpoint or throw an error. The correct format would be https://www.storagecommander.com. Any necessary query parameters can also be placed in the callback if necessary.

A successful Get request will return a URL generated by DocuSign. Navigating to the URL will initiate an embedded signing session for the tenant to sign the document.

Upon completion of the signing session, the user should be redirected to the callbackUrl provided.

Should there be any questions or concerns please send an email to apisupport@storagecommander.com and one of the engineers will assist you.