1. Overview

With the Mail Sorting Machine API, you can significantly reduce the amount of time machine operators spend entering job information and easily set up a cycle on your sorting machine. The Mail Sorting Machine API uses HTTP and is RESTful.

We recommend using the ASP.NET Web API for your requests. To begin, add the Microsoft.AspNet.WebApi.Client library in your application. Then, after declaring and instantiating an `HttpClient` object, use that object for your requests, as demonstrated in the following example.

```
private void button_Click(System.Object sender, System.EventArgs e)
{
     HttpClient client = new HttpClient();
     client.BaseAddress = new Uri("http://localhost:51781/api/");

     string returnString;

     returnString = client.DownloadString("Default");
     label.Text = returnString;
}
```

2. Accessing the API

There are two ways to access the Mail Sorting Machine API.

- **Accessing the API Internally**: For production use, and when accessing the API inside the <COMPANY> secure network, use the following URI:

  ```
  http://example.com/example-production-endpoint
  ```

- **Accessing the API Externally:** For testing use, and when accessing the API outside of the <COMPANY> secure network, use the following URI:

  ```
  http://example.com/example-sandbox-endpoint
  ```

3. Cycle

The `/cycle` endpoint is used to identify and create cycles (also known as "passes").

Create a new cycle when a machine operator successfully scans a job sheet into a sorting machine. Also, create a new cycle when a job sheet contains a changed mail class or mail type. Do not create a new cycle if the mail class or mail type has not changed between scanned job sheets.

For examples on when to create new cycles, refer to the following table.

Job Sheet ID	Mail Type	Mail Class	New Cycle?
100	Letter	First-Class	Yes (start of day)
100	Letter	First-Class	No
100	Letter	First-Class	No
101	Flat	First-Class	Yes (new mail type)
101	Flat	First-Class	No
102	Flat	First-Class	No
102	Flat	First-Class	No
103	Letter	Standard-Class	Yes (new mail class)

3.1. Request methods

The `/cycle` endpoint supports the following HTTP methods and URLs.

Method	Production URL	Sandbox URL
GET	http://example.com/example-production-endpoint/cycle/{cycleID}	http://example.com/example-sandbox-endpoint/cycle/{cycleID}
POST	http://example.com/example-production-endpoint/cycle	http://example.com/example-sandbox-endpoint/cycle

3.2. Request query parameters

For POST requests, the `/cycle` endpoint uses optional query parameters, such as:

Parameter	Example	Notes
SorterName	http://example.com/example-production-endpoint/cycle?SorterName=PMT003	Returns a single cycle record, if found.
CycleDescription	http://example.com/example-production-endpoint/cycle?CycleDescription=PMT_18_05_07_07_1103	Returns a single cycle record, if found.

3.3. Request body parameters

For POST requests, the `/cycle` endpoint uses optional and required body parameters via a JSON transaction object, such as:

Name	Type	Description	Required?	Example
mailTypeID	Integer	The type of mail, identified as follows: 1 = letters (default) 2 = flats	True	1
mailClassTypeID	Integer	The class of mail, identified as follows: 1 = first-class (default) 3 = standard-class	True	3
…	…	…	…	…

3.4. Response body

If successful, a GET request for the `/cycle/{cycleID}` endpoint returns a response body via a JSON transaction object with the following structure:

```
{
  "cycleID": Integer,
  "mailTypeID": Integer,
  "mailClassTypeID": Integer,
  "sorterName": String,
  "sorterID": Integer
}
```

For more information on supported fields, see the table below.

Name	Type	Description	Example
cycleID	Integer	A unique identifier for a cycle; must be unique for a sorting machine.	123
mailTypeID	Integer	The type of mail, identified as follows: 1 = letters (default) 2 = flats	1
mailClassTypeID	Integer	The class of mail, identified as follows: 1 = first-class (default) 3 = standard-class	3
sorterName	String	A unique name for a sorting machine.	"FCM_LTR"
sorterID	Integer	A unique identifier for a sorting machine.	500

3.5. Samples

Provides sample requests and responses for the `/cycle` endpoint.

3.5.1. Sample GET request

```
GET http://example.com/example-production-endpoint/cycle/{cycleID}
```

3.5.2. Sample GET response

If successful, the request returns the HTTP status of 200 and a JSON transaction object.

For example responses, refer to the following table.

Status	Response

200	{
	"cycleID": 123,
	"mailTypeID": 1,
	"mailClassTypeID": 1,
	"sorterName": PMT018",
	"sorterID": 527
	}

404	Resource Not found
503	Service Unavailable

3.5.3. Sample POST request

```
POST http://example.com/example-production-endpoint/cycle

BODY {

    "MailTypeID": 1,

    "MailClassTypeID": 1

}
```

3.5.4. Sample POST response

If successful, the request creates a cycle and returns the HTTP status of 201.

For an example response, refer to the following table.

Status	Response
201	"Cycle created at: http://example.com/example-production-endpoint/cycle/{cycleID}"