Tool 8.7.0 - 9.0.x

24/10/2025 2:16 pm BST

Tool
Tags: API

Versions

Versions this documentation is relevant for:

- *8.7.0 - 9.0.x: This document*
- *9.1.0+*

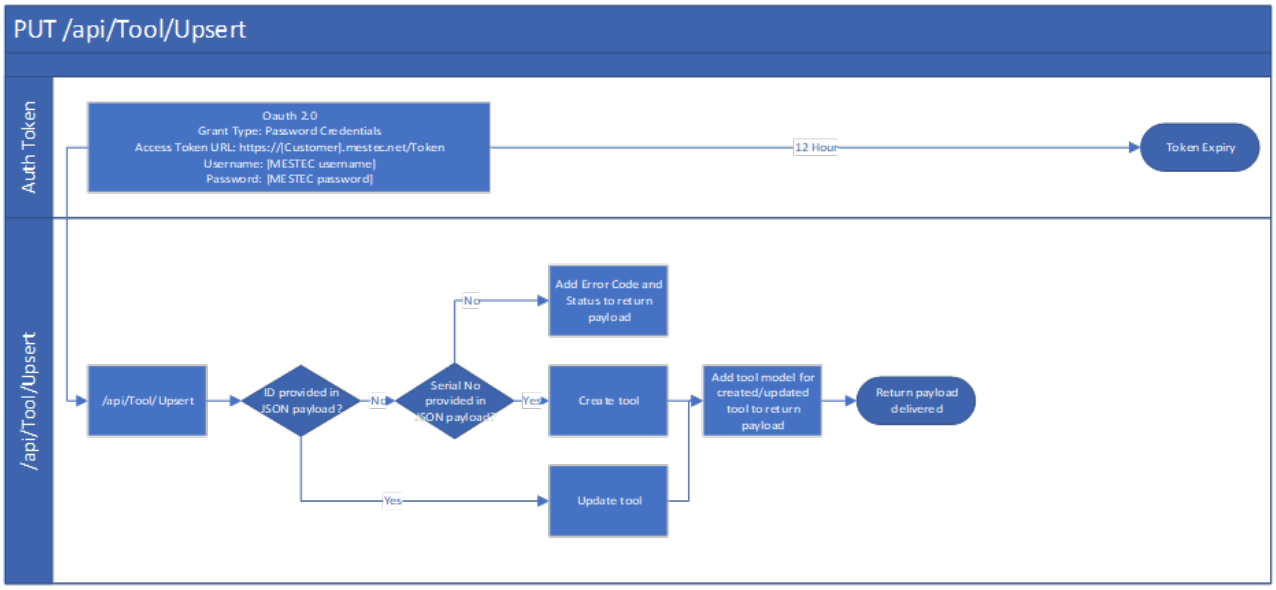
A tool is a piece of equipment used to aid in the manufacturing process. Tools could be things such as hammers and spanners or things such as jigs.

Upsert

When tool ID is provided, the Tool – Upsert API call checks if a tool with the given ID already exists. If it does, the appropriate fields are updated in the matching tool in Eyelit MES-M.

When the tool ID is omitted, a new tool will be created.

Figure 1 - Logic within Tool/Upsert API Call



Data Prerequisites

There is no data required in Eyelit MES-M to act as a pre-requisite to make the API call to upsert tools. If parameters are passed in through the body that return no valid results a new tool will be created, otherwise the tool identified will be updated.

Request

Table 1 shows the method and endpoint required to make the API call to upsert tools.

Table 1 - Outbound Message Detail for Tool - Upsert

Method	URL Structure	Endpoint
PUT	https://[environment].mestec.net	/api/Tool/Upsert

The body of the payload should follow the format below.

JSON Structure for Tool Upsert

```
{
  "id": 0,
  "serialNo": "string",
  "toolType": "string",
  "status": "string",
  "location": "string",
  "approvalStatus": "string",
  "calibrationDT": "2025-06-06T08:25:03.047Z",
  "suid": "string",
  "properties": [
    {
      "name": "string",
      "value": "string"
    }
  ]
}
```

See Table 2 for information on which fields are optional, the appropriate data types and the mappings to fields in Eyelit MES-M.

Table 2 – Parameter Information for Tool – Upsert

Parameter Name	Data Mapping	Data Type	Mandatory		Case Sensitive	Match Type
			Create	Update		
id	TOOL.ID	Integer	No	Yes	N/A	Exact
serialNo	TOOL.SERIALNO	String	Yes	No	N/A	N/A
toolType	TOOLTYPE.NAME	String	Yes	No	N/A	Exact
status	TOOLSTATUS.NAME	String	Yes	No	N/A	Exact
location	LOCATION.NAME	String	Yes	No	N/A	Exact
approvalStatus	TOOLAPPROVALSTATUS.NAME	String	Yes	No	N/A	Exact
calibrationDT	TOOL.CALIRATIONDT	DateTime	No	No	N/A	N/A
suid	TOOL.SUID	String	No	No	N/A	N/A
properties	N/A	JSON Array	No	No	N/A	N/A
properties\name	TABLEPROPERTYDEFINITION.NAME	String	No	No	N/A	Exact

properties\value	TOOLPROP.VALUE	String	No	No	N/A	N/A

Note: For any fields where the match type is 'Like ', a percent symbol should be used as a wildcard character to indicate a number of characters within the given string.

Sample Request

See below for sample use cases with examples of the JSON payload format required.

To create a tool:

JSON Sample for Tool Upsert to create a new tool

```
{
  "serialNo": "Hammer1234",
  "toolType": "Hammer",
  "status": "Available",
  "location": "Hammer Bay 1",
  "approvalStatus": "Approved",
  "calibrationDT": "2025-06-06T08:25:03.047Z",
  "suid": "Hammer1234",
  "properties": [
    {
      "name": "Size",
      "value": "10cm"
    }
  ]
}
```

To update a tool's status, location, and approval status:

JSON Sample for Tool Upsert to update a tool

```
{
  "id": 12345,
  "status": "Decommisioned",
  "location": "Storage Unit A",
  "approvalStatus": "Unapproved"
}
```

Response

When using the Tool – Upsert API call, a JSON payload will be returned containing data in the following structure:

JSON Structure for Tool Upsert

```
{
  "id": 0,
  "serialNo": "string",
  "toolType": "string",
  "status": "string",
  "location": "string",
  "approvalStatus": "string",
  "calibrationDT": "2025-06-06T14:46:05.970Z",
  "suid": "string",
  "properties": [
    {
      "name": "string",
      "value": "string"
    }
  ]
}
```


Sample Response

See below for sample use cases with examples of the JSON payload format returned.

JSON Sample for Tool Upsert


```
{
  "id": 12345,
  "serialNo": "S12345",
  "toolType": "Spanner",
  "status": "Available",
  "location": "Storage",
  "approvalStatus": "Approved",
  "calibrationDT": "2025-06-10T10:30:44.771Z",
  "suid": "S12345",
  "properties": [
    {
      "name": "Size",
      "value": "2.5mm"
    }
  ]
}
```

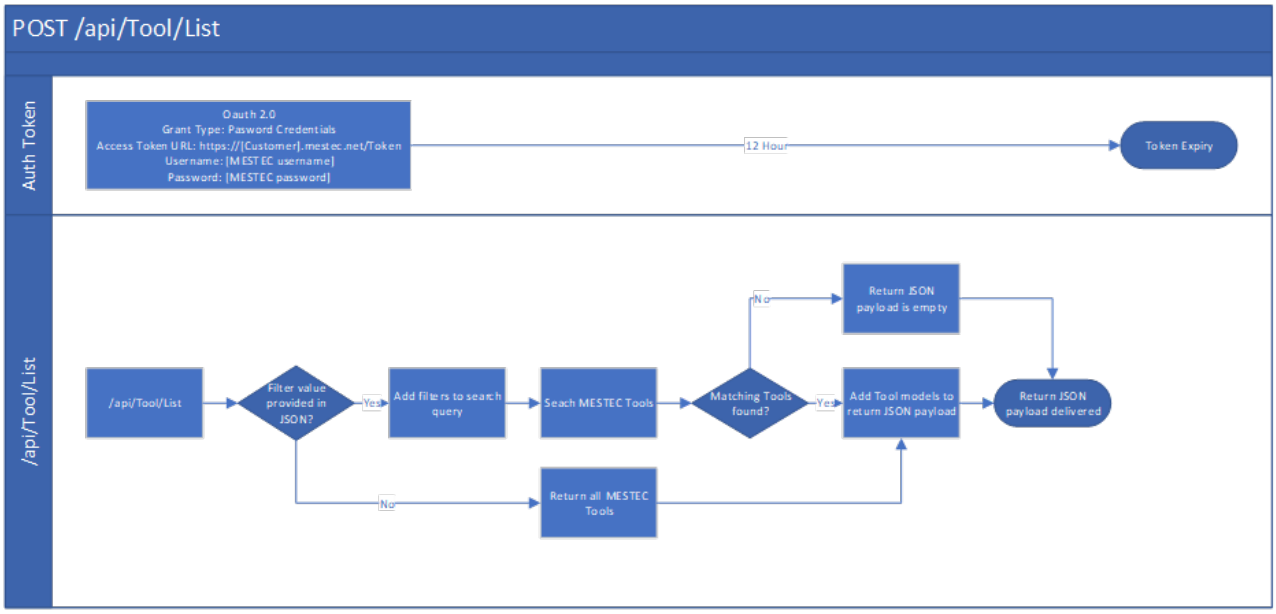

List

The Tool – List API call reads existing tools from Eyelit MES-M. The data can be filtered based on the parameters passed in the body/payload of the JSON packet giving the flexibility to search for one or multiple tools.

Using the Tool – List API call has no impact on the data within the given Eyelit MES-M application, it is read-only.

Figure 2 shows the logic used within the Tool – List API call.

Figure 2 - Logic within Tool/List API Call



Data Prerequisites

There is no data required in Eyelit MES-M to act as a pre-requisite to make the API call to list tools. If parameters are passed in through the body that return no valid results, an empty payload will be returned.

Request

Table 3 shows the method and endpoint required to make the API call to list tools.

Table 3 - Outbound Message Detail for Tool- List

Method	URL Structure	Endpoint
POST	https://[environment].mestec.net	/api/Tool/List

The body of the payload should follow the format below.

JSON Structure for Tool Upsert

```
{
  "id": 0,
  "serialNo": "string"
}
```

See Table 4 for information on which fields are optional, the appropriate data types and the mappings to fields in Eyelit MES-M.

Table 4 – Parameter Information for Tool – List

Parameter Name	Data Mapping	Data Type	Mandatory	Case Sensitive	Match Type
id	TOOL.ID	Integer	No	N/A	Exact
serialNo	TOOL.SERIALNO	String	No	N/A	Exact

Note: For any fields where the match type is 'Like', a percent symbol should be used as a wild card character to indicate a

number of characters within the given string.

Sample Request

See below for sample use cases with examples of the JSON payload format required.

To list all tools:

JSON Structure for Tool List – list all tools

```
{

}
```

To list tools where the serial number is “Tool12345”:

JSON Structure for Tool List – list a specific tool

```
{
  "serialNo": "Tool12345"
}
```

Response

When using the Tool – List API call, a JSON payload will be returned containing data in the following structure:

JSON Sample for Tool List

```
[
  {
    "id": 0,
    "serialNo": "string",
    "toolType": "string",
    "status": "string",
    "location": "string",
    "approvalStatus": "string",
    "calibrationDT": "2025-06-10T10:30:44.774Z",
    "suid": "string",
    "properties": [
      {
        "name": "string",
        "value": "string"
      }
    ]
  }
]
```

Sample Response

See below for sample use cases with examples of the JSON payload format returned.

No results were found that matched the given parameters:


```
[

]
```


Tool found that matched the given parameters:

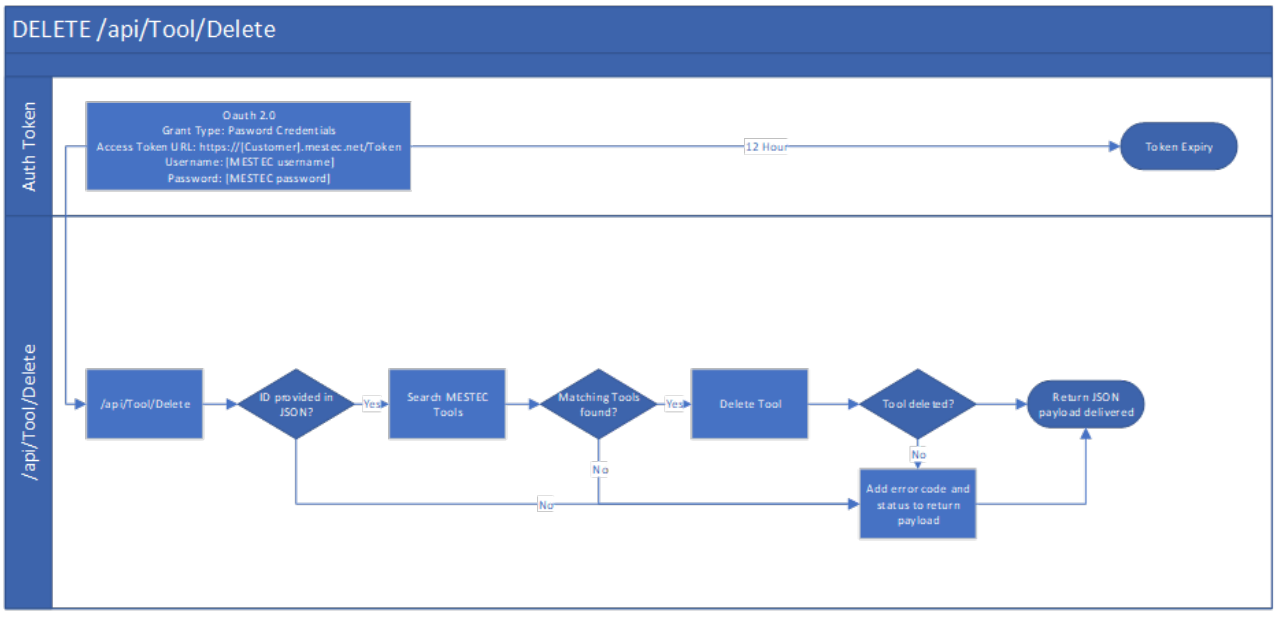

```
[
  {
    "id": 12345,
    "serialNo": "Tool12345",
    "toolType": "Hammer",
    "status": "Available",
    "location": "Hammer Storage",
    "approvalStatus": "Approved",
    "calibrationDT": "2025-06-10T10:30:44.774Z",
    "suid": "T12345"
  }
]
```


Delete

The Tool – Delete API call attempts to delete a tool. A tool can only be deleted if not referenced against any other entities (e.g., Tool Group).

Figure 3 shows the logic used within the Tool – Delete API call.

Figure 3 – Logic within Tool/Delete API call



Prerequisites

In order to delete a tool, the tool must exist in Eyelit MES-M.

The tool has not been used and not been referenced against any other entities.

Request

Table 5 shows the method and endpoint required to make the API call to delete tools.

Table 5 - Outbound Message Detail for Tool - Delete

Method	URL Structure	Endpoint
DELETE	https://[environment].mestec.net	/api/Tool/Delete

The body of the payload should follow the format below.

JSON Structure for Tool Delete

```
{
  "id": 0
}
```

See Table 6 for information on which fields are optional, the appropriate data types and the mappings to fields in Eyelit MES-M.

Table 6 – Parameter Information for Tool – Delete

Parameter Name	Data Mapping	Data Type	Mandatory	Case Sensitive	Match Type
id	TOOL.ID	Integer	No	N/A	Exact

Note: For any fields where the match type is 'Like', a percent symbol should be used as a wildcard character to indicate a number of characters within the given string.

Sample Request

See below for sample use cases with examples of the JSON payload format required.

Delete by ID:

```
{
  "id": 12345
}
```

Response

When using the Tool – Delete API call, if a tool has been deleted a JSON payload will be returned containing data in the following structure:

```
{

}
```