# LilxAPI

LilxAPI is a simpler, flat version of xAPI. Each statement is required to have only three properties: actor, verb, and object. Each property is limited to a single value. Statements may be formed either as a JSON object with name-value pairs or as an array.

LilxAPI (or Lil' xAPI) is a simpler, flat version of an [xAPI](https://edtechbooks.org/elearning_hacker/xapi) statement that allows it to be easily stored in a finite data structure, such as a spreadsheet or single table of a SQL database. In this simplified approach, each statement is required to have only three properties: actor, verb, and object. Each property is limited to a single value. Statements may be represented either as a JSON object with name-value pairs or as an array (using the array keys provided in Table 1).

Table 1

Overview of the LilxAPI structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| name | description | value format | default | example | array key |
| actor | The identity of the actor. | string | required | "janedoe@fake.com, Jane Doe""janedoe@fake.com""Jane Doe" or "janedoe1234" or "1234" | 0 |
| verb | The action performed by the actor. | human-readable term | required | "completed", "clicked", "interacted", "viewed", "answered", etc. | 1 |
| object | The target of the statement or what the actor interacted with. | url | required | "https://edtechbooks.org/education\_research/research" | 2 |
| result | The outcome of the action. | floatbooleanstring | null | .75true"success" | 3 |
| context | Contextual information about the action. | string | null | "web""app" | 4 |
| language\_id | The language of the schema. | ISO 639-1 Format | en | "en" or "fr" or "sp" | 5 |
| timestamp | The time of the interaction. | ISO 8601 format | current time | "2023-11-10T12:34:56Z" | 6 |
| xapi | The fully rendered xAPI statement. | json | null | - | 7 |

## Example Statements

### An Actor Viewed a Page

#### Object format

{
  "actor": "janedoe@fake.com",
  "verb": "viewed",
  "object": "https://edtechbooks.org/education\_research/research"
}

#### Array format

["janedoe@fake.com", "viewed", "https://edtechbooks.org/education\_research/research"]

### An Actor Correctly Answered a Question

#### Object Format

{
  "actor": "janedoe@fake.com",
  "verb": "viewed",
  "object": "https://edtechbooks.org/education\_research/research#question1",  "result": true
}

#### Array Format

["janedoe@fake.com", "answered", "https://edtechbooks.org/education\_research/research#question1", true]

## EdTech Books as LRS Example

### Storing LilxAPI Statements

### Retrieving LilxAPI Statements

You can retrieve all LilxAPI statements associated with your API key at this endpoint:

https://edtechbooks.org/api.v2.php?action=lilxapi\_get&api\_key=YOURAPIKEY

You can find your ETB API key by logging in and going to Account > Settings. You may then access the ETB LRS to access and download stored statements by going to Account > Developer.



## Google Sheets Automation

You can create dashboards or graphs in Google Sheets to display data and import the data directly from a website using the built-in ImportData function. Sheets does not currently have an import feature for JSON, but you can append &format=csv to a LilxAPI call to receive the results as a csv. Here is an example endpoint:

https://edtechbooks.org/api.v2.php?action=lilxapi\_get&format=csv&api\_key=YOURAPIKEY

## Excel Automation

You can create dashboards and graphs in Excel to display data and use an automation script to retrieve or update data directly from the LRS. Here is an example automation script using the ETB LRS:

const apiKey = "YOURAPIKEYHERE";

interfaceExcelRow {

statement\_id:number;

author\_id:number;

actor:string;

verb:string;

object:string;

result:string;

context:string;

language\_id:string;

timestamp:string;

xapi:string

}

asyncfunctionmain(workbook:ExcelScript.Workbook) {

constdata=awaitfetchData();

constnewWorksheet=workbook.getActiveWorksheet();

newWorksheet.activate();

constheadingRange=newWorksheet.getRangeByIndexes(

0,

0,

1,

Object.keys(json[0]).length);

headingRange.setValues([Object.keys(json[0])]);

constdataRange=newWorksheet.getRangeByIndexes(

1,

0,

json.length,

Object.keys(json[0]).length);

dataRange.setValues(json.map(row=>Object.values(row)));

}

asyncfunctionfetchData():Promise<string[][]> {

constapiUrl="https://edtechbooks.org/api.v2.php?action=lilxapi\_get&api\_key="+apiKey;

constresponse=awaitfetch(apiUrl);

if (!response.ok) {

thrownewError(`Failed to fetch data. Status: ${response.status}`);

 }

constjsonData:ExcelRow[] =awaitresponse.json();

returnjsonData;

}

Read this online at <https://edtechbooks.org/userguide/lilxapi>