LilxAPI

xAPI LilxAPI ISO 8601 Format JSON Object SQL Database Array Keys Finite Data Structure

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 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.	float boolean string	null	.75 true "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	ISO 8601	current	"2023-11-10T12:34:56Z"	6

name	description	value format	default	example	array key
	interaction.	format	time		
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

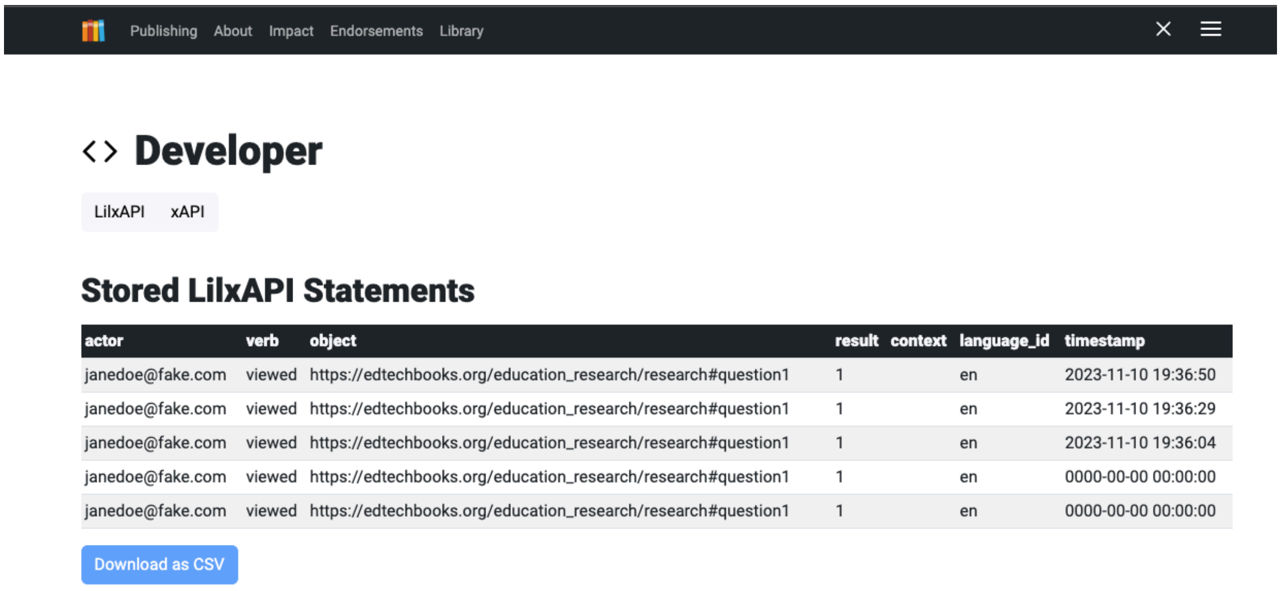
Storing LilxAPI Statements

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`.



Developer

Stored LilxAPI Statements

actor	verb	object	result	context	language_id	timestamp
janedoe@fake.com	viewed	https://edtechbooks.org/education_research/research#question1	1		en	2023-11-10 19:36:50
janedoe@fake.com	viewed	https://edtechbooks.org/education_research/research#question1	1		en	2023-11-10 19:36:29
janedoe@fake.com	viewed	https://edtechbooks.org/education_research/research#question1	1		en	2023-11-10 19:36:04
janedoe@fake.com	viewed	https://edtechbooks.org/education_research/research#question1	1		en	0000-00-00 00:00:00
janedoe@fake.com	viewed	https://edtechbooks.org/education_research/research#question1	1		en	0000-00-00 00:00:00

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";
interface ExcelRow {
statement_id: number;
author_id: number;
actor: string;
verb: string;
object: string;
result: string;
context: string;
language_id: string;
timestamp: string;
xapi: string
}
async function main(workbook: ExcelScript.Workbook) {
const data = await fetchData();
const newWorksheet = workbook.getActiveWorksheet();
newWorksheet.activate();
const headingRange = newWorksheet.getRangeByIndexes(
0,
0,
1,
Object.keys(json[0]).length);
headingRange.setValues([Object.keys(json[0])]);
const dataRange = newWorksheet.getRangeByIndexes(
1,
0,
json.length,
Object.keys(json[0]).length);
dataRange.setValues(json.map(row => Object.values(row)));
}
async function fetchData(): Promise<string[][]> {
const apiUrl = "https://edtechbooks.org/api.v2.php?action=lilxapi_get&api_key=" + apiKey;
const response = await fetch(apiUrl);
if (!response.ok) {
throw new Error(`Failed to fetch data. Status: ${response.status}`);
}
const jsonData: ExcelRow[] = await response.json();
return jsonData;
}
```