# Kippy Microsoft 365 Integration POC

Nauman Khan

25-Mar-2023

## Intro

This paper provides details of a proof of concept that integrated kippy with Microsoft Dynamics 365. The outcome successfully showed how kippy users could dynamically interface with at least a subset of the Microsoft Dynamics 365. More details are provided below.

## Background

### What is Microsoft Dynamics 365?

Microsoft Dynamics 365 is a cloud-based suite of enterprise resource planning (ERP) and customer relationship management (CRM) applications. It combines various business applications and services into a single platform, allowing organizations to manage their operations, customer interactions, and data more efficiently.

The suite includes a range of modules and functionalities that can be customized and tailored to meet specific business needs. Some key components of Microsoft Dynamics 365 include:

1. Dynamics 365 Sales: Helps organizations manage their sales processes, track leads and opportunities, automate sales workflows, and gain insights into customer interactions.

2. Dynamics 365 Customer Service: Provides tools for managing customer service operations, including case management, knowledge base, service level agreements (SLAs), and customer self-service portals.

3. Dynamics 365 Field Service: Enables organizations to efficiently manage field service operations, including scheduling and dispatching technicians, tracking work orders, and providing mobile access to field service personnel.

4. Dynamics 365 Marketing: Allows organizations to plan, execute, and analyze marketing campaigns, track customer engagement, and generate leads.

5. Dynamics 365 Finance: Provides financial management capabilities, including general ledger, accounts payable and receivable, budgeting, cash flow management, and financial reporting.

6. Dynamics 365 Supply Chain Management: Helps organizations optimize their supply chain operations, including inventory management, order fulfillment, demand forecasting, and vendor management.

7. Dynamics 365 Human Resources: Offers tools for managing HR processes, including employee onboarding, performance management, benefits administration, and workforce analytics.

8. Dynamics 365 Business Central: A comprehensive ERP solution for small and medium-sized businesses, providing finance, sales, purchasing, inventory, and project management capabilities.

Microsoft Dynamics 365 integrates with other Microsoft products such as Office 365, Power Platform, and Azure, providing a seamless ecosystem for organizations to manage their business processes, customer relationships, and data. It offers flexibility, scalability, and the ability to customize and extend functionalities through built-in tools and APIs.

## How to integrate with Microsoft Dynamics 365?

Microsoft Dynamics 365 provides APIs (Application Programming Interfaces) that allow developers to interact with and extend the functionality of the various modules and components within the platform. These APIs enable integration with external systems, the development of custom applications, and the automation of business processes.

Microsoft provides a set of RESTful APIs known as the Dynamics 365 Web API, which allows developers to interact with entities and data within Dynamics 365. The Web API supports CRUD operations (Create, Read, Update, Delete), querying data, executing actions, and more. It follows the OData protocol and can be accessed using standard HTTP requests.

By leveraging the APIs provided by Microsoft Dynamics 365, developers can integrate the platform with other systems, build custom applications that interact with Dynamics 365 data, automate business processes, and extract valuable insights from the platform.

## Which protocols and standards do those KPIs follow?

The APIs provided by Microsoft Dynamics 365 follow industry-standard protocols and standards to ensure compatibility, interoperability, and security. Here are some key protocols and standards commonly used by Microsoft Dynamics 365 APIs:

1. REST (Representational State Transfer): The Dynamics 365 APIs are built using RESTful principles. REST is a widely adopted architectural style for designing networked applications, leveraging standard HTTP methods like GET, POST, PUT, and DELETE to perform operations on resources.

2. OData (Open Data Protocol): The Dynamics 365 Web API conforms to the OData protocol, which is an open standard for building and consuming RESTful APIs. OData provides a consistent way to query and manipulate data, making it easier to integrate and interact with Dynamics 365 using standard HTTP requests.

3. OAuth 2.0: OAuth 2.0 is an industry-standard authorization framework used for secure authentication and authorization. Dynamics 365 APIs utilize OAuth 2.0 for authentication, allowing developers to obtain access tokens to authenticate and authorize API requests on behalf of users or applications.

4. JSON (JavaScript Object Notation): JSON is a lightweight data interchange format widely used by Dynamics 365 APIs. Data is typically exchanged in JSON format, making it easy to parse, manipulate, and transmit data between client applications and the Dynamics 365 platform.

5. HTTPS (Hypertext Transfer Protocol Secure): All communication with Dynamics 365 APIs occurs over HTTPS, which adds a layer of encryption and ensures secure transmission of data between the client and the server.

These protocols and standards provide a foundation for building robust, scalable, and secure integrations with Microsoft Dynamics 365. They facilitate interoperability with other systems, simplify development and consumption of APIs, and adhere to widely accepted best practices in the industry.

## How does kippy integrate with other systems?

Kippy provides many approaches to integrate with other systems [see https://www.kippy.cloud/integrate for more details]. One of those approaches include using adapters that connect to Restful JSON APIs via Oauth2.0 over HTTPS.

Kippy has many existing adapters to connect to external systems using those protocols and standards. There are a few options for these adapters in terms of configurability, flexibility and hosting.

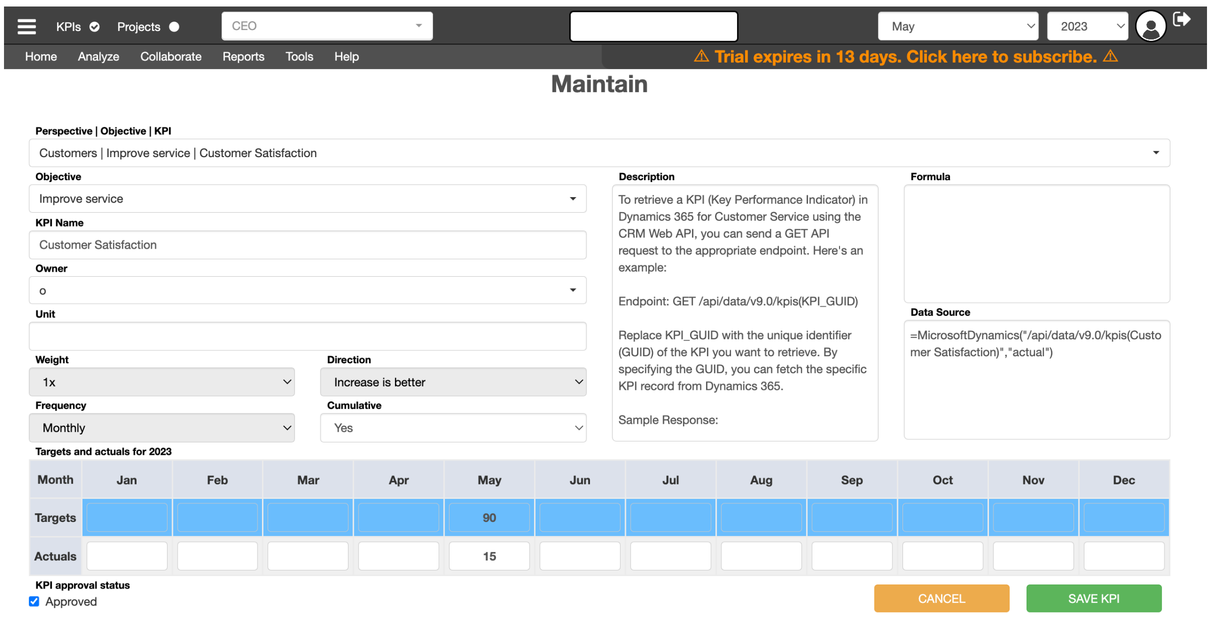
For example, existing generic adapters can be used and leave the config to the end user. Alternatively, existing adapters can be modified to abstract some of that complexity away from end users.

These adapters can be code by the kippy team or coded by the client. The code can then be hosted and run on the kippy infrastructure or hosted on the customer’s infrastructure.

The good news is that integration with standard-based APIs is extremely easy.

## How did kippy integrate with Micrsoft Dynamics 365 during the proof of concept?

A simple data-driven adapter was coded and deployed on kippy which would look for all kippy KPIs with a datasource of MicrosoftDynamics, and use the information entered by the end user to pull the appropriate information via an API call as described above.



## Maintain

Perspective | Objective | KPI: Customers | Improve service | Customer Satisfaction

Objective: Improve service

KPI Name: Customer Satisfaction

Owner: o

Unit:

Weight: 1x

Direction: Increase is better

Frequency: Monthly

Cumulative: Yes

Description: To retrieve a KPI (Key Performance Indicator) in Dynamics 365 for Customer Service using the CRM Web API, you can send a GET API request to the appropriate endpoint. Here's an example:

Endpoint: GET /api/data/v9.0/kpis(KPI_GUID)

Replace KPI_GUID with the unique identifier (GUID) of the KPI you want to retrieve. By specifying the GUID, you can fetch the specific KPI record from Dynamics 365.

Sample Response:

Formula:

Data Source: =MicrosoftDynamics("/api/data/v9.0/kpis(Customer Satisfaction)","actual")

Targets and actuals for 2023

| Month | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug | Sep | Oct | Nov | Dec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Targets | | | | | 90 | | | | | | | |
| Actuals | | | | | 15 | | | | | | | |

KPI approval status

☑ Approved

In this example, a call to a Microsoft Dynamics 365 API was made using the following data source entered by the kippy end user.

*=MicrosoftDynamics("/api/data/v9.0/kpis(Customer Satisfaction)","actual")*

This specifies that:

1) The MicrosoftDynamics adapter should be used to get the value for this kippy kpi – Customer Satisfaction.
2) The MicrosoftDynamics API to make a remote call to is */api/data/v9.0/kpis*
3) The configuration for which MicrosoftDynamics instance to call (i.e. it’s url and access credentials) are abstracted away from the end user.
4) The name of the MicrosoftDynamics kpi also (happens to be) *Customer Satisfaction*
5) The value in that response payload can be found in the json element named *actual*

FYI Below is a snippet from the documentation for the */api/data/v9.0/kpis* MicrosoftDynamics API

To retrieve a KPI (Key Performance Indicator) in Dynamics 365 for Customer Service using the CRM Web API, you can send a GET API request to the appropriate endpoint. Here's an example:

Endpoint: GET /api/data/v9.0/kpis(KPI_GUID)

Replace KPI_GUID with the unique identifier (GUID) of the KPI you want to retrieve. By specifying the GUID, you can fetch the specific KPI record from Dynamics 365.

Sample Response:

```
{
  "@odata.context": "$metadata#kpis/$entity",
  "kpiid": "KPI_GUID",
  "kpiname": "Customer Satisfaction",
  "target": 90,
  "actual": 87,
  "trend": "Improving",
  "parentkpiid@odata.bind": "/kpis(Parent_KPI_GUID)"
}
```

## What did the proof of concept prove?

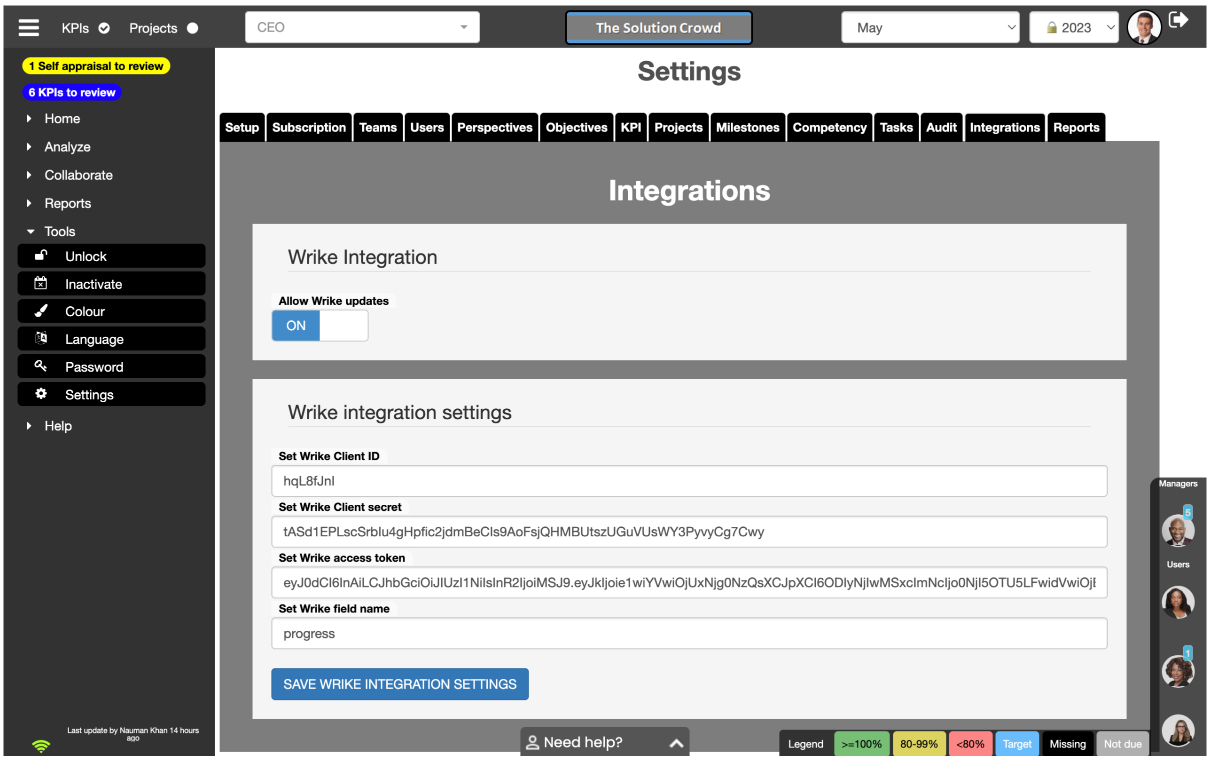
The proof of concept proved the following.

1) A kippy end user could simply specify which Microsoft Dynamics API and value should be used to populate the KPI actual.
2) Kippy would invoke a url to get that value and persist it into kippy against the actual for that kippy KPI.

## What was out of scope of the proof of concept?

The proof of concept did not look to prove the following:

1) Oauth2.0 integration with a deployed Microsoft Dynamics instance using credentials managed by a kippy end user admin was out of scope. The rationale for this is because this has already been proved in the Wrike demo. Also, a Microsoft Dynamics environment is not available so instead a stub for the same API spec was created.



Settings

Integrations

Wrike Integration

Allow Wrike updates: ON

Wrike integration settings

Set Wrike Client ID: hqL8fJnl

Set Wrike Client secret: tASd1EPLscSrblu4gHpfic2jdmBeCls9AoFsjQHMBUtszUGuVUsWY3PyvyCg7Cwy

Set Wrike access token: eyJ0dCI6InAiLCJhbGciOiJIUzI1NiIsInR2IjoiMSJ9.eyJkljoie1wiYVwiOjUxNjg0NzQsXCJpXCI6ODIyNjIwMSxclmNcljo0NjI5OTU5LFwidVwiOjI

Set Wrike field name: progress

SAVE WRIKE INTEGRATION SETTINGS

2) Invoking different APIs on other Microsoft Dynamics 365 modules was out of scope of this proof of concept. The rational for this was that the other APIs expose the data in exactly the same way (i.e. RESTful JSON APIs using Oauth2.0 over HTTPs). Therefore, the same adapter could be invoked with different data source parameters by the end user. The two parameters being a) the api url and b) the json element to get the value from.
3) Retrieving values from nested elements in the APIs as that was already proven in a previous demo.

## How did the proof of concept work technically?

A new adapter was created by the kippy team and deployed on the kippy infrastructure. This adapter would find any kippy KPIs with a data and call the Microsoft Dynamics API.

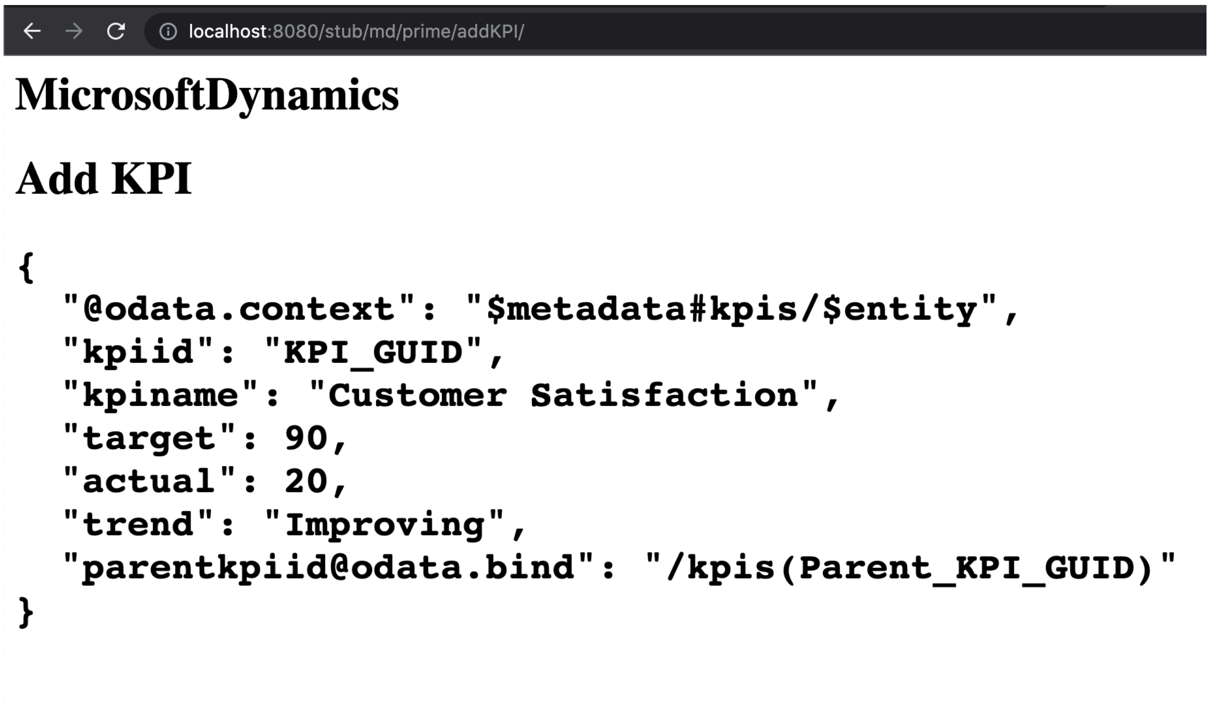
```java
public class MicrosoftDynamicsBean extends AbstractAPIBean {

    private static final String ADAPTER_ID = "=MicrosoftDynamics(";

    public void init() throws Exception{
        sanityCheck();
        setup();
        Pattern pattern = Pattern.compile("\"([^\"]*)\"");
        String json = getInfo(new URL(getUrl()));
        for (Principle principle : getAllPrinciplesFromAllBoards().values()) {
            PrincipleAttributes pa = getAllPrincipleAttributesFromAllBoards().get(principle.getIdentifier());
            if (pa!=null) {
                String dataSource = pa.getDataSource();
                if (dataSource != null && dataSource.startsWith(ADAPTER_ID)) {
                    Matcher matcher = pattern.matcher(dataSource);
                    matcher.find(); //ignore view name for PoC
                    String report = matcher.group(1);
                    matcher.find(); //get json path from 2nd data source parameter
                    String jsonPath = matcher.group(1);
                    JsonElement value = traverseJsonHierarchy(new Gson().fromJson(json, JsonObject.class), jsonPath);
                    String boardName = getBoardCache().getBoards().get(principle.getBoardIdentifier()).getName();
                    updateKippyKPI(boardName, principle.getName(), value.getAsString());
                    message = message + ( "Latest value " + value.getAsString() + " pulled from " + report + " " + jsonPath
                }
            }
        }
    }
```

This adapter could be invoked on a schedule or by Microsoft Dynamics itself to 'pull on demand'.

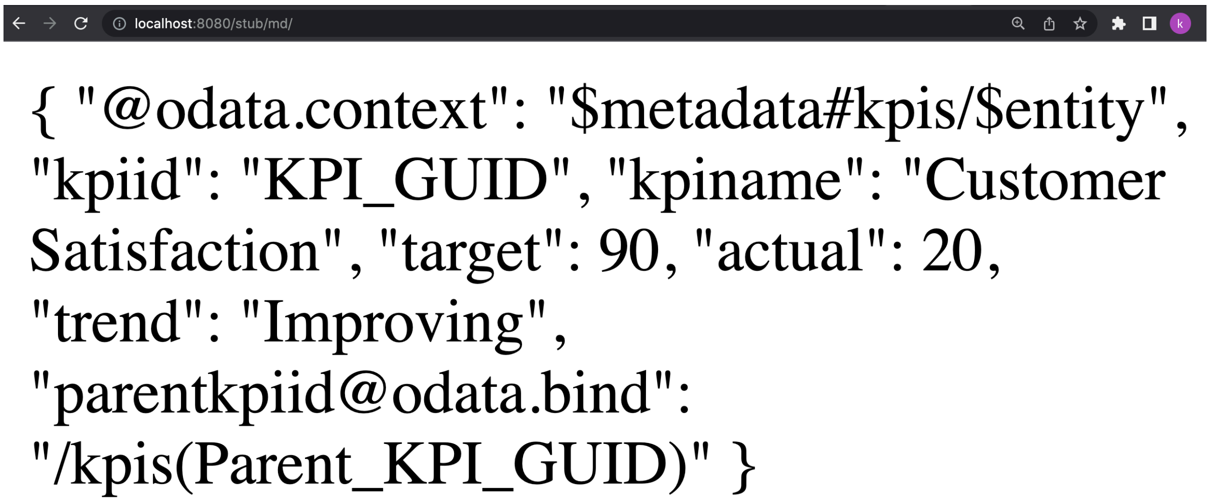
Below are screen shots of the primer, stub, scheduler output and updated KPI.


localhost:8080/stub/md/prime/addKPI/

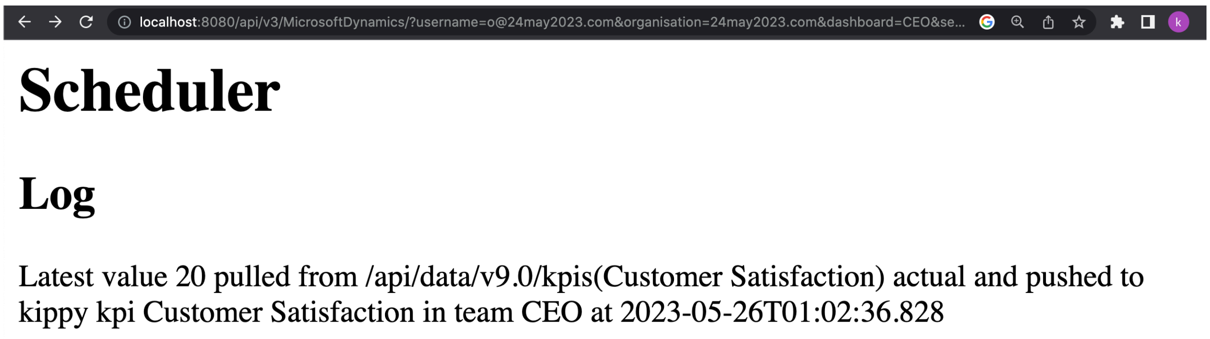
MicrosoftDynamics

Add KPI

```
{
  "@odata.context": "$metadata#kpis/$entity",
  "kpiid": "KPI_GUID",
  "kpiname": "Customer Satisfaction",
  "target": 90,
  "actual": 20,
  "trend": "Improving",
  "parentkpiid@odata.bind": "/kpis(Parent_KPI_GUID)"
}
```



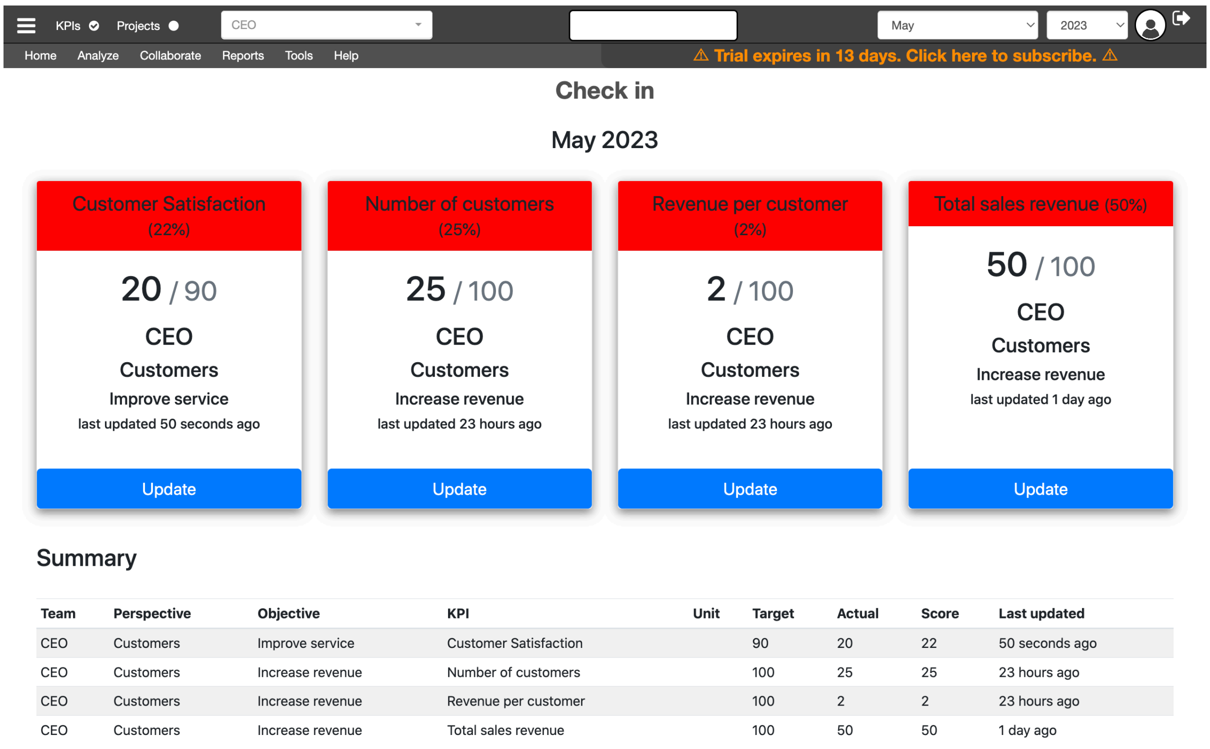
```
{ "@odata.context": "$metadata#kpis/$entity",
"kpiid": "KPI_GUID", "kpiname": "Customer
Satisfaction", "target": 90, "actual": 20,
"trend": "Improving",
"parentkpiid@odata.bind":
"/kpis(Parent_KPI_GUID)" }
```


localhost:8080/api/v3/MicrosoftDynamics/?username=o@24may2023.com&organisation=24may2023.com&dashboard=CEO&se...

## Scheduler

### Log

Latest value 20 pulled from /api/data/v9.0/kpis(Customer Satisfaction) actual and pushed to kippy kpi Customer Satisfaction in team CEO at 2023-05-26T01:02:36.828



Check in

May 2023

Customer Satisfaction (22%)
20 / 90
CEO
Customers
Improve service
last updated 50 seconds ago
Update

Number of customers (25%)
25 / 100
CEO
Customers
Increase revenue
last updated 23 hours ago
Update

Revenue per customer (2%)
2 / 100
CEO
Customers
Increase revenue
last updated 23 hours ago
Update

Total sales revenue (50%)
50 / 100
CEO
Customers
Increase revenue
last updated 1 day ago
Update

Summary

| Team | Perspective | Objective | KPI | Unit | Target | Actual | Score | Last updated |
|---|---|---|---|---|---|---|---|---|
| CEO | Customers | Improve service | Customer Satisfaction | | 90 | 20 | 22 | 50 seconds ago |
| CEO | Customers | Increase revenue | Number of customers | | 100 | 25 | 25 | 23 hours ago |
| CEO | Customers | Increase revenue | Revenue per customer | | 100 | 2 | 2 | 23 hours ago |
| CEO | Customers | Increase revenue | Total sales revenue | | 100 | 50 | 50 | 1 day ago |