Office 365 Timesheet

Extensibility Guide for Partners

Contents

[1 Overview 1](#_Toc518053656)

[1.1 Add more categories 1](#_Toc518053657)

[1.2 Connecting from existing UI 2](#_Toc518053658)

# Overview

Timesheet is designed to be easily extensible to support potential new requirements for partners and customers across multiple industry verticals and regions.

## Add more categories

One key area of extensibility is support for additional categories. At this time, the solution supports three different categories of work hours:

* Email
* Meeting
* Other (currently used to enable user to account for any additional time under other categories)

This can be extended to include other categories such as activity in Teams or Yammer, Skype Meetings, Office application usage and Login activity information.

Adding support for a new category involves changes in two areas:

* Compute Logic

Computation logic for calculating work hours for a user will have to be updated to include the additional category, which may necessitate adding new settings in the application configuration to customize as needed for each deployment instance.

* User Interface

The three supported categories (Email, Meeting, Other) are clearly differentiated in the Edit Hours interface to provide further insights into the total hours at a day level. When adding a new category, one should consider adding the information as a separate item here or in other areas such as in the Dashboard, or alternately add them to the Other category from a visualization standpoint

As an example, let us consider the case where a partner wants to add a new service, say, Yammer, to the work hours compute logic.

Note: Yammer APIs do not currently support app context

To accomplish this, the first step is to update Settings files (Appsettings.json in the Dashboard and host.json in Azure Functions) to add a new property for time calculation configuration for Yammer activity. Following this, one will need to update the following files in both middle tier and Azure Functions:

**SharePointListsSchemaHelper.cs**

Add the following columns to the GetWorkHoursJsonSchema method:

* YammerHours
* YammerMinutes
* YammerAdjustedHours
* YammerAdjustedMinutes

**GraphAppYammerService.cs**

Add new file under /services with all the logic needed to get the Yammer data using Graph API (once it supports application context)

**WorkHoursRepository.cs**

Add the logic needed in the ComputeHours, ComputeHoursforWeek and ConvertToWorkHours methods to calculate and update the SharePoint list with the yammer values using the configuration in the app settings.

**WorkHours.cs**

Add the Yammer fields added to the SharePoint list to the WorkHoursFields class

## Connecting from existing UI

A partner can integrate an existing application with Timesheet by directly connecting to the interfaces in the middle tier which will help fetch the data using Graph API, persist the data in SharePoint and access them in another user interface or use the data for additional calculations and reporting. The compute logic will also need to be updated in associated Azure functions.

To proceed with this, a developer will need to refer to the following files:

* GraphSharepointService.cs to query work hours data
* WorkHoursRepository.cs to calculate and update data in the user work hours list in SharePoint