System Design  
Shifts-Kronos Integration

Contents

[Overview 3](#_Toc85461655)

[Considerations 3](#_Toc85461656)

[Partners and Dependencies 4](#_Toc85461657)

[Solution overview 4](#_Toc85461658)

[Shifts Kronos Integration Service Plan 6](#_Toc85461659)

[Configuration Web App 6](#_Toc85461660)

[Integration Service API 10](#_Toc85461661)

[Integration API Design Considerations 10](#_Toc85461662)

[Microsoft Graph APIs 17](#_Toc85461663)

[Microsoft Graph APIs for Approval/Decline Request workflow for Open Shift and Swap Shift Requests 22](#_Toc85461664)

[Open Shift Request Approval 23](#_Toc85461665)

[Open Shift Request Decline 25](#_Toc85461666)

[Approve Swap Shift Request 26](#_Toc85461667)

[Decline Swap Shift Request 28](#_Toc85461668)

[Kronos WFC APIs Description 29](#_Toc85461669)

[Data Store 34](#_Toc85461670)

[ConfigurationInfo 34](#_Toc85461671)

[User to User Mapping 35](#_Toc85461672)

[Team to Department Mapping with Job Mapping 35](#_Toc85461673)

[Paycode to TimeOffReasons Mapping 36](#_Toc85461674)

[ShiftEntityMapping 36](#_Toc85461675)

[Time Off Mapping 37](#_Toc85461676)

[Open Shift Entity Mapping 37](#_Toc85461677)

[Open Shift Request Mapping 38](#_Toc85461678)

[Swap Shift Request Mapping 39](#_Toc85461679)

[Time Off Reason – PayCode Configuration 40](#_Toc85461680)

[Accessing the Kronos WFC PayCodes via API 41](#_Toc85461681)

[Inserting a Time Off Reason into Shifts App 42](#_Toc85461682)

[Terminated and Inactive Users 43](#_Toc85461683)

[Application Insights 43](#_Toc85461684)

# Overview

This document explains the design of Shifts-Kronos Integration solution, hosted in Microsoft Azure environment.

**Kronos Workforce Central (Kronos WFC 8.1, or Kronos WFC) is a Workforce Management system designed for First Line Managers (FLMs) and First Line Workers (FLWs). Kronos provides various capabilities to handle schedules of FLWs in organizations with multiple departments and job categories. First Line Workers can access their schedule, create schedule requests for Time Offs, Open Shifts, Swap Shifts etc. FLMs can create, and access their FLWs' schedules, schedule requests and approve those.**

**Shifts App in Microsoft Teams keeps FLWs connected and in sync. It's built mobile first for fast and effective time management and communication for teams. Shifts App lets FLWs and FLMs use their mobile devices to manage schedules.**

**Shifts-Kronos Integration application is built to sync data between Kronos Workforce Central (v8.1) and Microsoft Shifts App in Teams in seamless fashion. It helps FLWs access their schedules in Shifts App, which are originally created in Kronos system. And further it enables FLMs to access requests from Kronos system which are originally submitted in Shifts by FLWs.**

# Considerations

The following considerations have been accounted for while designing the Shifts-Kronos Integration:

* IT Administrators have a functional understanding of Kronos WFC 8.1 and Microsoft Teams Shifts App. IT Administrators are also the SuperUsers of their on-premises instance of Kronos WFC 8.1.
* Kronos WFC serves as the single source of truth for all entities
  + Shifts App can be used by FLWs to view their schedules, create requests for Time Offs, Open Shifts, and Swap Shifts
  + **The Offer Shift entity and related functionalities are not in scope of this integration**
* FLW requests (Open Shift Request, Swap Shift Request and time off request) will be synced from Shifts to Kronos in synchronous manner using Shifts Outbound APIs and Kronos WFC 8.1 data submission (POST) APIs
* Approved schedules for Shifts, Time Offs, Open Shifts and Swap Shifts will be sync’d from Kronos to Shifts App in asynchronous manner using Kronos WFC 8.1 GET APIs and Shifts/Graph POST APIs
* Status of requests created in Shifts App and synced to Kronos WFC will be synced back to Shifts App to keep both systems in sync
* Approved schedules for Shifts, Time Offs, Open Shifts and Swap Shifts will be synced from Shifts to Kronos in a synchronous manner using outbound API’s
* To sync all the requests initiated in Shifts App (by FLWs) to Kronos, SuperUser account credentials are used.
* Users must be created in Azure/Teams prior to User-to-User mapping step to be performed in Configuration Web App (Configuration Web App is one of the components of this integration as explained in below sections)
* Teams and Scheduling groups must be created in Shifts App prior Teams to Department mapping step in Configuration Web App
* CommentText values must be added to Kronos prior to deployment, allowing us to sync notes
* Done button on Configuration Web App should be used only for first time sync
* First time sync is expected to take longer time since it may sync data for larger time interval. The time would vary based on amount of data i.e. number of users, number of teams, number of entities (such as Shifts, Time Offs, Open Shifts etc.) to be synced and date span of the Time interval for which the sync is happening. So, it may take time to reflect this complete data in Shifts. Done button click will initiate background process to complete the sync

# Partners and Dependencies

The dependencies for this project are:

1. Microsoft Azure – for hosting and deployment of Sync Job, Integration Service API, and Configuration Web App
2. Microsoft Graph APIs – for posting data into Shifts
3. Kronos WFC APIs – for data retrieval and data submission

# Solution overview

The Shifts-Kronos Integration has the following components built using ASP.Net core 2.2 and hosted on Microsoft Azure:

* Configuration Web App
* Integration Service API
* Azure Logic App for periodic data sync
* Kronos WFC Solution library created to retrieve data and post data to Kronos



1. Hosting Plan – for hosting environments for both the Integration Service API and the Configuration Web App, defines details such as region and pricing tier
2. Web App Services – For Configuration Web App and the Integration Service API. The Configuration Web App and the Integration Service API are both written in ASP.NET Core technologies
3. Application Insights – capture necessary telemetry at the time of necessary events, and will be used by the Configuration Web App and the Integration Service API
4. Azure Table Storage – the database account which contains the necessary tables required for the entire Shifts-Kronos Integration to work successfully
5. Azure Logic App – this is the schedule job that will sync data between Kronos WFC and Shifts on a configured interval of time
6. Azure Key Vault – to store all the connection strings, client Ids, client secrets
7. Azure Redis Cache – Stores access token for accessing graph API as well as the Kronos login token.
8. Kronos WFC Solution library – This is custom library project which part of Integration Service API. It will be used to query and submit data to Kronos WFC.

Tenant Admin is responsible for

1. Deploying Shifts-Kronos Integration in target environment
2. Performing necessary Configuration Steps
3. Establishing the trigger for the Azure Logic app

### Shifts Kronos Integration Service Plan

App service plan includes two key components:

1. Configuration Web App: to be used by Tenant Admin to configure the Integration App settings
2. Integration Service API:
   1. Asynchronous data transfer of various schedule entities from Kronos to Shifts. This data includes: Shifts, Time Offs, Open Shifts, Acceptance Statuses of Time Off requests, Open Shifts requests, and Swap Shift requests
   2. Synchronous data transfer from Shifts to Kronos. This data includes Open Shift requests, Swap Shift requests, time off requests and their approvals as well as the creation of open shifts by a manager and the creation, update and deletion of shifts submitted in Shifts App

# Configuration Web App

The Configuration Web App serves as a helpful aid to establish the necessary configurations to properly integrate an instance of Kronos WFC v8.1 with Shifts App.

After successful login, Tenant Admin shall use Configuration Web App to perform following actions:

1. Capture Kronos WFC endpoint, SuperUser name and password, and store in Azure Key Vault (done on Submit button click of Configuration Web App homepage, please see Figure1 below)
2. Workforce Integration Registration:Map configuration entity mapped to a workforce integration registration (stored in Azure storage, please see Figure2 below)
3. Map Users in Kronos to Shifts users (AAD users) (stored in Azure storage, please see Figure3 which explains the process of Exporting users from Kronos and Shifts; and Figure4 below which explains User to User mapping via Import action)
4. Map Kronos Departments to Teams in Shifts (stored in Azure storage, this mapping can be achieved in similar fashion as User to User mapping)

Please refer to the figures below which explain workflows such as saving Kronos credentials, registering a Workforce integration, and mapping (Export and Import workflows) users between Kronos and Azure AD (Shifts App users),



Figure: Saving Kronos credentials



Figure: Workforce Integration registration



Figure: Export Users



Figure: Import Users to Users mapping

# Integration Service API

Integration Service API includes following endpoints which are responsible for transferring various entities between Kronos WFC and Shifts App.

|  |  |
| --- | --- |
| **Endpoint (Controller) Name** | **Actions** |
| ConnectController | Establishes connection with Microsoft Graph |
| OpenShiftController | Transfers the Open Shift entities between Kronos WFC to Shifts App |
| OpenShiftRequestController | Transfers the Open Shift Requests, and approved/declined Open Shift Requests between Kronos WFC and Shifts App |
| ShiftController | Transfers the Shift entities between Kronos WFC to Shifts App |
| SwapShiftController | Transfers the Swap Shift Requests, and approved/declined Swap Shift Requests between Kronos WFC and Shifts App |
| SwapShiftEligibilityController | Contains the necessary logic for swap shift filtering capabilities |
| SyncKronosToShiftsController | Initiates the sync process from Kronos WFC to Shifts App |
| TeamsController | Listens to real time outbound requests made from Shifts App |
| TimeOffController | Transfers the approved Time Off entities from Kronos WFC to Shifts App  Transfers the Time Off Requests and approved/declined Time Off Requests between Kronos WFC and Shifts App |
| TimeOffReasonController | Maps Kronos WFC paycodes to Time Off Reasons in Shifts App |
| UserController | Checks for any users that are no longer active in Kronos and excludes them from processing |

The *TeamsController* endpoint listens to outbound calls from Shifts App and accordingly invokes endpoints such as *OpenShiftRequestController* or *SwapShiftController* depending on the JSON payload received to send respective requests data to Kronos.

*SyncKronosToShiftsController* endpoint is invoked through either the Azure Logic App or Done button initial sync, which internally calls endpoints such as *UserController*, *OpenShiftController*, *ShiftController*, *TimeOffReasonController* and *TimeOffController* to transfer the respective entities from Kronos to Shifts App in asynchronous manner. *SyncKronosToShiftsController* also invokes *OpenShiftRequestController* and *SwapShiftController* endpoints to transfer approved or declined requests from Kronos WFC to Shifts App.

## Integration API Design Considerations

* For Shift, Open Shift entities in Kronos, there are no unique identifiers available from Kronos APIs (Unique identifiers is available for Time Off requests, Open Shift requests, and Swap Shift requests from Kronos)
* Delta (changes from a specified time) changes are not available from Kronos APIs nor any outbound calls available from Kronos, which implies comparing all the entities for given duration for all users to find the delta between two runs
* Lookup tables are maintained for each of entity sync (Shift, Open Shift, Time Off, Swap Shift) between Kronos and Shifts app to avoid duplication of data and achieve update/delete of already synchronized data
* Design considers large data size (entities for up to 100,000 employees’ sync) and longer duration for data sync (ex: one year) as part first time sync. Duration should be configurable from weeks to months. Initial data sync could be setup for longer duration (ex: 3 months in past and 1 month in future from current date) while usual duration chosen by admins could be up to one month (ex: It should sync data from past one week to three weeks ahead)
* Design considers that there is one instance of Shifts App to correspond with one instance of Kronos WFC
* As there are no unique identifiers returned by Kronos APIs for Shifts, OpenShift entities, we need to calculate delta based on specific fields for Employees. Below is detailed structure of Integration Service API to calculate delta and posts **Shift, Open Shift, Time Off sync from Kronos to Shifts**
* **Unique Identifier For Each Entity**: For Shift, as the unique identifier is not available, Hash value would be computed for Person Number, Start Date Time, End Date Time, Shift Activity Type, Shift Activity Start Date Time, Shift Activity End Date Time, and Notes for each Shift for each of users fetched within the batch. Similarly, for OpenShift, Hash value would be computed using Schedule Group Id, Start Date Time, End Date Time, Shift Activity Type, Shift Activity Start Date Time, Shift Activity End Date Time, Notes.

For Time Off, request id is available from Kronos APIs. It will be used as unique identifier

* For first time sync from Kronos to Shifts app, *Done* button in Configuration app will initiate the sync for different entities in BatchSize of users and predefined BatchDuration for Start Date to End Date specified by Admin. Admin will be updated once the sync is completed
* For delta sync, logic app will sync different entities in BatchSize of users and predefined BatchDuration for current month till End Date specified by admin
* **Mapped Users Fetch:** Integration Service will fetch all mapped users from User Mapping table in Azure Storage
* **Fetch Entity For Sync In Batch**: Out of all mapped users, Integration Service users batch of predefined number of users (BatchSize) to fetch data for given entity from Kronos APIs for a predefined duration (BatchDuration), so that it will not be blocked or fail due to API Rate Limit Exception, if any and also to ensure Kronos APIs do not timeout while fetching large data size. BatchSize and BatchDuration are app config variables, to be set based upon performance benchmarks / fetch limits
* **Lookup in Mapping table**: For each user in given BatchSize and month assuming BatchDuration=month, check if Hash value (for entity fetched from Kronos) or request id exists in Mapping table. This lookup would be done in memory by getting lookup data for users within Batch (Row Key of table) and month (PartitionKey of table). There would be mapping tables for each of entity, details are given in [Data Store](#_Data_stores) section
* For Shift, and Open Shift entities:
  + if count and value of Hash for user match with count and value of Hash for same user in mapping table then there is no update needed
  + if count and value of Hash for user do not match with count and value of Hash for same user in mapping table then create the following collections – KronosHashNotFoundInLookUp and OrphanEntryInLookUp. KronosHashNotFoundInLookUp represents either updates to existing shift or new creation. OrphanEntryInLookUp represents updates to existing shift or deletion.
  + Due to lack of unique identifier, time information for update to entities and ability to duplicate Shift entries in UI, it is not possible to find out which Shift entity is updated vs newly created vs deleted.
  + So, we are proposing to delete all Shifts in Shifts app corresponding to OrphanEntryInLookUp and Create new Shifts in Shifts app corresponding to KronosHashNotFoundInLookUp. This will ensure that Shifts will be in sync from Kronos to Shifts App. Similar, implementation for Open Shifts as well.
* For Time Off entities:
  + If RequestId for given user (for entity fetched from Kronos) exists in mapping table then existing details need to be replaced in Shifts app
  + If RequestId for given user (for entity fetched from Kronos) not found in mapping table then need to create in Shifts app.
  + If RequestId for given user from mapping table not matching with any of RequestId (for entity fetched from Kronos) then need to delete from Shifts app.
* For Time Off Request processing from Shifts to Kronos
  + Fetch
* For all delta changes (based upon above lookup), post to Shifts app.
  + For each of creation in Shift apps, a new mapping entry to be created in corresponding mapping table
* For each of deletion in Shift apps, relevant mapping entry to be deleted in corresponding mapping table



Figure: Shifts entity transfer from Kronos to Shifts App

* **Time Off request processing** 
  + Shifts to Kronos

When an Time Off Request is created in the Shifts App, transfer the request to Kronos. If successful, then create a mapping entry in the TimeOffMapping table with the KronosStatus as Submitted and return the appropriate response to the Shifts App. If failure, the Kronos Time Off Request results in an error, the response message is returned to the Shifts App.

The manager can approve and deny the time off request in Shifts and it is synced in the same way. If the request is approved in Kronos, we sync this on the next logic app triggered sync from Kronos.

* **Open Shift Request Processing**
  + **Shifts App to Kronos WFC**

When an Open Shift Request is created in the Shifts App, transfer the request to Kronos. If successful, then create a mapping entry in the OpenShiftRequestMapping table with the KronosStatus as Submitted and the ShiftsStatus as Pending, and return the appropriate response to the Shifts App. If failure, the Kronos Open Shift request results in an error, the response message is returned to the Shifts App.



Figure: Open Shift request from Shifts App to Kronos

* + Kronos to Shifts

Fetch all FLM approved/declined OpenShift requests from Kronos for a given user. Update KronosStatus in OpenShiftRequestMapping table accordingly. Invoke OpenShift Approve/Decline Graph API endpoint. Accordingly update ShiftsStatus in the OpenShiftRequestMapping table.



Figure: Open Shift approval from Kronos to Shifts App

* **Swap Shift Request Processing**
  + Shifts to Kronos
  + Step1 – FLW1 creates a Swap Shift request in Shifts App which triggers an outbound call. Integration endpoint API transfers it to Kronos as *Offered.* Update KronosStatus in SwapShiftRequestMapping table accordingly
  + Step2 - FLW2 approves that Swap Shift request in Shifts App which triggers another outbound call. Integration endpoint API transfers it to Kronos as *Submitted* (for FLM approval). Update KronosStatus in SwapShiftRequestMapping table accordingly



Figure: Swap Shift request from Shifts to Kronos

* + Kronos to Shifts

Fetch all FLM approved/declined Swap Shift requests from Kronos. For approvals, swap user Ids between two mapping entries in ShiftMapping table which correspond to SenderShiftId & RecipientShiftId. Invoke SwapShift Approve/Decline Graph API endpoint. Accordingly update ShiftsStatus in the SwapShiftRequestMapping table.



Figure: Swap Shift approval from Kronos to Shifts App

# Microsoft Graph APIs

All the Microsoft Graph APIs listed below use delegated permissions to fetch data from Graph. Below is the list of Microsoft Graph APIs being used in the Configuration Web App as well as the Integration Service API:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.no.** | **Use Case** | **API** | **Permissions** | **Module** |
| 1. 1 | Register Workforce Integrations | Request Type: POST  <https://graph.microsoft.com/beta/teamwork/workforceIntegrations>  *Request Body: {*  *"displayName": "KronosWFC2ShiftsIntegration",*  *"apiVersion": 1,*  *"isActive": true,*  *"encryption": {*  *"protocol": "sharedSecret",*  *"secret": "a1f3196a46f14f489c71da0c8e1bc7e2d3f60988453646bca794daca356573d9"*  *},*  *"url": "https://KronosWFC2ShiftsIntegration-azurewebsites.net/Contoso/",*  *"supports": "Shift, SwapRequest"*  *}* | WorkforceIntegration.ReadWrite.All | Configuration Web App |
|  | Getting Workforce Integrations at the time of team to department mapping | Request Type: GET  <https://graph.microsoft.com/beta/teamwork/workforceIntegrations> | WorkforceIntegration.ReadWrite.All | Configuration Web App |
|  | Getting Teams list in organization for team to department mapping | Request Type: GET  https://graph.microsoft.com/beta/groups?$filter=resourceProvisioningOptions/Any(x:x eq 'Team') | Group.ReadWrite.All, Directory.ReadWrite.All, Directory.AccessAsUser.All | Configuration Web App |
|  | Getting Users list in organization for user to user mapping | Request Type: GET  <https://graph.microsoft.com/beta/groups/8396e1f4-9e0e-42d9-96b3-b525992792b1/members> | User.ReadBasic.All, User.Read.All, Group.ReadWrite.All, Directory.ReadWrite.All | Configuration Web App |
|  | Create Shift | Request Type: POST  [https://graph.microsoft.com/beta/teams/{teamId}/schedule/shifts](https://graph.microsoft.com/beta/teams/%7bteamId%7d/schedule/shifts)  *Request Body:*  *{*  *"id": "SHFT\_577b75d2-a927-48c0-a5d1-dc984894e7b8",*  *"userId": "c5d0c76b-80c4-481c-be50-923cd8d680a1",*  *"schedulingGroupId": "TAG\_228940ed-ff84-4e25-b129-1b395cf78be0",*  *"sharedShift": {*  *"displayName": "Day shift",*  *"notes": "Please do inventory as part of your shift.",*  *"startDateTime": "2019-03-11T15:00:00Z",*  *"endDateTime": "2019-03-12T00:00:00Z",*  *"theme": "blue",*  *"activities": [*  *{*  *"isPaid": true,*  *"startDateTime": "2019-03-11T15:00:00Z",*  *"endDateTime": "2019-03-11T15:15:00Z",*  *"code": "",*  *"displayName": "Lunch"*  *}*  *]*  *},*  *"draftShift": {*  *"displayName": "Day shift",*  *"notes": "Please do inventory as part of your shift.",*  *"startDateTime": "2019-03-11T15:00:00Z",*  *"endDateTime": "2019-03-12T00:00:00Z",*  *"theme": "blue",*  *"activities": [*  *{*  *"isPaid": true,*  *"startDateTime": "2019-03-11T15:00:00Z",*  *"endDateTime": "2019-03-11T15:30:00Z",*  *"code": "",*  *"displayName": "Lunch"*  *}*  *]*  *}*  *}* | Group.ReadWrite.All | Integration Service API – Post to Shifts App |
|  | Delete Shift | Request Type: DELETE  [https://graph.microsoft.com/beta/teams/{teamId}/schedule/shifts/{shiftId}](https://graph.microsoft.com/beta/teams/%7bteamId%7d/schedule/shifts/%7bshiftId%7d) | Group.ReadWrite.All | Integration Service API – Post to Shifts App |
|  | Create TimeOffReason | Request Type: POST  [https://graph.microsoft.com/beta/teams/{teamId}/schedule/timeOffReasons](https://graph.microsoft.com/beta/teams/%7bteamId%7d/schedule/timeOffReasons)  *Request Body* *{*  *"displayName": "Vacation",*  *"iconType": "plane",*  *"isActive": true*  *}* | Group.ReadWrite.All | Integration Service API – Post to Shifts App |
|  | List TimeOffReason | Request Type: GET  [https://graph.microsoft.com/beta/teams/{teamId}/schedule/timeOffReasons](https://graph.microsoft.com/beta/teams/%7bteamId%7d/schedule/timeOffReasons) | Group.ReadWrite.All | Integration Service API – Get from Shifts App |
|  | Create Time Off | Request Type: POST  [https://graph.microsoft.com/beta/teams/{teamId}/schedule/timesOff](https://graph.microsoft.com/beta/teams/%7bteamId%7d/schedule/timesOff)  *Request Body* *{*  *"userId": "c5d0c76b-80c4-481c-be50-923cd8d680a1",*  *"sharedTimeOff": {*  *"timeOffReasonId": "TOR\_891045ca-b5d2-406b-aa06-a3c8921245d7",*  *"startDateTime": "2019-03-11T07:00:00Z",*  *"endDateTime": "2019-03-12T07:00:00Z",*  *"theme": "white"*  *},*  *"draftTimeOff": {*  *"timeOffReasonId": "TOR\_891045ca-b5d2-406b-aa06-a3c8921245d7",*  *"startDateTime": "2019-03-11T07:00:00Z",*  *"endDateTime": "2019-03-12T07:00:00Z",*  *"theme": "pink"*  *}*  *}* | Group.ReadWrite.All | Integration Service API – Post to Shifts App |
|  | Approve Time Off Request | Request Type: POST  [https://graph.microsoft.com/beta/teams/{id}/schedule/timeOffRequests/approve](https://graph.microsoft.com/beta/teams/%7bid%7d/schedule/timeOffRequests/approve)  *{*  *"message": "message-value"*  *}* | Group.ReadWrite.All | Integration Service API – Post to Shifts App |
|  | Decline Time Off Request | Request Type: POST  [https://graph.microsoft.com/beta/teams/{id}/schedule/timeOffRequests/decline](https://graph.microsoft.com/beta/teams/%7bid%7d/schedule/timeOffRequests/decline)  *{*  *"message": "message-value"*  *}* | Group.ReadWrite.All | Integration Service API – Post to Shifts App |
|  | Create Open Shift | Request Type: POST  [https://graph.microsoft.com/beta/teams/{TeamsId}/schedule/openshifts](https://graph.microsoft.com/beta/teams/%7bTeamsId%7d/schedule/openshifts)  *{*  *"schedulingGroupId":"TAG\_228940ed-ff84-4e25-b129-1b395cf78be0",*  *"sharedOpenShift":{*  *"notes":"InventoryManagement",*  *"openSlotCount":2,*  *"displayName":"Dayshift",*  *"startDateTime":"2018-10-04T00: 58: 45.340Z",*  *"endDateTime":"2018-10-04T09: 50: 45.332Z",*  *"theme":"white",*  *"activities":[*  *{*  *"isPaid":true,*  *"startDateTime":"2018-10-04T00: 58: 45.340Z",*  *"endDateTime":"2018-10-04T01: 58: 45.340Z",*  *"code":"",*  *"displayName":"Lunch"*  *}*  *]*  *},*  *"draftOpenShift":{*  *"notes":"InventoryManagement",*  *"openSlotCount":3,*  *"displayName":"Dayshift",*  *"startDateTime":"2018-10-04T00: 58: 45.332Z",*  *"endDateTime":"2018-10-04T08: 58: 45.340Z",*  *"theme":"white",*  *"activities":[*  *{*  *"isPaid":true,*  *"startDateTime":"2018-10-04T00: 58: 45.340Z",*  *"endDateTime":"2018-10-04T07: 58: 45.332Z",*  *"code":"Break",*  *"displayName":"Lunch"*  *}*  *]*  *},*  *"createdDateTime":"2019-03-14T04: 32: 51.451Z",*  *"lastModifiedDateTime":"2019-03-14T05: 32: 51.451Z",*  *"lastModifiedBy":{*  *"application":null,*  *"device":null,*  *"conversation":null,*  *"user":{*  *"id":"366c0b19-49b1-41b5-a03f-9f3887bd0ed8",*  *"displayName":"JohnDoe"*  *}*  *}*  *}* | Group.ReadWrite.All | Integration Service API – Post to Shifts App |
|  | Delete Open Shift | Request Type: DELETE  [https://graph.microsoft.com/beta/teams/{id}/schedule/openShifts/{openShiftId}](https://graph.microsoft.com/beta/teams/%7bid%7d/schedule/openShifts/%7bopenShiftId%7d) | Group.ReadWrite.All | Integration Service API – Post to Shifts App |
|  | Approve Open Shift Request | Request Type: POST [https://graph.microsoft.com/beta/teams/{id}/schedule/openShiftsChangeRequests/{openShiftChangeRequestId}/approve](https://graph.microsoft.com/beta/teams/%7bid%7d/schedule/openShiftsChangeRequests/%7bopenShiftChangeRequestId%7d/approve)  {  “message”: “message-value”  } | Group.ReadWrite.All | Integration Service API – Post to Shifts App |
|  | Decline Open Shift Request | Request Type: POST  [https://graph.microsoft.com/beta/teams/{id}/schedule/openShiftsChangeRequests/{openShiftChangeRequestId}/decline](https://graph.microsoft.com/beta/teams/%7bid%7d/schedule/openShiftsChangeRequests/%7bopenShiftChangeRequestId%7d/decline)  {  “message”: “message-value”  } | Group.ReadWrite.All | Integration Service API – Post to Shifts App |
|  | Get Open Shift Request | Request Type: GET  [https://graph.microsoft.com/beta/teams/{id}/schedule/openShiftsChangeRequests/SREQ\_0b87dd20-d5ed-4764-9c3e-cfc8516def09](https://graph.microsoft.com/beta/teams/%7bid%7d/schedule/openShiftsChangeRequests/SREQ_0b87dd20-d5ed-4764-9c3e-cfc8516def09) | Group.ReadWrite.All | N/A |
| 1. A | Approve Swap Shift Request | Request Type: POST  [https://graph.microsoft.com/beta/teams/{id}/schedule/swapShiftsChangeRequests/{swapShiftChangeRequestId}/approve](https://graph.microsoft.com/beta/teams/%7bid%7d/schedule/swapShiftsChangeRequests/%7bswapShiftChangeRequestId%7d/approve)  {  “message”: “message-value”  } | Group.ReadWrite.All | Integration Service API – Post to Shifts App |
|  | Decline Swap Shift Request | Request Type: POST  [https://graph.microsoft.com/beta/teams/{id}/schedule/swapShiftsChangeRequests/{swapShiftChangeRequestId}/decline](https://graph.microsoft.com/beta/teams/%7bid%7d/schedule/swapShiftsChangeRequests/%7bswapShiftChangeRequestId%7d/decline)  {  “message”: “message-value”  } | Group.ReadWrite.All | Integration Service API – Post to Shifts App |
|  | Share Schedule | Request Type: POST  [https://graph.microsoft.com/beta/teams/{id}/schedule/share](https://graph.microsoft.com/beta/teams/%7bid%7d/schedule/share)  {  “notifyTeam”: true,  “startDateTime”: “*2019-03-14T04: 32",*  *“endDateTime”: 2019-03-14T04: 32",*  } | Group.ReadWrite.All | Integration Service API – Post to Shifts App |

# Microsoft Graph APIs for Approval/Decline Request workflow for Open Shift and Swap Shift Requests

The passthrough header (Name: X-MS-WFMPassthrough) is added whenever below mentioned Graph APIs are called from Integration (ex: Open Shift request approval). It used to validate the source of the Graph API call.

Whenever integration submits Graph API requests, it sets the value of header to the Workforce Integration Id, which is retrieved from the ConfigurationInfo table. This value is validated in below scenario:

1. Whenever an outbound callback is received from Shifts App because of an action performed by the Integration service, above header value gets passed as part of the same. Integration services reads it and accordingly treats it as valid call (200 OK Request)
2. If the callback is not result of an action performed by Integration service, above pass through header value will result into mismatch scenario. Integration service treats it as HTTP 400 (Bad Request) and returns appropriate message. Ex: Either the approval or decline of the Open Shift Request or Swap Shift Request is made in the Shifts App UI by the FLM OR request approval/decline outbound call is coming from a different Workforce Integration.

Graph calls made via the Integration Service API to approve or decline Open Shift Requests, or Swap Shift Requests, generate a callback to the Integration Service API with different JSON payloads. Below are the Graph API calls and their corresponding payloads:

## Open Shift Request Approval

[https://graph.microsoft.com/beta/teams/{id}/schedule/openShiftsChangeRequests/{openShiftChangeRequestId}/approve](https://graph.microsoft.com/beta/teams/%7bid%7d/schedule/openShiftsChangeRequests/%7bopenShiftChangeRequestId%7d/approve)

Request Type- POST

Generates below outbound call

POST https://xyzWorkforceIntegration.com/Contoso/v1/teams/a3751044-a1b5-568e-b742-f78d01d68a67/update

Content-type: application/json

X-MS-WFMPassthrough: <WorkforceIntegrationId>

{

"requests":[

{

"id":"SHFT\_0dc759e4-148f-4fb5-a94d-aa61c93062ad",

"method":"POST",

"url":"/shifts/SHFT\_0dc759e4-148f-4fb5-a94d-aa61c93062ad",

"headers":{

"X-MS-Transaction-ID":"1",

"X-MS-Expires":"2019-10-05T00:30:46.3615394Z"

},

"body":{

"sharedShift":{

"displayName":"Test case #2",

"activities":[

],

"isActive":true,

"startDateTime":"2019-12-02T17:00:00.000Z",

"endDateTime":"2019-12-03T01:00:00.000Z",

"theme":"Pink"

},

"userId":"1c717a55-febd-4850-b5f6-101f3a29972c",

"schedulingGroupId":"TAG\_d18fd675-3ac8-41b2-8038-d17fdac8b0d3",

"lastModifiedDateTime":"2019-10-05T00:30:16.222Z",

"lastModifiedBy":{

"user":{

"id":"73d77339-b44a-4c1d-9606-c1e33559b7e2",

"displayName":"Vijay Agrawal"

}

},

"id":"SHFT\_0dc759e4-148f-4fb5-a94d-aa61c93062ad"

}

},

{

"id":"SREQ\_f999db32-a0f9-4365-b20e-12d87f941ef8",

"method":"PUT",

"url":"/openshiftrequests/SREQ\_f999db32-a0f9-4365-b20e-12d87f941ef8",

"headers":{

"X-MS-Transaction-ID":"1",

"X-MS-Expires":"2019-10-05T00:30:46.3615394Z"

},

"body":{

"assignedTo":"Manager",

"state":"Approved",

"senderDateTime":"2019-10-05T00:25:42.125Z",

"senderMessage":"This open shift request should fail",

"senderUserId":"1c717a55-febd-4850-b5f6-101f3a29972c",

"managerActionDateTime":"2019-10-05T00:30:16.222Z",

"managerActionMessage":"Test case #2 approval",

"managerUserId":"73d77339-b44a-4c1d-9606-c1e33559b7e2",

"openShiftId":"OPNSHFT\_1c21d1bc-7ceb-4219-937b-ef14266e74c8",

"createdDateTime":"2019-10-05T00:25:42.125Z",

"lastModifiedDateTime":"2019-10-05T00:30:16.222Z",

"id":"SREQ\_f999db32-a0f9-4365-b20e-12d87f941ef8"

}

},

{

"id":"OPNSHFT\_1c21d1bc-7ceb-4219-937b-ef14266e74c8",

"method":"PUT",

"url":"/openshifts/OPNSHFT\_1c21d1bc-7ceb-4219-937b-ef14266e74c8",

"headers":{

"X-MS-Transaction-ID":"1",

"X-MS-Expires":"2019-10-05T00:30:46.3615394Z"

},

"body":{

"sharedOpenShift":{

"openSlotCount":0,

"displayName":"Test case #2",

"activities":[

],

"isActive":true,

"startDateTime":"2019-12-02T17:00:00.000Z",

"endDateTime":"2019-12-03T01:00:00.000Z",

"theme":"Pink"

},

"schedulingGroupId":"TAG\_d18fd675-3ac8-41b2-8038-d17fdac8b0d3",

"createdDateTime":"2019-10-05T00:19:09.156Z",

"lastModifiedDateTime":"2019-10-05T00:30:16.222Z",

"id":"OPNSHFT\_1c21d1bc-7ceb-4219-937b-ef14266e74c8"

}

},

{

"id":"SREQ\_fe3f3c5a-fd46-471a-9f6b-978471cb6a35",

"method":"PUT",

"url":"/openshiftrequests/SREQ\_fe3f3c5a-fd46-471a-9f6b-978471cb6a35",

"headers":{

"X-MS-Transaction-ID":"1",

"X-MS-Expires":"2019-10-05T00:30:46.3615394Z"

},

"body":{

"assignedTo":"System",

"state":"Declined",

"senderDateTime":"2019-10-05T00:27:28.223Z",

"senderMessage":"",

"senderUserId":"66b4f2a4-425d-4dec-8172-7e889950885e",

"managerUserId":"73d77339-b44a-4c1d-9606-c1e33559b7e2",

"eTag":"2b6f011c-8517-4fbe-9269-99a0c4766811",

"openShiftId":"OPNSHFT\_1c21d1bc-7ceb-4219-937b-ef14266e74c8",

"createdDateTime":"2019-10-05T00:27:28.223Z",

"lastModifiedDateTime":"2019-10-05T00:30:16.222Z",

"id":"SREQ\_fe3f3c5a-fd46-471a-9f6b-978471cb6a35"

}

}

]

}

Above payload has following list of requests:

a. Open shift request with state as ‘Approved’

b. Open Shift to be deleted

c. New Shift to be created

When same Open Shift is requested by more than one user, FLM can approve it only for one of them. In that case, payload also includes multiple open shift requests with state as system declined for other users.

## Open Shift Request Decline

[https://graph.microsoft.com/beta/teams/{id}/schedule/openShiftsChangeRequests/{openShiftChangeRequestId}/decline](https://graph.microsoft.com/beta/teams/%7bid%7d/schedule/openShiftsChangeRequests/%7bopenShiftChangeRequestId%7d/decline)

Request Type: POST

Generates below outbound call

POST https://xyzWorkforceIntegration.com/Contoso/v1/teams/a3751044-a1b5-568e-b742-f78d01d68a67/update

Content-type: application/json

X-MS-WFMPassthrough: <worforceIntegrationId>

{

"requests":[

{

"id":"SREQ\_1e12a3c9-53ae-4884-b444-19440efbf236",

"method":"PUT",

"url":"/openshiftrequests/SREQ\_1e12a3c9-53ae-4884-b444-19440efbf236",

"headers":{

"X-MS-Transaction-ID":"1",

"X-MS-Expires":"2019-10-05T00:40:44.8303933Z"

},

"body":{

"assignedTo":"Manager",

"state":"Declined",

"senderDateTime":"2019-10-05T00:38:58.235Z",

"senderMessage":"",

"senderUserId":"3041ccde-7544-4ae0-b44f-3618b08ba1ce",

"managerActionDateTime":"2019-10-05T00:40:14.686Z",

"managerActionMessage":"Test case #3 decline",

"managerUserId":"73d77339-b44a-4c1d-9606-c1e33559b7e2",

"openShiftId":"OPNSHFT\_53c1dcc6-f0c5-4c43-ba23-42ef016ea1d1",

"createdDateTime":"2019-10-05T00:38:58.235Z",

"lastModifiedDateTime":"2019-10-05T00:40:14.686Z",

"id":"SREQ\_1e12a3c9-53ae-4884-b444-19440efbf236"

}

}

]

}

Above payload has following request:

a. Open shift request with state ‘Declined’

## Approve Swap Shift Request

https://graph.microsoft.com/beta/teams/{id}/schedule/swapShiftsChangeRequests/{swapShiftChangeRequestId}/approve

Request Type: POST

Generates below outbound call

POST https://xyzWorkforceIntegration.com/Contoso/v{apiVersion}/teams/{aad\_group\_id}/update Accept-Language: en-us

Content-type: application/json

X-MS-WFMPassthrough: <WorkforceIntegrationId>

{

"requests": [

{

"id": "SREQ\_b61adbf8-631f-4121-9bf8-cb7e6b5382b2",

"method": "PUT",

"url": "/swapRequests/SREQ\_b61adbf8-631f-4121-9bf8-cb7e6b5382b2",

"headers": {

"X-MS-Transaction-ID": "1",

"X-MS-Expires": "2019-07-11T22:31:01.8637693Z"

},

"body": {

"assignedTo": "Manager",

"state": "Approved",

"senderDateTime": "2019-07-11T22:28:27.951Z",

"senderMessage": "",

"senderUserId": "b0df9d3a-397a-41f8-8f1f-f19c7ccc408c",

"managerActionDateTime": "2019-07-11T22:30:31.713Z",

"managerUserId": "1c717a55-febd-4850-b5f6-101f3a29972c",

"recipientUserId": "f2a98117-1850-423e-bf25-5a38a81cdf4f",

"recipientActionMessage": "Recipient accepting",

"recipientActionDateTime": "2019-07-11T22:29:38.536Z",

"eTag": "583e848c-fc7b-4f29-a3ae-ad2cebdedd84",

"senderShiftId": "SHFT\_9f3f2596-e0fc-47b6-97c2-38d570df6fd1",

"recipientShiftId": "SHFT\_0ac3ad5b-68c1-468a-ab57-7c4caeb2d984",

"createdDateTime": "2019-07-11T22:28:27.951Z",

"lastModifiedDateTime": "2019-07-11T22:30:31.713Z",

"id": "SREQ\_b61adbf8-631f-4121-9bf8-cb7e6b5382b2"

}

},

{

"id": "SHFT\_9f3f2596-e0fc-47b6-97c2-38d570df6fd1",

"method": "DELETE",

"url": "/shifts/SHFT\_9f3f2596-e0fc-47b6-97c2-38d570df6fd1",

"headers": {

"X-MS-Transaction-ID": "1",

"X-MS-Expires": "2019-07-11T22:31:01.8637693Z"

}

},

{

"id": "SHFT\_0ac3ad5b-68c1-468a-ab57-7c4caeb2d984",

"method": "DELETE",

"url": "/shifts/SHFT\_0ac3ad5b-68c1-468a-ab57-7c4caeb2d984",

"headers": {

"X-MS-Transaction-ID": "1",

"X-MS-Expires": "2019-07-11T22:31:01.8637693Z"

}

},

{

"id": "SHFT\_b3bd54a3-f825-494f-bc22-96b66cc0e0d7",

"method": "POST",

"url": "/shifts/SHFT\_b3bd54a3-f825-494f-bc22-96b66cc0e0d7",

"headers": {

"X-MS-Transaction-ID": "1",

"X-MS-Expires": "2019-07-11T22:31:01.8637693Z"

},

"body": {

"sharedShift": {

"displayName": "S20",

"activities": [],

"isActive": true,

"startDateTime": "2019-07-12T15:00:00.000Z",

"endDateTime": "2019-07-13T00:00:00.000Z",

"theme": "Blue"

},

"schedulingGroupId": "TAG\_642c89b0-e478-453d-9430-19fdb8a931b5",

"lastModifiedDateTime": "2019-07-11T22:30:31.713Z",

"id": "SHFT\_b3bd54a3-f825-494f-bc22-96b66cc0e0d7"

}

},

{

"id": "SHFT\_4a302963-5748-4a34-a2ce-bf444ce72535",

"method": "POST",

"url": "/shifts/SHFT\_4a302963-5748-4a34-a2ce-bf444ce72535",

"headers": {

"X-MS-Transaction-ID": "1",

"X-MS-Expires": "2019-07-11T22:31:01.8637693Z"

},

"body": {

"sharedShift": {

"displayName": "S30",

"activities": [],

"isActive": true,

"startDateTime": "2019-07-12T15:00:00.000Z",

"endDateTime": "2019-07-13T00:00:00.000Z",

"theme": "Blue"

},

"schedulingGroupId": "TAG\_642c89b0-e478-453d-9430-19fdb8a931b5",

"lastModifiedDateTime": "2019-07-11T22:30:31.713Z",

"id": "SHFT\_4a302963-5748-4a34-a2ce-bf444ce72535"

}

}

]

}

Above payload has following requests

* Swap Shift Request with state as ‘Approved’
* Old shift to be deleted for requestor (FLW1)
* Old shift to be deleted for requested for user (FLW2)
* New shift to be created for requestor (FLW1)
* New shift to be created for requested for user (FLW2)

In case of multiple Swap requests for same Shift by a user, FLM can approve it only for one Swap. In that case, payload also includes multiple swap shift requests with state as system declined for other users.

## Decline Swap Shift Request

[https://graph.microsoft.com/beta/teams/{id}/schedule/swapShiftsChangeRequests/{swapShiftChangeRequestId}/approve](https://graph.microsoft.com/beta/teams/%7bid%7d/schedule/swapShiftsChangeRequests/%7bswapShiftChangeRequestId%7d/approve)

Request Type: POST

Generates below outbound call

POST https://xyzWorkforceIntegration.com/Contoso/v{apiVersion}/teams/{aad\_group\_id}/update

Accept-Language: en-us

Content-type: application/json

X-MS-WFMPassthrough: <WorkforceIntegrationId>

{

"requests": [

{

"id": "1",

"method": "PUT",

"url": "/swapRequests/SREQ\_b61adbf8-631f-4121-9bf8-cb7e6b5382b2",

"headers": {

"X-MS-Transaction-ID": "123",

"X-MS-Expires": "2019-07-11T22:31:01.8637693Z"

},

"body": {

"id": "SREQ\_b61adbf8-631f-4121-9bf8-cb7e6b5382b2",

"senderShiftId": "5ad10161-6524-4c7c-9beb-4e8677ba2f6d",

"recipientShiftId": "e73408ca-3ea5-4bbf-96a8-2e06c95f7a2c",

"assignedTo": "manager",

"state": "declined ",

"senderUserId": "3fe0bc21-1398-4fd9-9713-52511b434c1e",

"senderDateTime": "2019-05-01T10:00:00Z",

"senderMessage": "I can't make my shift, any chance we can swap?",

"recipientUserId": "567c8ea5-9e32-422a-a663-8270201699cd",

"recipientActionDateTime": 2019-05-01T11:00:00Z,

"recipientActionMessage": "Approved!",

"managerUserId": "349dh38s-4597-7sg2-2836-34979f379w7f",

"managerActionDateTime": 2019-05-01T11:30:00Z,

"managerActionMessage": " You can't take the 8pm shift as you don't have proper training."

}

}

]

}

Above payload has following request

a. Swap shift request with status as ‘declined’

## Kronos WFC APIs Description

Below table lists necessary Kronos WFC API methods used by this integration as part of SOAP request calls.

**SOAP API Endpoint**: All the APIs listed below require the same endpoint

Ex: <https://kronosoua-dev.kronos.net/wfc/webservice/WFCWebServices?contenttype=text/xml&SoapAction=http://localhost/wfc/XMLAPISchema/>

|  |  |  |
| --- | --- | --- |
| Sr. No | Use Case | SOAP XML Request |
|  | Login to Kronos | *<?xml version="1.0" encoding="UTF-8"?>*  *<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:hs="http://localhost/wfc/XMLAPISchema" >*  *<soapenv:Body>*  *<hs:KronosWFC>*  *<Kronos\_WFC version="1.0">*  *<Request Object="System" Action="Logon" Username="<<>>" Password="<<>>"/>*  *</Kronos\_WFC>*  *</hs:KronosWFC>*  *</soapenv:Body>*  *</soapenv:Envelope>* |
|  | Get User jobs | Request Type: GET  *<?xml version='1.0' encoding='UTF-8'?>*  *<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/' xmlns:hs='http://localhost/wfc/XMLAPISchema'>*  *<soapenv:Body>*  *<hs:KronosWFC>*  *<Kronos\_WFC version='1.0'>*  *<Request Action='Load'>*  *<JobAssignment>*  *<Identity>*  *<PersonIdentity PersonNumber ='<<>>' />*  *</Identity>*  *</JobAssignment>*  *</Request>*  *</Kronos\_WFC>*  *</hs:KronosWFC>*  *</soapenv:Body>*  *</soapenv:Envelope>* |
|  | Get list of all Users | Request Type: GET  *<?xml version="1.0" encoding="UTF-8"?>*  *<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:hs="http://localhost/wfc/XMLAPISchema">*  *<soapenv:Body>*  *<hs:KronosWFC>*  *<Kronos\_WFC version="1.0">*  *<Request Action ='RunQuery'>*  *<HyperFindQuery HyperFindQueryName="<<>> " VisibilityCode='Public' QueryDateSpan='<<5/11/2019 - 6/22/2019 >>'>*  *</HyperFindQuery>*  *</Request>*  *</Kronos\_WFC>*  *</hs:KronosWFC>*  *</soapenv:Body>*  *</soapenv:Envelope>* |
|  | Get all shifts of a user | Request Type: GET  *<?xml version='1.0' encoding='UTF-8'?>*  *<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/' xmlns:hs='http://localhost/wfc/XMLAPISchema'>*  *<soapenv:Body>*  *<hs:KronosWFC>*  *<Kronos\_WFC version='1.0'>*  *<Request Action ='Load'>*  *<Schedule OrgJobPath="<<Corporate/Grocery/Region 1/District 1/Store 0404/Frontend/Cashier>>" QueryDateSpan ='<<9/09/2019 - 9/13/2019>>'>*  *<Employees>*  *<PersonIdentity PersonNumber ='<<1>>' />*  *</Employees>*  *</Schedule>*  *</Request>*  *</Kronos\_WFC>*  *</hs:KronosWFC>*  *</soapenv:Body>*  *</soapenv:Envelope>* |
|  | Get list of all time offs for a user | Request Type: GET  *<?xml version='1.0' encoding='UTF-8'?>*  *<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/' xmlns:hs='http://localhost/wfc/XMLAPISchema'>*  *<soapenv:Body>*  *<hs:KronosWFC>*  *<Kronos\_WFC version='1.0'>*  *<Request Action='Retrieve'>*  *<RequestMgmt QueryDateSpan='<<1/8/2019-9/15/2019>>' RequestFor="TOR">*  *<Employees>*  *<PersonIdentity PersonNumber='<<117>>'/>*  *</Employees>*  *</RequestMgmt>*  *</Request>*  *</Kronos\_WFC>*  *</hs:KronosWFC>*  *</soapenv:Body>*  *</soapenv:Envelope>* |
|  | Get list of all Swap Shifts of a user | Request Type: GET  *<?xml version='1.0' encoding='UTF-8'?>*  *<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/' xmlns:hs='http://localhost/wfc/XMLAPISchema'>*  *<soapenv:Body>*  *<hs:KronosWFC>*  *<Kronos\_WFC version='1.0'>*  *<Request Action='RetrieveWithDetails'>*  *<RequestMgmt QueryDateSpan='<<6/1/2019-8/30/2019>>' RequestFor="Shift Swap Request">*  *<Employees>*  *<PersonIdentity PersonNumber='<<117>>'/>*  *</Employees>*  *</RequestMgmt>*  *</Request>*  *</Kronos\_WFC>*  *</hs:KronosWFC>*  *</soapenv:Body>*  *</soapenv:Envelope>* |
| 10. | Create Time Off | Request Type: POST  *<?xml version ='1.0' encoding='UTF-8'?> <soapenv:Envelope xmlns:soapenv=' xmlns:hs = '*[*http://localhost/wfc/XMLAPISchema'>*](http://localhost/wfc/XMLAPISchema%27%3e) *<soapenv:Body>         <hs:KronosWFC>             <Kronos\_WFC version='1.0'>                 <Request Action="Add" Sequence="1">                     <RequestMgmt QueryDateSpan="<<06/14/2019-06/16/2019>>">                         <Employees>                             <PersonIdentity PersonNumber="<<1>>" />                         </Employees>                         <RequestItems>                             <GlobalTimeOffRequestItem>                                 <TimeOffPeriods>                                     <TimeOffPeriod StartDate="<<06/27/2019>>" EndDate="<<06/27/2019>>" Duration="FULL\_DAY" PayCodeName="Personal"></TimeOffPeriod>                                 </TimeOffPeriods>                                 <RequestFor>TOR</RequestFor>                                 <Employee>                                     <PersonIdentity>                                         <PersonNumber><<117>></PersonNumber>                                     </PersonIdentity>                                 </Employee>                             </GlobalTimeOffRequestItem>                         </RequestItems>                     </RequestMgmt>                 </Request>             </Kronos\_WFC>         </hs:KronosWFC>     </soapenv:Body> </soapenv:Envelope>* |
| 10. | Create Swap Shift | Step 1.  Request Type: GET  *<Request Action='LoadJobs'>*  *<SwapShiftJobs QueryDate ='5/20/2019' StartTime ='09:00AM' EndTime='02:00PM'>*  *<Employee>*  *<PersonIdentity PersonNumber='<<117>>'/>*  *</Employee>*  *</SwapShiftJobs>*  *</Request>*  Step 2.  Request Type: GET  *<Request Action='LoadJobs'>*  *<SwapShiftJobs QueryDate ='7/03/2019' StartTime ='09:00AM' EndTime='02:00PM'>*  *<Employee>*  *<PersonIdentity PersonNumber='<<107>>'/>*  *</Employee>*  *</SwapShiftJobs>*  *</Request>*  *</Kronos\_WFC>*  *</hs:KronosWFC>*  *</soapenv:Body>*  *</soapenv:Envelope>*   1. Request Type : GET   *<Request Action='LoadEligibleEmployees'>*  *<SwapShiftEmployees QueryDate ='06/21/2019' ShiftSwapDate='06/24/2019' StartTime ='9:00AM' EndTime='2:00PM'>*  *<Employee>*  *<PersonIdentity PersonNumber='<<107>>'/>*  *</Employee>*  *</SwapShiftEmployees>*  *</Request>*  *</Kronos\_WFC>*  *</hs:KronosWFC>*  *</soapenv:Body>*  *</soapenv:Envelope>*  Step 3.  Request Type: POST  *<?xml version='1.0' encoding='UTF-8'?>*  *<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/' xmlns:hs='http://localhost/wfc/XMLAPISchema'>*  *<soapenv:Body>*  *<hs:KronosWFC>*  *<Kronos\_WFC version='1.0'>*  *<Request Action ='AddRequests'>*  *<EmployeeRequestMgmt QueryDateSpan='08/29/2019 - 09/29/2019'>*  *<Employee>*  *<PersonIdentity PersonNumber='<<107>>' />*  *</Employee>*  *<RequestItems>*  *<SwapShiftRequestItem RequestFor='Shift Swap Request'>*  *<Employee>*  *<PersonIdentity PersonNumber='<<107>>' />*  *</Employee>*  *<OfferedShift>*  *<ShiftRequestItem StartDateTime='8/29/2019 9:00AM' EndDateTime='8/29/2019 2:00PM' OrgJobPath='<<Corporate/Grocery/Region 1/District 1/Store 0404/Frontend/Cashier>>'>*  *<Employee>*  *<PersonIdentity PersonNumber='<<107>>' />*  *</Employee>*  *</ShiftRequestItem>*  *</OfferedShift>*  *<RequestedShift>*  *<ShiftRequestItem StartDateTime='8/29/2019 9:00AM' EndDateTime='8/29/2019 2:00PM' OrgJobPath='<<Corporate/Grocery/Region 1/District 1/Store 0404/Frontend/Cashier>>'>*  *<Employee>*  *<PersonIdentity PersonNumber='<<205>>' />*  *</Employee>*  *</ShiftRequestItem>*  *</RequestedShift>*  *</SwapShiftRequestItem>*  *</RequestItems>*  *</EmployeeRequestMgmt>*  *</Request>*  *</Kronos\_WFC>*  *</hs:KronosWFC>*  *</soapenv:Body>*  *</soapenv:Envelope>* |

## Data Store

Integration Service API endpoints and Configuration Web App uses Azure Table storage for storing and looking up data. Below is the list of tables:

1. ConfigurationInfo
2. UserToUserMapping
3. TeamToDepartmentWithJobMapping
4. PayCodeToTimeOffReasons
5. ShiftEntityMapping
6. TimeOffMapping
7. OpenShiftEntityMapping
8. OpenShiftRequestMapping
9. SwapShiftMapping

### ConfigurationInfo

Stores the following information about the configuration of the workforce integrations. All the information defined below will come from the first page of the Configuration Web App.

Table 1: The table for storing the Configuration info.

|  |  |  |
| --- | --- | --- |
| Property Name | Data Type | Description |
| PartitionKey | String | The Tenant ID of the Microsoft Teams instance |
| RowKey | String | The newly generated unique id |
| AdminAadObjectId | String | This is the AAD Object Id of the Tenant Admin that has logged into the Configuration Web App |
| ConfigurationId | String | The unique configuration id |
| WfmApiEndpoint | String | The Kronos WFC SOAP API endpoint |
| WfmProvider | String | A value that is set to a constant with the value of Kronos WFC |
| WorkforceIntegrationId | String | This is the Workforce Integration ID which comes from a Graph API call to register the workforce integration |
| WorkforceIntegrationSecret | String | A 64-character string that would be used for decrypting encrypted JSON payloads coming from the Shifts App |

Kronos Endpoint, Kronos SuperUser name and Kronos SuperUser password details are stored in the Azure Key Vault.

### User to User Mapping

The missing component for all the mappings to properly work together is the mapping of users between the Kronos WFC system, and Microsoft Teams. The following table below outlines the necessary schema involved when mapping users between Kronos and Teams:

Table 2: UserToUserMapping between Kronos and Shifts

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| PartitionKey | String | Kronos OrgJobPath |
| RowKey | String | KronosPersonNumber - The unique identifier of the user in Kronos |
| IsActive | Boolean | If the user is still active in Kronos. If they are inactive we will not include them in processing, therefore no related entities will sync. |
| KronosUserName | String | User’s complete name in Kronos |
| ShiftUserAadObjectId | String | The AAD Object Id of the user |
| ShiftUserDisplayName | String | User’s display name in Teams |
| ShiftUserUpn | String | User’s AAD principle name |

### Team to Department Mapping with Job Mapping

Another key aspect of the Configuration Web App is the ability to establish the mapping between a Team and its Scheduling Groups in Shifts to its respective Department (Labor level) and its Jobs in Kronos WFC. The following table outlines the necessary schema:

Table 3: TeamDepartmentWithJobMapping table

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| PartitionKey | String | This is the Workforce Integration Id for the integration |
| RowKey | String | Kronos Org Job Path |
| KronosTimeZone | String | The Kronos teams time zone. |
| ShiftsTeamName | String | The name of the Team in Microsoft Shifts App |
| TeamsId | String | Unique Id of a Team in tenant |
| TeamsScheduleGroupId | String | Scheduling Group Id in a team |
| TeamsScheduleGroupName | String | The scheduling group name |

The Workforce Integration Id is the Integration Id in Shifts App. The relationship established as follows:

* There is a 1:1 relationship between the ConfigurationInfo and WorkforceIntegration entities.
* There is a 1: Many relationship between the WorkforceIntegration and TeamDepartmentMapping entities
* There is a 1: Many relationship between Team and Scheduling Groups

### Paycode to TimeOffReasons Mapping

The PayCode mapping of Kronos with their corresponding TimeOffReason where there is a 1:1 relationship of PayCode of Shifts with TimeOffReason in Kronos.

Table 4: PayCodeToTimeOffReasonsMapping

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| PartitionKey | String | Shifts App Team id |
| RowKey | String | TimeOffReason – Shifts App Time Off Reason name |
| TimeOffReasonID | String | Shifts App unique Time Off Reason Id |

### ShiftEntityMapping

The OpenShiftEntityMapping table manages the Shift entities that are transferred between Kronos WFC and the Shifts App.

Table 5: ShiftEntityMapping

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| PartitionKey | String | Monthwise partition for storing Shift details, it would be Month\_Year |
| RowKey | String | Unique id for Shift Entity in Teams |
| AadUserId | String | AAD object id of the user |
| KronosPersonNumber | String | KronosPersonNumber - Kronos unique id for the user |
| KronosUniqueId | String | Hash of different attributes like Startdatetime, enddatetime, type, notes for a given shift segment |
| ShiftStartDate | String | Start date of the Shift |
| ShiftEndDate | String | End date of the shift |

### Time Off Mapping

This table is used to perform operations such as approve, decline on Time off and Time Off Requests created in Teams & Kronos.

Table 6: TimeOffMapping

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| PartitionKey | String | Monthwise partition for storing TimeOff details, it would be Month\_Year |
| RowKey | String | Kronos Request ID for time off |
| Duration | String | Time off duration |
| EndDate | String | Time off end date. |
| IsActive | Boolean | Flag to check whether Kronos operation is successful or not |
| KronosPersonNumber | String | Kronos Person number of the user |
| KronosRequestId | String | Kronos Request ID for time off |
| PayCodeName | String | Time off pay code name |
| ShiftsRequestId | String | Teams request id |
| StartDate | String | Time off start date |
| StartTime | String | Time off start time |
| ShiftsStatus | String | Time off status in Shifts. (Approved, Submitted etc.) |
| KronosStatus | String | Time off status in Kronos. (Approved, Submitted etc.) |

### Open Shift Entity Mapping

The OpenShiftEntityMapping table manages the Open Shifts that are transferred between Kronos WFC and the Shifts App.

Table 7: OpenShiftEntityMapping

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| PartitionKey | String | Monthwise partition for storing Open Shift Request details, it would be Month\_Year |
| RowKey | String | Hash of attributes of Open Shift created using Kronos open shift details |
| KronosSlots | Integer | Number of slots for Open Shifts |
| OpenShiftStartDate | DateTime | Open Shift start date |
| OrgJobPath | String | Kronos org job path of Open Shift |
| SchedulingGroupId | String | Teams Scheduling Group Id |
| TeamsOpenShiftId | String | Teams Open Shift Id |

### Open Shift Request Mapping

This table is used to manage open shift requests mapping in Teams & Kronos WFC. The OpenShiftRequest Mapping table is to be populated after there is a successful Open Shift Request submission in Kronos WFC.

Table 8: OpenShiftRequestMapping

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| PartitionKey | String | Monthwise partition for storing Open Shift Request details, it would be Month\_Year |
| RowKey | String | TeamsOpenShiftRequestId - FLW is raising request only through Shifts |
| TeamsOpenShiftId | String | Teams Open shift Id |
| KronosOpenShiftUniqueId | String | Hash of attributes of Open Shift, need to be obtained by using above table OpenShiftMapping for a given Teams Open Shift Id |
| AadUserId | String | AAD object id of the Shifts user who has requested Open Shift |
| KronosSenderPersonNumber | String | Kronos Person number of the user corresponding to AAD of user who initiated Open Shift Request in Teams |
| KronosOpenShiftRequestId | String | Request Id generated once Open Shift Request gets created in Kronos |
| KronosStatus | String | Status of Open Shift Request processing in Kronos |
| ShiftsStatus | String | Status of Open Shift Request processing in Shifts |

### Swap Shift Request Mapping

This table is used to manage swap shift requests mapping in Teams & Kronos.

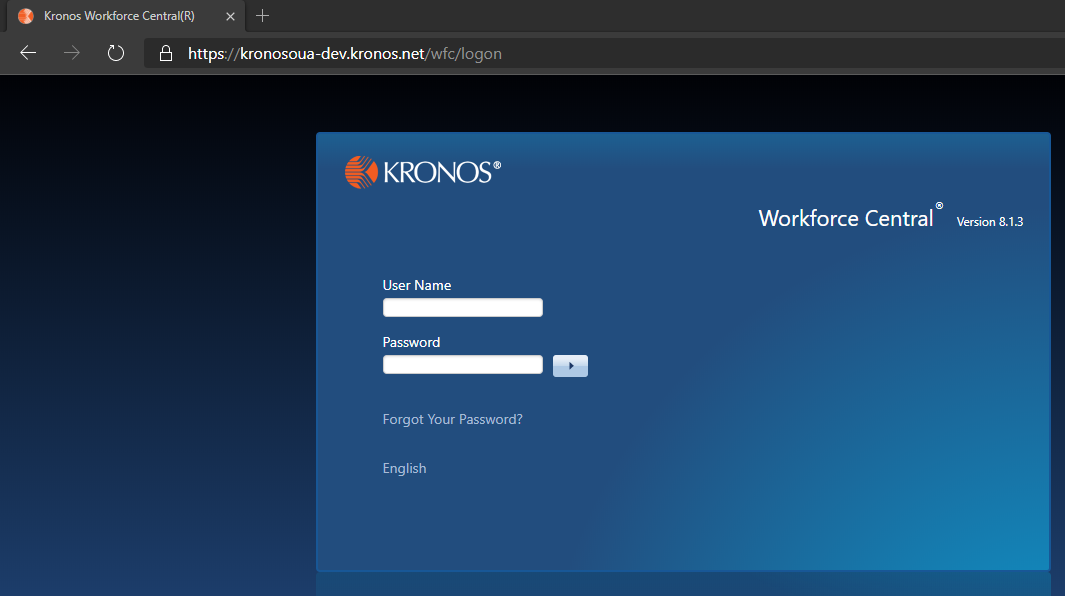
Table 9: SwapShiftMappingEntity

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| PartitionKey | String | Monthwise partition for storing Swap Shift Request details, it would be Month\_Year |
| RowKey | String | TeamsSwapShiftRequestId - The request id generated while submitting swap shift request from Shifts App |
| AadUserId | String | AAD object id of the user who has submitted Swap Shift Request |
| ShiftsStatus | String | Status of Swap Shift Request in Shifts App |
| KronosStatus | String | Status of Swap Shift Request in Kronos WFC |
| RequestedKronosPersonNumber | String | Kronos person number of the user corresponding to AAD of user to whom Swap Shift request is submitted in Teams |
| RequestorKronosPersonNumber | String | Kronos person number of the user corresponding to AAD of user who has submitted Swap Shift requested in Teams |
| KronosReqId | String | Kronos Swap Shift Request Id, generated once Swap Shift request created in Kronos |
| TeamsOfferedShiftId | String | Teams Shift Id for offered Shift |
| TeamsRequestedShiftId | String | Teams Shift Id for requested Shift |
| ShiftsTeamId | String | Teams Id in Shifts App |
| KronosUniqueIdForOffered Shift | String | Kronos Unique Id for offered Shift fetched from ShiftEntityMapping table |
| KronosUniqueIdForRequested Shift | String | Kronos Unique Id for requested Shift fetched from ShiftEntityMapping table |

# Time Off Reason – PayCode Configuration

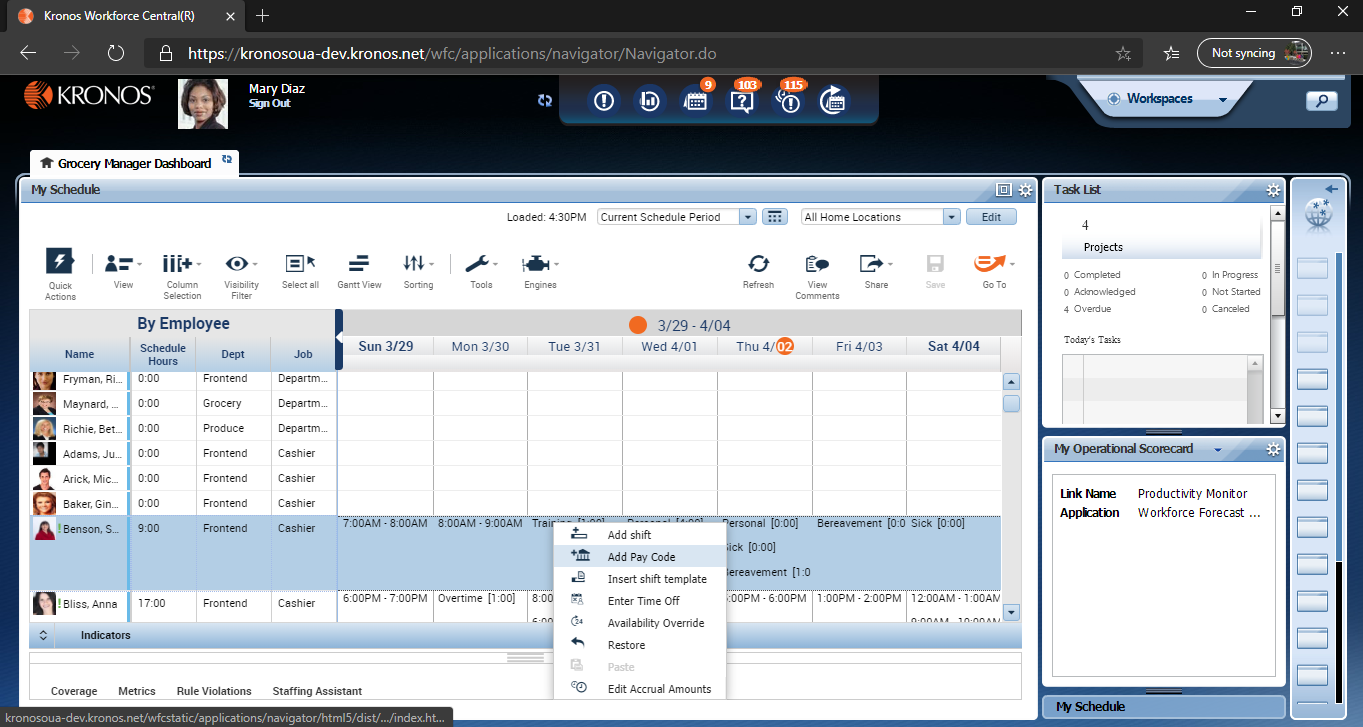
Follow the steps below to configure PayCodes in Kronos WFC.

1. Log into your on-premises version of Kronos WFC, as shown below:

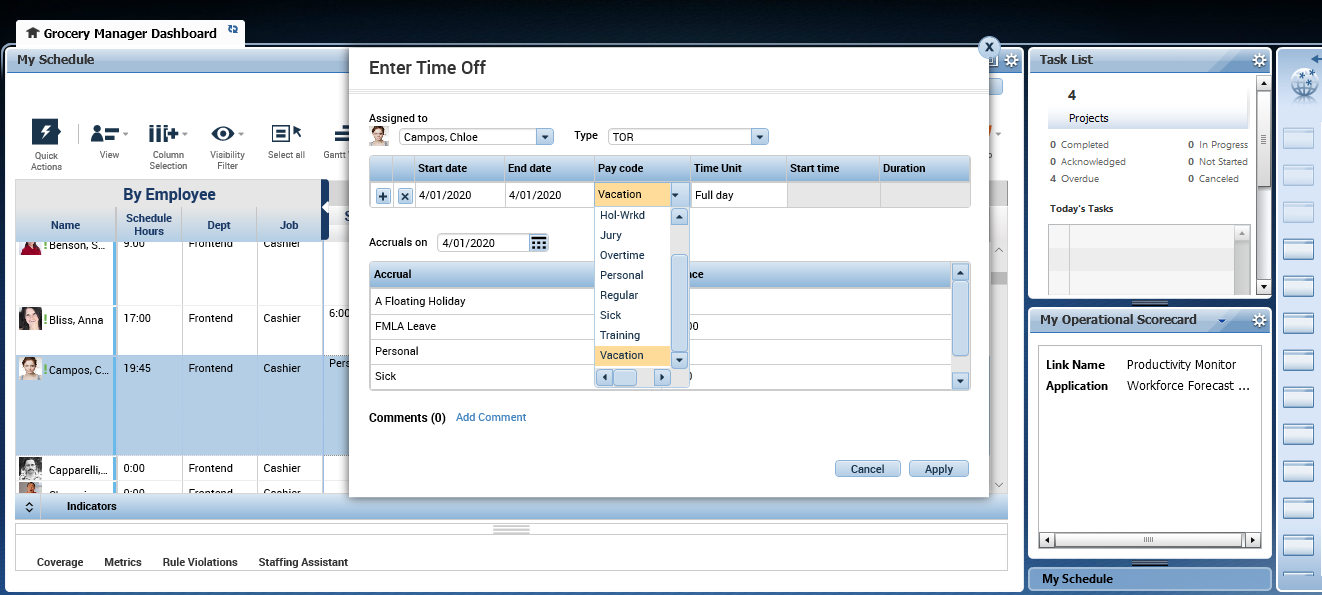


*Note*: The logon URL for your Kronos WFC on-premises instance will differ than what is shown in the screenshot above.

1. Once logged in with the appropriate SuperUser credentials, right click on any blank slot and click on “Enter Time Off”



1. On click of “Enter Time Off”, the following screen would be shown with the Kronos WFC PayCodes that are enabled



# Accessing the Kronos WFC PayCodes via API

To view the PayCodes via API, the Kronos WFC API URL – for example: <https://kronosoua-dev.kronos.net/wfc/webservice/WFCWebServices>. This endpoint as mentioned before will be different. The following steps should be followed to get the PayCodes through the Kronos WFC APIs.

1. Perform the Kronos WFC Login as defined in serial number 1 in the table under the section, *Kronos WFC APIs Description*.
2. Submit a request using the following XML:

<?xml version='1.0' encoding='UTF-8'?>

<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/'xmlns:hs='http://localhost/wfc/XMLAPISchema'>

<soapenv:Body>

<hs:KronosWFC>

<Kronos\_WFC version='1.0'>

<Request Action="LoadAllPayCodes">

<PayCode/>

</Request>

</Kronos\_WFC>

</hs:KronosWFC>

</soapenv:Body>

</soapenv:Envelope>

1. Once the request XML is submitted above, the following XML would be the response that would resemble something like below. Keep in mind, the response XML below is snipped for brevity.

<?xml version="1.0" encoding="UTF-8"?>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <soapenv:Body>

        <Kronos\_WFC TimeStamp="4/03/2020 3:49AM GMT-04:00" version="1.0" WFCVersion="8.1.3.1531">

            <Response Status="Success" Action="LoadAllPayCodes">

                <PayCode DisplayOrder="99999999" ExcuseAbsenceFlag="false" IsCombinedFlag="false" IsCurrencyFlag="false" IsVisibleFlag="false" ManagerAccessFlag="false" PayCodeName="1st Meal Premium" ProfessionalAccessFlag="true" AffectsAvailability="false" IsDaysFlag="false"/>

<!--Other Kronos PayCode elements-->

            </Response>

        </Kronos\_WFC>

    </soapenv:Body>

</soapenv:Envelope>

## Inserting a Time Off Reason into Shifts App

Adding a new time off reason is as simple as just creating the paycode in Kronos. We have a controller that is on charge of keeping paycodes in both systems in sync. When this controller runs its logic will detect any paycodes that have been added or removed in Kronos and will update Shifts as well as the mapping table automatically.

# Terminated and Inactive Users

The solution processes only active users. Each sync we retrieve all active users from Kronos, if a user in our mapping table is not found is the response of this request, we know that the user is either inactive or terminated in Kronos.

We then update the mapping table to show that this user is no longer active and exclude it from any further processing.

If the user is ever changed back to an active employee the tenant admin will be required to update the mapping table, setting the isActive column for the employee to true. As shown below.  
Graphical user interface, text, application

Description automatically generated

Setting this column to true will include this employee in processing as long as they are also an active employee in Kronos.

# Application Insights

Application Insights is used to capture the success and failure events, where following properties are captured:

* Class names
* Method names
* Parameters passed to methods
* Response from APIs – capturing necessary Id values.
* Timestamps (at the time when methods are called)
* Data properties (i.e. any IDs that are of interest to Graph API calls, or local data retrieval)
* Tenant ID – where applicable­