Allocatus

- Security and Trust -

Content

Revision & Sign-off Sheet

Change Record

Date	Author	Version	Change Reference
24.03.22	Renke Holert	1.9	Penetration Test
28.03.22	Maximilian Münst	1.10	Updated transferred project data section, task sync permissions
19.04.22	Maximilian Münst	1.11	Token management, API protection, data integrity
03.05.22	Maximilian Münst	1.12	Data integrity for timesheet
31.05.22	Renke Holert	1.13	Roles/Admin Portal
25.10.22	Maximilian Münst	1.14	Move service description and add data segregation section
22.02.23	Maximilian Münst	1.15	Update stored data and add availability information
15.01.24	Maximilian Münst	1.16	Update stored data information

Reviewers

Name	Version Approved	Position	Date
Maximilian Wagner	1.9	CISO	24.03.22
Maximilian Münst	1.14	Product Manager	25.10.22

Distribution

Name	Position

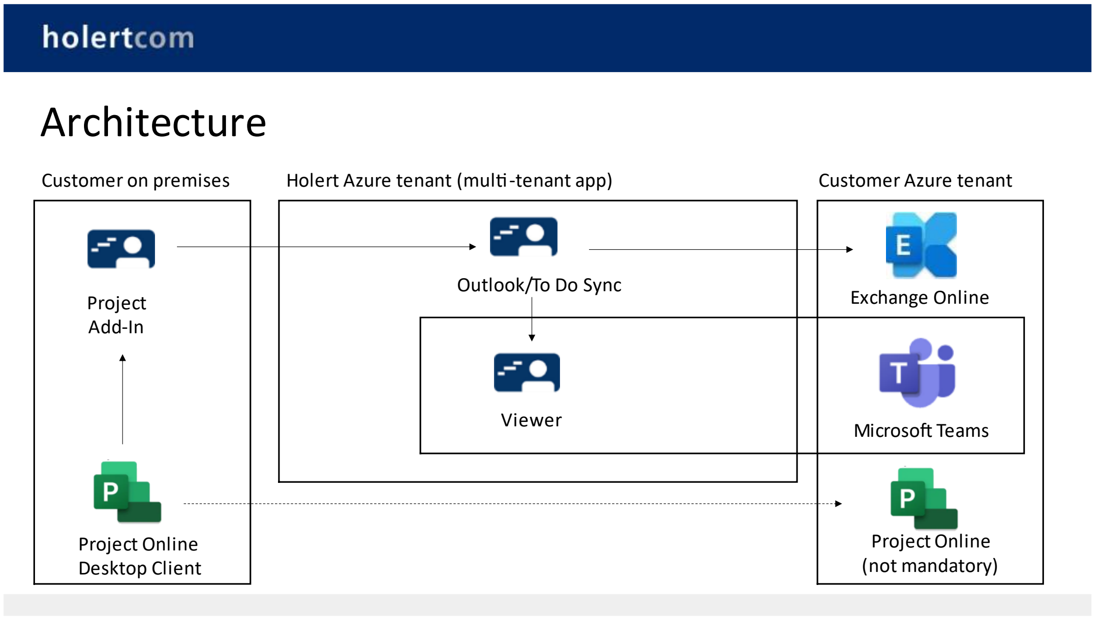
Overview and service description

Service description

The details on which features Allocatus has, is written down in the Allocatus service description, which can be found here: Allocatus service description

Architecture

We have followed the architectural guidelines from Microsoft for provider hosted solutions utilizing Microsoft 365/Azure environment.



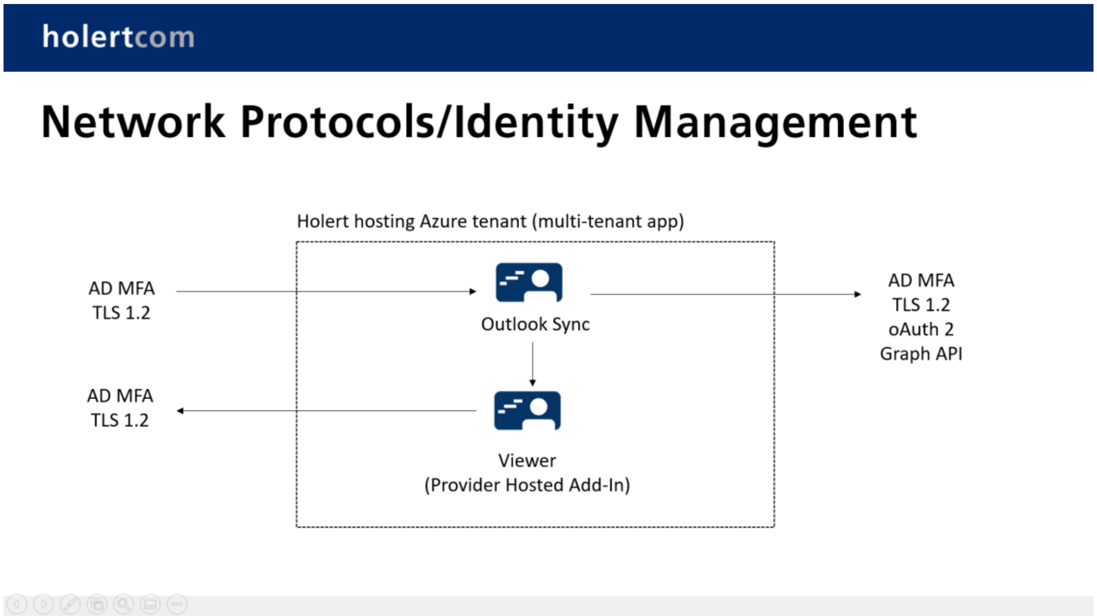
holertcom

Architecture

The Outlook calendar/task sync service and viewer are hosted in the Holert Azure tenant.

The project data is published through an Add-in in Microsoft Project. It connects to the API via a Microsoft Azure AD user account and sends the data to the dedicated customer database.

Then the service works on this data and connects via Microsoft Graph to the users' Outlook calendars or To Do lists. The data in the customer database is also used for the Allocatus Viewer and Teams App.



holertcom

Network Protocols/Identity Management

We enforce TLS 1.2 for the communication across the internet. All SSL connections are established using 2048-bit keys. We use Azure AD Microsoft Identity Platform alongside with OAuth 2.0 for authentication.

SLA

We strive to offer a great availability of our software. Because of that we have implemented Dev Ops practices so that only tested and working code is shipped to the customers.

The Allocatus Cloud infrastructure is based on Microsoft Azure, a leading provider of cloud services, and we can therefore refer to their standardized SLAs. Microsoft Azure offers a minimum availability guarantee of 99,9 % for their Azure services.

The following list contains examples of Azure services that Allocatus Cloud uses and their SLAs:

- SQL Database: 99.995%
- Kubernetes Service: 99.95%
- App Service: 99.95%
- Azure Functions: 99.95%
- Azure Storage: 99.99%

More information about each of the products can be found in the general Azure documentation by Microsoft: Azure Documentation

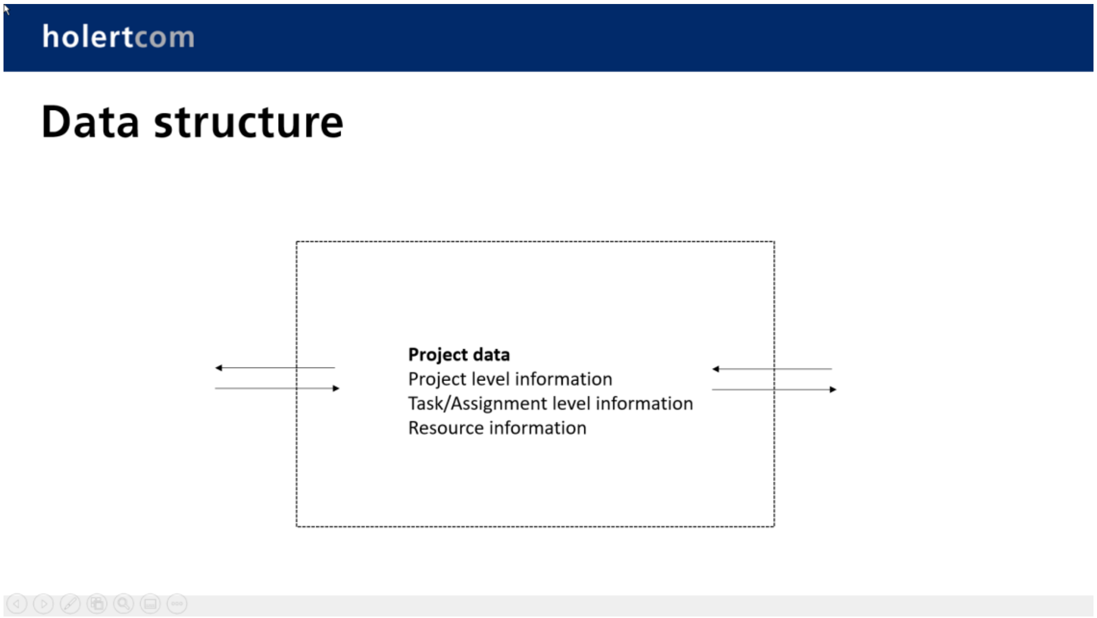
Stored Data

This section describes the data handled by the solution.

Data residency

All data is hosted within Microsoft Azure. All services of Allocatus are hosted in the West Europe or Germany West Central locations of Azure. The term location refers to the Microsoft Azure regions where your data or in-scope product content resides. Learn more about Microsoft Azure Regions.

Data structure



holertcom

Data structure

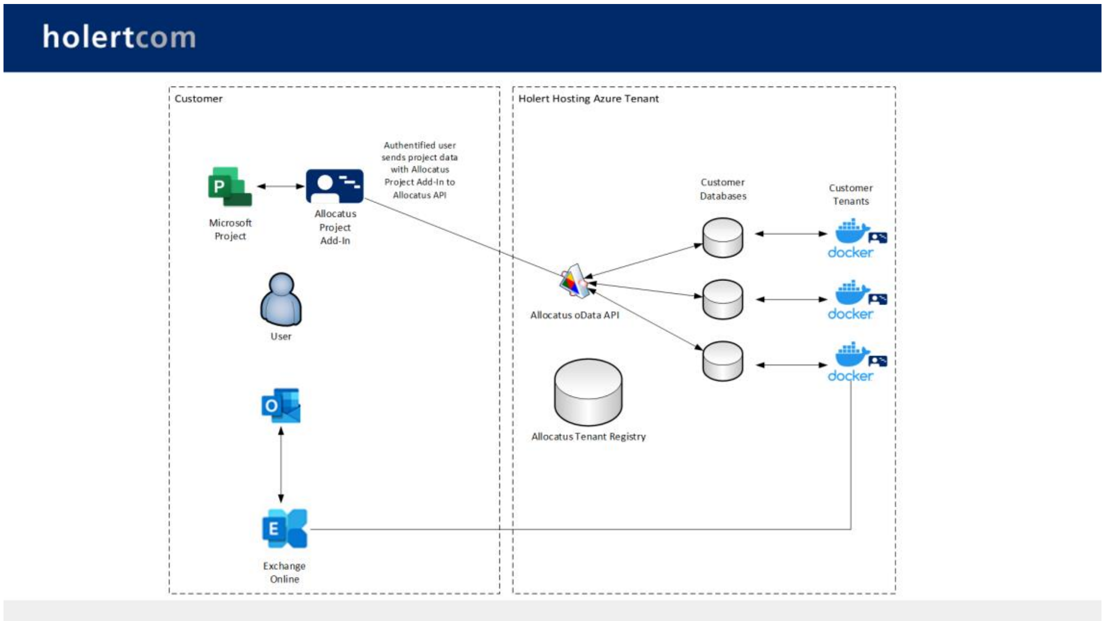
Project data
Project level information
Task/Assignment level information
Resource information

The solution manages data for project management. It includes information about projects including milestones and task alongside with assigned resources. Detailed information about the project data stored can be found in the appendix of this document.

Data segregation

Each customer tenant has its own database that only contains the own data. The Allocatus Sync Workers only work on one of the databases at a time so that no mixture between customer data can happen.

See the following architectural drawing to get an understanding of this:



Data integrity

General

Allocatus stores all its application data in Azure SQL Databases. Therefore, we rely on the advanced data integrity management by Microsoft e.g.:

- Extensive data integrity error alert monitoring
- Backup and restore integrity checks
- I/O system *lost write* detection
- Automatic Page Repair
- Data integrity at rest and in transit

More information can be found here: [Data Integrity in Azure SQL Database]

We have built into Allocatus protection mechanisms to prevent data corruption from happening, as well as systems and processes that enable us to recover data if it does. Those are in place at various stages of the engineering release process, including:

- System Design
- Code organization and structure
- Code reviews
- Historization of data

Allocatus Data

As Allocatus can be used to track work time for projects, which can be crucial for projects, the traceability and integrity of the time tracking data is necessary. Part of the above-mentioned mechanisms is to track all changes of time bookings in an history table. This table contains timestamps and all changed fields of a time tracking entry. Every change that is done is tracked there. By this measure it can be retraced at every time why a time tracking entry has changed.

Also, the SQL Server transaction log can be used to revert unwanted changes.

The time tracking process also contains an approval of the booked times. By this every work time entry that is booked by users must be approved by a project manager first. This ensures that no unwanted or wrong bookings are written back into MS Project.

Logging

To ensure reliable operations the solution also writes personal data to its log-files. You can configure it to only log technical GUIDs instead of username, email, project names, and task names. Logs are deleted after 90 days.

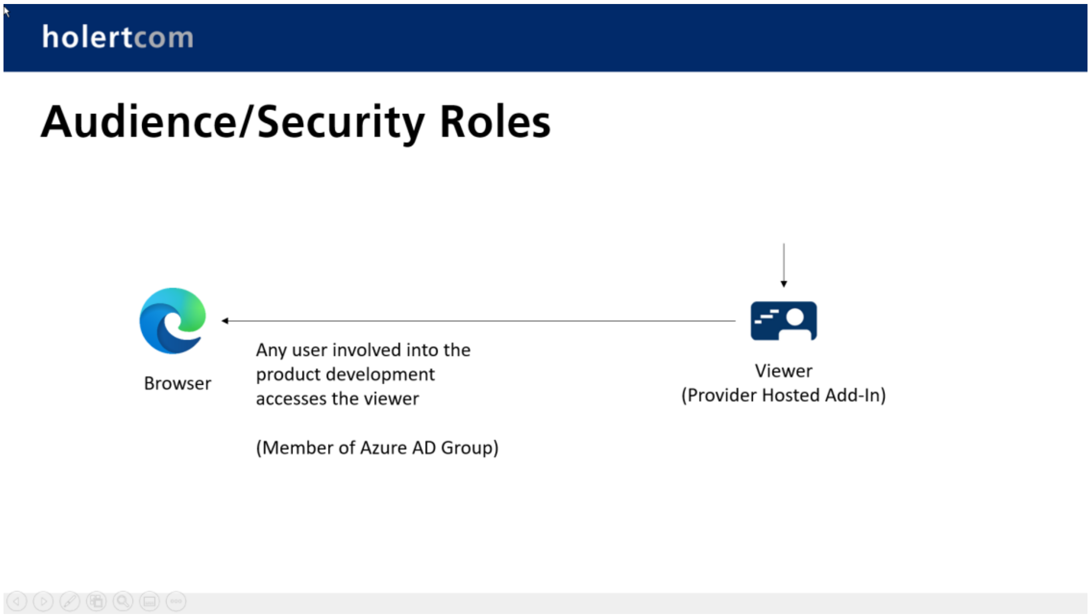
Access rights/Audiences

Two different audiences will use the solution: the project managers and all other people involved in the project. The project managers are the only group that uses Microsoft Project and the Project Add-in. The other users use the Outlook calendar or task sync and the viewer.

Project Add-in

The Add-in for Microsoft Project has the same access rights on the file system as the user who is running it. This is only used to read the data listed above from the project schedule and write log-files. This Add-in is only installed on the computers of the approx. 10 users involved in the project creation process.

Viewer



holertcom

Audience/Security Roles

Browser

Any user involved into the product development accesses the viewer

(Member of Azure AD Group)

Viewer
(Provider Hosted Add-In)

The viewer is by default accessible for all Allocatus users. You can change this setting, so that it is accessible for all users that belong to a given Azure AD group or to your Azure AD tenant.

The viewer uses the same Microsoft Graph access rights as the Calendar sync application

Calendar sync

The Outlook calendar sync utilizes the Graph API to read and write the above-mentioned data in Exchange. Allocatus needs to be granted the following **delegated** permissions:

- User.Read: Sign in and read user profile
- User.ReadBasic.All: Read all users' basic profiles
- Offline_access: Maintain access to your data you have given it access to
- Openid: Sign users in
- Calendars.ReadWrite: Have full access to user calendars
- MailboxSettings.ReadWrite: Read and write user mailbox settings

Task sync

The task sync utilizes the Graph API to read and write the above-mentioned data in Microsoft To-Do Tasks. Allocatus needs to be granted the following **delegated** permissions:

- User.Read: Sign in and read user profile
- MailboxSettings.Read: Read user mailbox settings
- User.ReadBasic.All: Read all users' basic profiles
- Offline_access: Maintain access to your data you have given it access to
- Openid: Sign users in
- Tasks.ReadWrite.Shared: Read and write user and shared tasks

Allocatus Teams Interface

The *Allocatus Teams Interface* is an add-on to Allocatus that enables automatic creation of a Microsoft Teams Team for every project. This needs the following **application** permissions:

- User.Read.All: Read all user’s full profiles
- ~~Group.ReadWrite.All: Read and write all groups~~
- TeamMember.ReadWrite.All: Add and remove members from all teams
- Team.Create: Create teams
- Team.ReadBasic.All: Get a list of all teams
- TeamSettings.ReadWrite.All: Read and change all teams’ settings
- TeamsTab.Create: Create tabs in Microsoft Teams
- TeamsTab.ReadWrite.All: Read and write tabs in Microsoft Teams

- ChannelSettings.ReadWrite.All: Read and write the names, descriptions, and settings of all channels
- ChannelMember.ReadWrite.All: Add and remove members from all channels
- Channel.ReadBasic.All: Read the names and descriptions of all channels
- Channel.Delete.All: Delete channels
- Channel.Create: Create channels

To grant admin consent for the app click the following link:
https://login.microsoftonline.com/common/adminconsent?client_id=cdd4a1cd-c268-4a39-9534-2fee4ed10db3&redirect_uri=https://webapp.cloud.allocatus.com

Microsoft Graph access and refresh tokens

As Allocatus uses delegated permissions for the above-mentioned applications, a user must be signed in to use the functionality of Graph API.

The Graph access tokens for the above-mentioned applications are obtained and refreshed when needed by the Microsoft authentication endpoints.

The Allocatus sync service stores the refresh token for each user encrypted in the database. This is needed for the service to run in the background.

Security

Security is of high value to us. Therefore, we are in the process of getting Allocatus certified under ISO 27001. We currently have no reliable timeline, but we aim to complete the certification until end of 2024.

Of course, we are already developing our software utilizing state of the art best practices. Access to the development system is granted using multi-factor authentication and single sign-on. We use encryption and only store credentials in key vaults. We are committed to use applicable standards as security frameworks.

Role Model

We have currently three roles: Allocatus Admin, Allocatus User and Allocatus Viewer. Admin is only granted on request. Users can be requested by any user who is user. Viewer are managed through an associated Microsoft 365 group.

Admin Access

There are two kinds of Admin Consoles: Settings Portal is for the Allocatus Admin to configure things like views, etc. The Admin Portal used by us internally to administer all customer tenants. Both consoles are publicly exposed and secured with Azure AD. The access to the Settings Portal is dependent on the customer’s

Azure AD configuration. The Admin Portal is dependent on our Azure configuration. We are using MFA to secure it.

Database security

The stored data is encrypted at rest. All the data is stored in Microsoft Azure SQL databases that use transparent data encryption (TDE) by Microsoft. The key for TDE is managed by Microsoft. More information about it can be found here: https://docs.microsoft.com/en-us/azure/azure-sql/database/transparent-data-encryption-tde-overview. The project data for different customers is stored in separated databases.

Microsoft Azure also has a strong focus on compliance. Their security measures and certifications are listed here: https://azure.microsoft.com/en-us/overview/trusted-cloud/compliance/

Also, database connections are stored in key vaults which only the Allocatus API and the sync worker has access to. Read more on this here: https://docs.microsoft.com/de-de/azure/key-vault/general/basic-concepts

Allocatus API security

Allocatus uses an OData API which is available under the following url: https://clientapi.cloud.allocatus.com

Documentation on how to use the API can be found here: https://documenter.getpostman.com/view/16173579/TzeWGnne

Each user must sign in with their Microsoft account to be able to send or receive data from the Allocatus API. This means the authentication is handled the same way as it is for Microsoft 365 and many other apps by Microsoft or third-party ISVs. More information on this can be found here: https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-overview

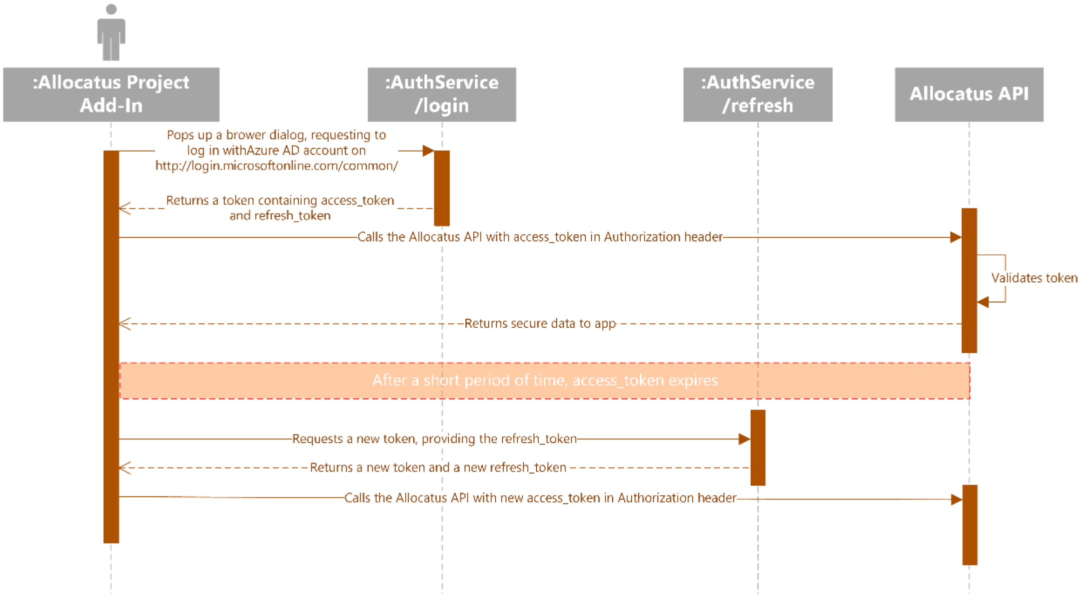
This means the API is secured by bearer authentication, also called token authentication. The bearer token is a cryptic string generated by the Allocatus Auth service in response to a login request. It contains a refresh token and an access token.

The client (Allocatus Add-In or other custom developed solution) must send the access token in the Authorization header when making request to the Allocatus API.

The access tokens have a lifetime of ~60 minutes. Afterwards the token must be refreshed by the Allocatus Auth service with the refresh token. The lifetime of a refresh token is 30 days after that a new login request must take place.

All the data is transferred by using HTTPS with TLS 1.2.

You can see the whole flow in the sequence diagram below:


:Allocatus Project Add-In
:AuthService /login
:AuthService /refresh
Allocatus API
Pops up a brower dialog, requesting to log in withAzure AD account on http://login.microsoftonline.com/common/
Returns a token containing access_token and refresh_token
Calls the Allocatus API with access_token in Authorization header
Validates token
Returns secure data to app
After a short period of time, access_token expires
Requests a new token, providing the refresh_token
Returns a new token and a new refresh_token
Calls the Allocatus API with new access_token in Authorization header

Appendix

Penetration Tests

Passed in January 2022 by binsec GmbH (Version 1.2 from Jan 31)

Project data transferred by the Project Add-In

Project-level information

- Project name
- Project owner’s name incl. their windows account and email address
- Notes
- Enterprise Custom fields
- Date & time of last publish to Allocatus.
- Technical GUID
- File location or PWA-URL
- Start, finish date
- Project last published date and published by

Task-level information

- Task name
- Task start & finish date
- Technical GUID

- Task duration
- Deadline
- IsMilestone
- IsMarked
- IsManuallyScheduled
- Estimated
- Priority
- Active
- IsPublished
- Work, actual work, remaining work
- Created date
- Notes
- Enterprise Custom fields
- IsSummary
- Task outline information
- Successor, predecessor
- Percent complete
- Cost, Actual cost, Remaining cost
- HyperlinkAddress
- TaskResourceNames

Assignment-level information (+assignment by day)

- Technical GUID
- Technical GUIDs of related task and resource
- Assignment start & finish date
- Work, actual work, remaining work
- Enterprise Custom fields

Resource information

- Resource & project technical GUID
- Resource name
- Resource email address
- Resources' windows accounts
- Notes
- Enterprise Custom fields