Whitepaper

A universal way to interact with data stored in third-party services

In this document, we explore the specialized infrastructure designed specifically to access resources hosted by many independent third-party repositories.

Deep Authorization Management

File services that span across multiple data sources present unique challenges for app developers. Solutions require specialized infrastructure designed specifically to access resources hosted by many independent third-party repositories.

The Cloudtenna global-name-space architecture provides a platform to interact with data stored in third-party services. It delivers a single set of powerful APIs to perform file operations, collaboration, search, and analytics across a dispersed data set.

Purpose:

Cloudtenna creates a metadata map of disparate storage repositories, providing a normalized construct of all data in one place. In order to read data and perform file operations on data living inside a third-party repository, Cloudtenna must obtain authorization from each app to perform actions on behalf of each user.

Authentication and *authorization* are often confused. Authentication grants a user access to their data. Authorization grants an application the right to perform an action on behalf of a user. In this document, we will explore what is required to augment a traditional identity management solution designed for authentication and update it to also obtain authorizations on behalf of users.

Background:

We are experiencing a shift from an application-centric storage model to a data-centric storage model. Employees now use an average of 6-8 different apps and repositories as part of their standard daily workflow. Email, file sync & share services, CRMs, and other productivity apps all serve a valuable purpose.

Each data source has its own proprietary protocols and APIs. Files are the common language between all of the independent repositories. For certain workflows, it makes more sense to work with a platform that has access to all files rather than build a microservice that attaches to only one application or repository. With files scattered across silos, a data-centric approach provides a collective view to all data within the enterprise – regardless of where that data is stored.

What is the difference between authentication and authorization?

The number of apps and services used by an enterprise has skyrocketed in recent years. Users are forced to remember a lot of distinct username and passwords, and IT administrators are forced to manage multiple user accounts for each employee, one for each service.
Identity management solutions with single sign-on features were introduced to curb this chaos. With a SSO solution, a user only needs to login once, and they can then launch their own pre-authenticated user session for any supported third-party application. Similarly, IT administrators

can centrally manage multiple applications such that they no longer have to manually deactivate twelve independent user accounts when an employee is fired. A single sign-on service manages user sessions for third-party applications, but is blind to the data inside of each of them. A single sign-on service, alone, is not positioned to consume each application's APIs.

Authorization adds an additional level of access management on top of the SSO authentication services above. API-consuming services like Cloudtenna are granted rights to act on behalf of a user. This differs from a user session. First, at any point in time these rights can be revoked from the API-consumer. Second, these rights can be limited in scope and set not to provide the full broad access allotted by a full user session. And third, these rights do not require the user to be present – important for persistent background processes like file indexing.

Traditional identity management solutions are designed to initiate user sessions for SSO purposes, but do not have any context for how a service like Cloudtenna would request API scopes to act on behalf of a user.

Cloudtenna is designed to complement traditional identity providers. The identity provider continues to manage authentication. Cloudtenna, via the pre-existing identity provider, is then responsible for obtaining, storing, managing, and refreshing the authorization necessary to act on behalf of the user for each app.

Why does a data-centric platform require API authorization?

Data-centric platforms like Cloudtenna consume proprietary third-party file storage APIs from disparate services and deliver a single normalized set of file services APIs. Cloudtenna essentially hides the differences between the many different file storage protocols and proprietary file storage APIs.

Underneath the hood, Cloudtenna requires API authorization in order to fulfill two core responsibilities: (1) maintaining the global namespace and (2) performing file operations on the source repositories.

Maintaining the global namespace:

To generate the global namespace, Cloudtenna indexes content from each data source and normalizes the collected metadata into a single abstraction. As employees update files on the source repositories, it is the responsibility of the Cloudtenna crawler to recognize the content delta and keep the global namespace up-to-date accordingly. The crawler must be authorized by the owner of the source repository to enumerate file changes in order to detect new updates. Cloudtenna performs this indexing process in the background, 24/7, in order to reflect the latest changes in the global namespace as quickly as possible. Authorization grants Cloudtenna the rights needed from the source repository to build the global namespace.

Performing file operations on the source repositories:

Cloudtenna offers a single set of comprehensive file services APIs that facilitate file operations, collaboration, and other analytics on disparate file repositories. Each file repository uses its own protocols and proprietary APIs, but Cloudtenna hides this complexity with a single set of APIs that seamlessly operate across all repositories. Underneath, Cloudtenna translates each API request appropriately and performs that desired action on behalf of the user.

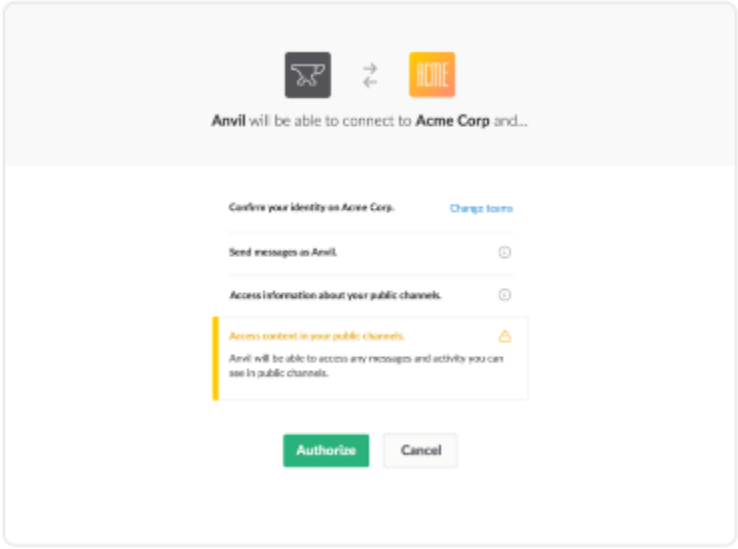
Authorization grants Cloudtenna the rights needed from the source repository to perform file operations like download, upload, and rename. Without this authorization, an application would be forced to create independent sessions via each repository's own proprietary APIs. The management of all of these user sessions alone is a challenge. An application which wants to work across a disparate data set would have to obtain user sessions from each and every repository, for each and every user, on each and every one of their devices. The application would then have to manually weave together complex APIs from all of the different storage environments. All of this aggravation can be avoided with Cloudtenna. Cloudtenna handles all of the complex differences between API paradigms and delivers a single set of APIs that are authorized to perform actions on behalf of the user.

Deep dive:

Obtaining authorization:

Users grant authorization for Cloudtenna to access their data or act on their behalf. This is a common and standard process – the same as, for instance when a user grants Slack the right to launch a Google Hangout from within Slack. The most common type of authorization is via an OAuth token, and Cloudtenna also supports SAML and basic authentication systems as well. The OAuth 2.0 protocol is a delegated authorization mechanism, where an application requests access to resources controlled by the user (the resource owner) and hosted by an API (the resource server). The authorization server issues the application a more restricted set of credentials than those of the user in the form of an Access Token.

When a user initiates a process that requires authorization for the first time, the third-party repository presents a prompt to start the authorization flow. Authorization is limited to only the scope of rights required by the requesting app. A user then approves the authorization, granting the requesting app the right to act on behalf of the user within the scopes allowed.



The permissions represented by the Access Token in OAuth 2.0 terms are known as `scopes`. The `scope` parameter allows the application to express the desired scope of the access request. Once a user has approved authorization, the authorization tokens are securely stored and appropriately refreshed such that the application can continue to access the target API and act on behalf of the user.

Case Study:
Notify users when new files are added to a folder

A Cloudtenna partner was looking for a way to notify users when content had been added to a folder inside a third-party repository. Specifically, their customers saved invoices (PDF) to a Microsoft OneDrive folder, and the partner wanted to deliver an automated workflow that would notify key stakeholders whenever an invoice was added.

An automated notification would greatly reduce the time and effort required to inform other collaborators that a new invoice has been posted. Prior, the person who created the invoice would have to manually contact other teammates to let them know that invoice had been posted. The result was that teammates often would not see the latest document until they manually browsed into the enclosing folder and stumbled upon the new invoice themselves.

The Cloudtenna partner offers a software suite for project management. Users create tickets with rich metadata including descriptions, milestones, comments, key stakeholders, and timelines. Updates to any of this metadata appear in a personalized newsfeed for each user with updates associated to any subscribed tickets. The partner wanted to also present a post in each user's newsfeed when a file associated with the ticket had been updated.

The challenge is that the files are hosted by a third party, in this case Microsoft OneDrive. For the Cloudtenna partner to recognize new files, it would have to call the Microsoft OneDrive APIs on behalf of the user. Without proper authorization, this requires the user to login with their OneDrive account to create a user session every time. The value of notifications, though, would be to present updates to the user proactively – without user intervention. The partner needed a way to access content in Microsoft OneDrive so it could continuously check for new files in the background without the user having to manually authenticate themselves.

The partner looked to Cloudtenna for the opportunity to gain app-to-app persistent authorization so it could enumerate `watched` folders in the background without user intervention. Cloudtenna obtains, stores, manages, and refreshes API scopes to third-party repositories like Microsoft OneDrive. With this persistent API authorization, Cloudtenna can periodically check those folders on behalf of the user and can send a notification through the partner to the user when a new file is detected.

Before Cloudtenna can check for new files, the user must complete a one-time authorization process that grants Cloudtenna the right to enumerate the folder on behalf on the user. This is typically initiated by a standard OAuth login process. To streamline this authorization request further, the Cloudtenna partner leveraged it's pre-existing identity provider platform which includes single sign-on authentication features. Via SSO, a user is authenticated automatically can more easily provide the authorization necessary for Cloudtenna to operate persistently in the background.

Together with Cloudtenna, the partner is able to provide a solution that notifies its user whenever files in watched folders has been updated. It does this by periodically enumerating the folder in the background and cross referencing for new content. In order to access the APIs necessary to enumerate the directory, Cloudtenna works in conjunction with the partner's identity provider service to obtain, store, manage, and refresh an API authorization token that is granted the necessary API scopes.

The net result for the user is a seamless newsfeed that presents notifications from both (1) when the metadata for a ticket has been updated and (2) when a file hosted in a third-party repository has been updated.

Conclusion:

File services that span across multiple data sources present unique challenges for app developers. Solutions require specialized infrastructure designed specifically to access resources hosted by many independent third-party repositories.

The Cloudtenna global-name-space architecture provides a platform to interact with data stored in third-party services. It delivers a single set of powerful APIs to perform file operations, collaboration, search, and analytics across a dispersed data set. In order to read data and

perform file operations on data living inside a third-party repository, Cloudtenna obtains authorization from each app to perform actions on behalf of each user. In this document, we explored what is required to augment a traditional identity management solution designed for authentication and update it to also obtain these API authorizations on behalf of users.