Salesforce Email Integration Security Guide

Salesforce, Spring '20

@salesforcedocs

Last updated: March 26, 2020

CONTENTS

SECURITY GUIDE OVERVIEW

Integrate the world's best CRM with your email and calendar. This document covers technical guidelines for desktop solutions that are provided with a Sales Cloud license. This document also covers guidelines for desktop and mobile solutions provided with an Inbox license.

GET TO KNOW THE SALESFORCE EMAIL INTEGRATION PORTFOLIO

These Salesforce options to integrate with email applications are covered in this document.

- Outlook integration, and Outlook integration with Inbox
- Gmail integration, and Gmail integration with Inbox
- Salesforce Inbox mobile apps for iOS (available in the App Store) and Android (available in the Google Play Store)

Sync contacts and calendar events between Salesforce and Outlook and Gmail using Einstein Activity Capture or Lightning Sync. Set up automated email and event logging using Einstein Activity Capture. For a comparison of the two products, see Salesforce Help. For security considerations, see the Einstein Activity Capture Security Guide and the Lightning Sync Design and Security Guide.

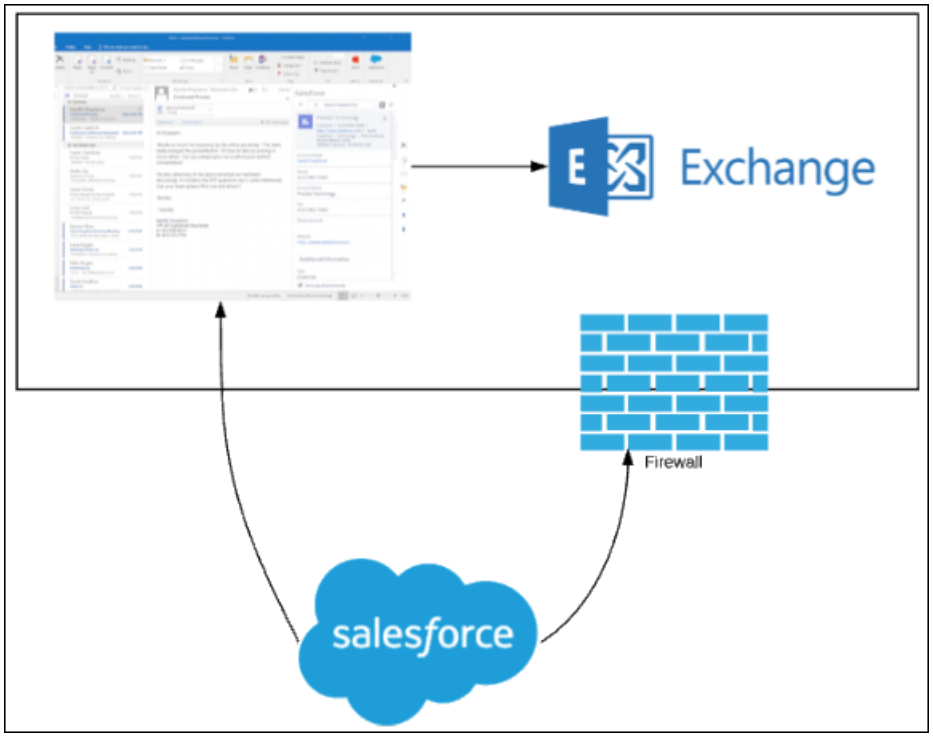
OUTLOOK INTEGRATION

Setting up the Outlook integration requires access to your Exchange server. How you choose to set up that access depends on the versions of Outlook you use, your internal security policies, and the features that sales reps need within the integration.

Background

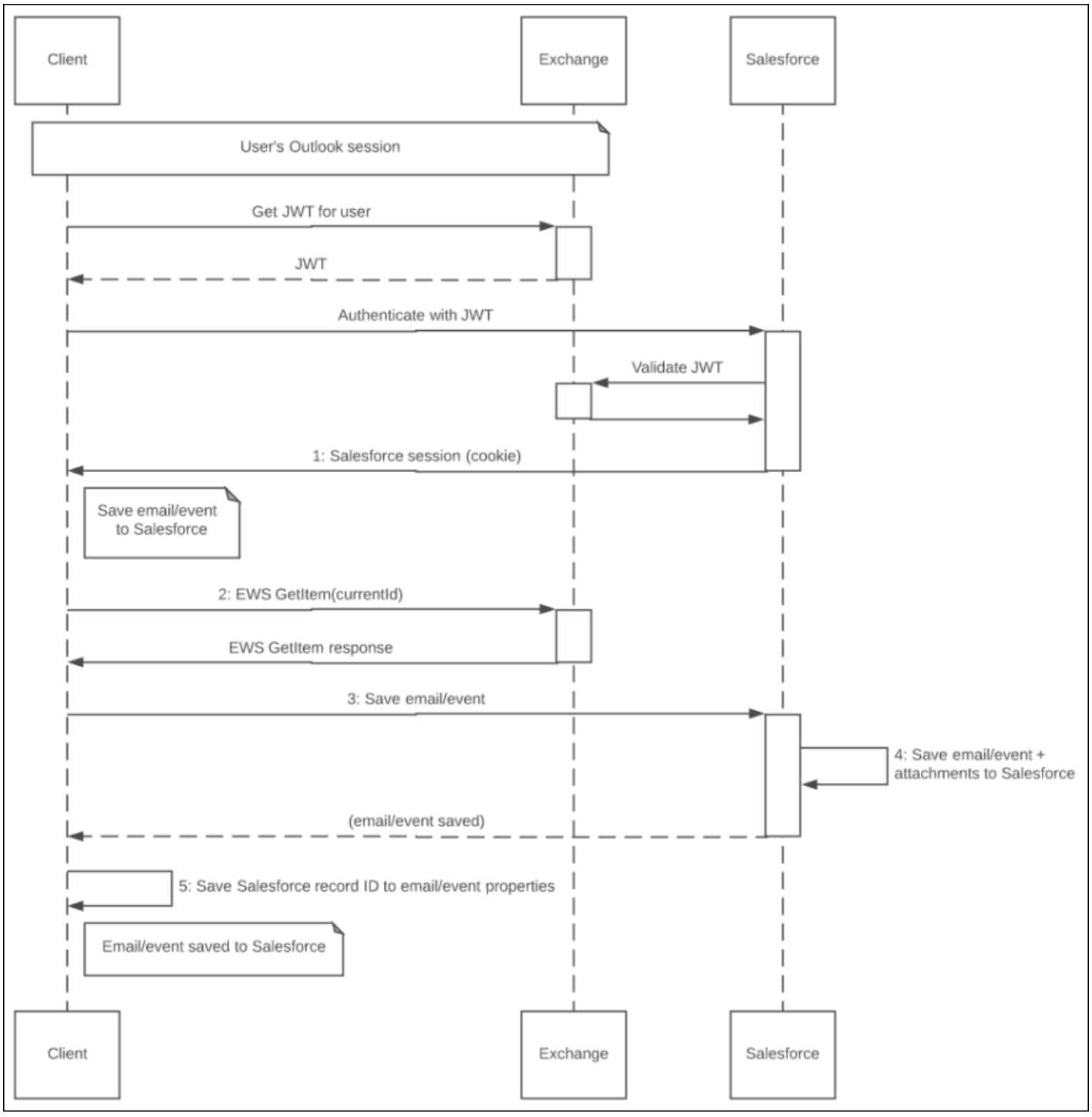
The Outlook integration add-in is built on the Microsoft Office Add-In Framework. To log emails from Outlook to Salesforce (among other end-user actions) within that framework, Salesforce is required to make calls to the Exchange server.

In a typical Exchange on-premises setup, a firewall blocks access from the internet.



Historically, the Outlook integration tapped into the Exchange API and placed Exchange Web Services (EWS) calls from Salesforce application servers. The add-in calls were placed with an Exchange-provided JSON Web Token (JWT) at the URL provided by Exchange itself, via EWS. The JWT calls required an exposed EWS endpoint.

Microsoft enhanced the Outlook API, and the historic EWS server calls are now client calls in the Office.js API that Outlook provides, as shown in this diagram.



The JavaScript API v1.8, available in recent versions of Outlook and in Exchange Online, means fewer requirements to integrate Salesforce with Outlook. With the correct versions of Outlook and Exchange, there's no need to expose an EWS endpoint to power the features in the Outlook integration.

The latest builds of Exchange Online run JavaScript API v1.8, or later. To determine if your Outlook client runs the JavaScript API v1.8 or later, see Outlook JavaScript API requirement sets. To take advantage of the simplified configuration, it's up to you to give access to

Outlook applications running JavaScript API v1.8 to your users. If the Exchange server and Outlook run different versions of the JavaScript API, the earlier API is used.

Important: If the Outlook version is running JavaScript API v1.7 or earlier, or your Exchange server is on-premises, the Outlook integration reverts to placing EWS calls. In that scenario, an EWS endpoint is required to access all the features in the integration. Without the EWS endpoint, integration users can't log attachments from the integration.

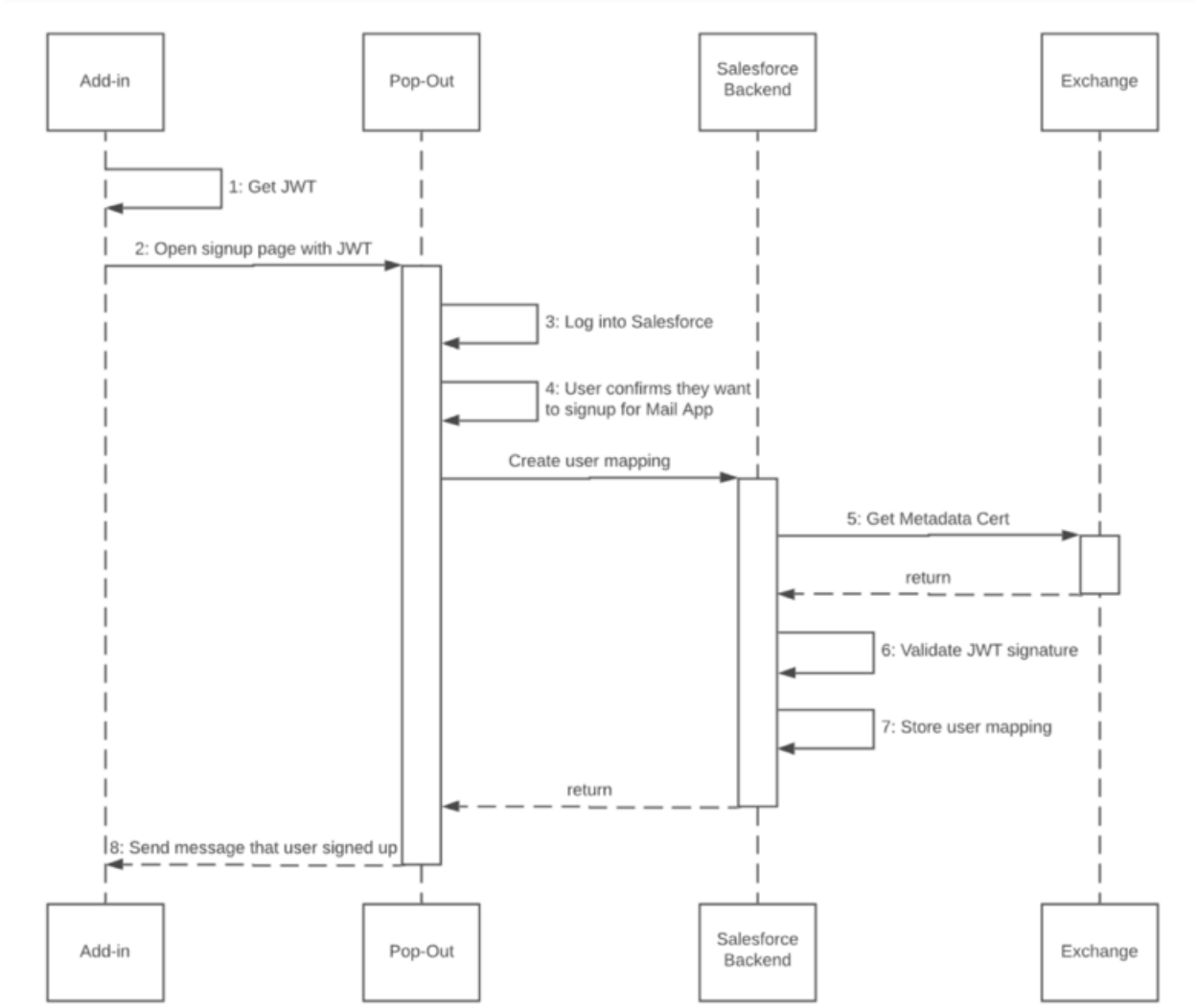
Features available with an Inbox license, such as insert availability and sent later, require access to the Exchange server, regardless of the Outlook API version. If you have an Inbox license, review Outlook Integration with an EWS Endpoint on page 7 and Outlook and Gmail Integrations with an Inbox License on page 11.

First-Time User Authentication Login Flow

Salesforce connects to Exchange to authenticate a user via the metadata URL and is a separate consideration from EWS. This diagram details the flow for how Exchange is mapped to the corresponding Salesforce user the first time the user loads the Outlook integration add-in.

This flow applies to all versions of Outlook and Exchange, regardless of the JavaScript API version.

This diagram details the flow for how the Exchange mail is mapped to the corresponding Salesforce user the first time they load the Outlook integration add-in.



1. The Outlook add-in retrieves an identity token with a simple JavaScript method:

   ```
   Office.context.mailbox.getUserIdentityTokenAsync(callback, userContext);
   ```

   The JavaScript method requests an Exchange user identity token (a JSON Web Token or JWT) from the Exchange server. The add-in opens the sign-up page in a popup window hosted on Salesforce.

2. The user authenticates with their Salesforce credentials.
3. Salesforce prompts the user to connect their Exchange account (specified in the identity token) with the authenticated Salesforce user.
4. The user clicks the prompt, confirming they want to sign in.
5. Salesforce serves then validates the Exchange token contents and fetches the public certificate of the metadata URL. Salesforce expects the EWS endpoint to have a valid certificate. See Salesforce Help for information about supported SSL certificates.
6. Salesforce validates the identity token signature by accessing the public signing key from the authentication metadata document on the Exchange server.

   When the Exchange server initially provides the JSON Token to the add-in, it specifies the following:

   - An Exchange Metadata Endpoint URL inside the payload part of the token itself

- The Salesforce add-in

The add-in sends a request to the defined metadata URL to validate the signature. The Exchange metadata URL must be publicly accessible for validation of the user's identity token.

To learn more about validating a token, see Microsoft documentation.

7. The Exchange to Salesforce user mapping is then stored within the user's Salesforce org data.

Outlook Integration with an EWS Endpoint

This section covers the authenticated calls that the Outlook integration add-in uses in the following scenarios.

- Outlook versions are running JavaScript API 1.7 or earlier. Check which version of the API your Outlook application runs in Outlook JavaScript API requirement sets.
- Implementations are using Exchange on-premises.
- You've added an Inbox license, which enables features including insert availability, sent later, text shortcuts, and email tracking. These features require access to the Exchange server. Also review Outlook and Gmail Integrations with an Inbox License on page 11 in this guide. That section includes security and implementation considerations beyond what is discussed in this section.

Important: Without the EWS endpoint in these scenarios, integration users can't log attachments from the integration or use any Inbox productivity features.

Configuration Requirements

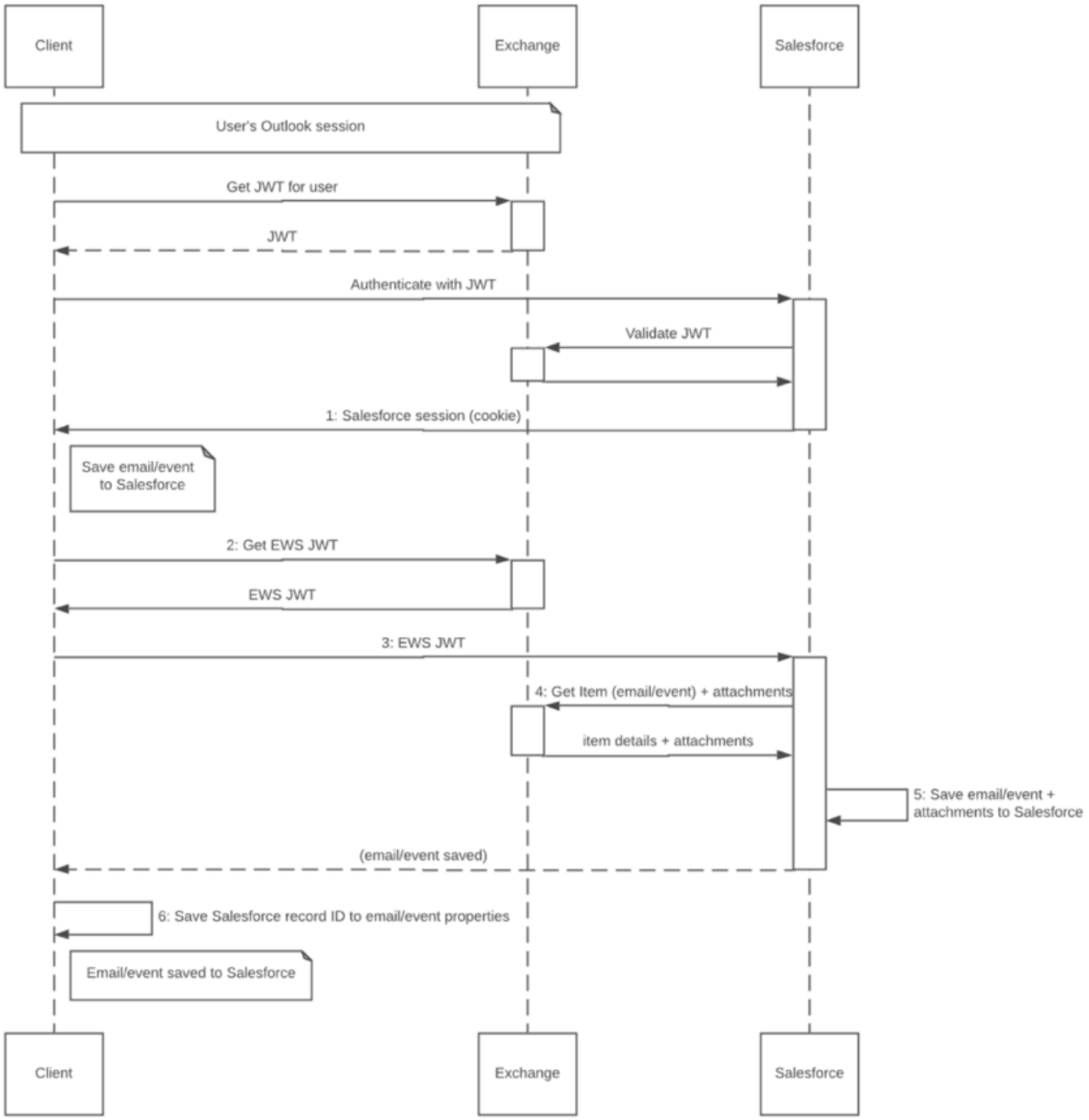
Configuring the Outlook integration requires the public exposure of URLs.

- Exchange metadata URL that permits unauthenticated HTTP access. See the First-Time User Authentication Login Flow on page 5
- Exchange Web Service URL

Because Salesforce makes outgoing calls to these Exchange endpoints, they must each have a valid SSL certificate supported by Salesforce. Specify any custom Outlook Web App (OWA) URLs, such as non-Office 365 URLs, in the Outlook integration settings in Salesforce setup. Ensure that the custom URL is whitelisted in the Outlook integration settings in the Salesforce setup. Custom URLs don't require public exposure because only the client browser needs access to OWA. These settings apply only when using OWA.

Logging Emails with Attachments to Salesforce Flow

From the Outlook integration, users can manually log a selected Outlook email message and its attachments to Salesforce. The add-in uses the following flow to complete the logging:



1. Authenticates with Salesforce (see Login flow) for details.
2. Makes an authenticated call to Exchange Web Services (EWS) via the API provided to Outlook add-ins. See Microsoft Office API documentation. Salesforce servers are now allowed to fetch the current email or event to be logged.
3. Performs the EWS operations EWS GetItem + GetAttachment(s) for the current email or event and its attachments.
4. Saves the email or event and the attachments to Salesforce and associates both to the selected Salesforce records.
5. Modifies the email or event in Exchange to include the Salesforce record ID in the extended properties of the Exchange object.

APIs Used

We make client-side API calls via Office.js and server EWS calls, limited to GetItem and GetAttachment operations. The EWS calls that we make are initiated from the client side and from the Salesforce app servers. A user action triggers these calls in the context of a particular email or event. The calls coming from the Salesforce app servers to your EWS URL come from the published IP address ranges.

The Outlook integration specifies ReadWriteMailbox so that it can read the email or event and its attachments. The Write access is to write the Salesforce task or event ID back to the Exchange record via an EWS call placed through the Office.js API. See the Office.js documentation for details about the configuration requirements for making this EWS call.

Exchange Web Services (EWS)

The EWS request contains:

- HTTP headers
  - Authorization: Bearer token (from Office.js getCallbackTokenAsync)
  - User-Agent: ExchangeServicesClient/0.0.0.0
- SOAP request body XML

EWS APIs Used

We make the following calls via EWS to get the email or event and its attachments. We also write the Salesforce record ID to the properties of the Exchange item. Click the links for Microsoft documentation about the specific call.

- GetItem (client side and server side) to get and set AdditionalProperties and the content of the current email message when saving to Salesforce records.
- GetAttachment (server side) to retrieve the attachments from Exchange and add to Salesforce records (associated with the Salesforce email message representation).
- UpdateItem (client side)
- GetFolder (client side) to get the drafts folder.
- CreateItem (client side), which we use to create a draft message.

"Client side" refers to calls made via the Office.js API makeEwsRequestAsync. "Server side" refers to calls made from Salesforce app servers to EWS endpoint. For these server-side calls, we use a five-minute token from getCallbackTokenAsync.

GMAIL INTEGRATION

This section covers login authentication and the authenticated calls that the features in the Gmail integration Chrome extension use. If your email integration includes Inbox, also review the Integrations with an Inbox License section of this guide.

Configuration Requirements

Review Salesforce Help to set up the integration with Gmail. That Salesforce Help section also includes the Gmail Integration system requirements.

Authentication

Salesforce uses the OAuth 2.0 protocol to connect to a user's Google accounts. The Salesforce server obtains and stores an OAuth refresh token and access token for making requests to Google. This token is a single-user token that provides access to that user's Gmail account. The Chrome extension doesn't use this token directly. It's stored within the connected Salesforce org and treated as customer data.

The Gmail Integration uses Authentication Providers, a Salesforce platform feature, to store and manage the Google access tokens. Authentication Providers allow Apex to retrieve the access token and to refresh it. To learn more, see Authentication Providers in Salesforce Help

To stay logged in with Google, enable the Keep Gmail and Salesforce Connected preference, available on the Gmail Integration and Sync page in Salesforce Setup. That setting allows users to obtain a Salesforce session based on their Google identity. The Salesforce session follows the expiration time and other rules, such as allowable IP range, as set within Salesforce. When the Salesforce session expires, users can establish a new session based on their Google identity. This setup requires the user's browser to be logged into their Google account. When the Keep Gmail and Salesforce Connected preference is disabled, a user logs in the same way that they log in to Salesforce desktop. The same admin-controlled session rules apply. When the Salesforce session expires, users are required to log in again.

INTEGRATIONS WITH AN INBOX LICENSE

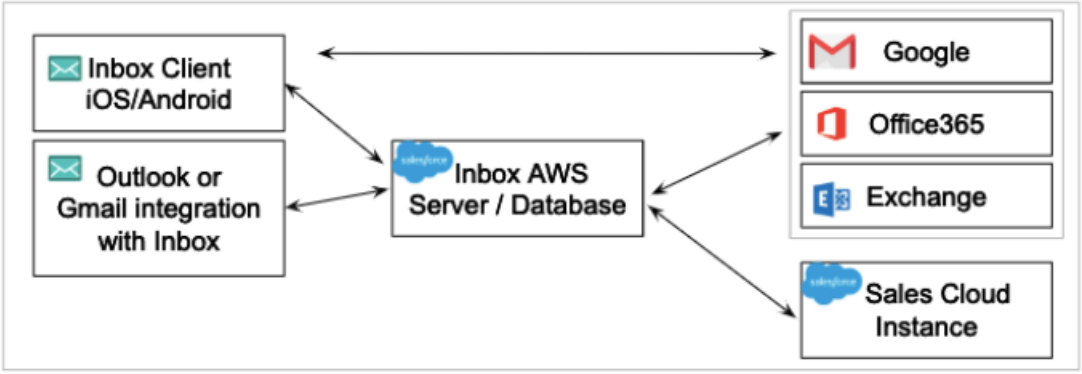
The following section details security items for connectivity of the integrations for users with an Inbox license, and users of the Inbox Mobile apps.

When users are assigned an Inbox license, their email mailbox is connected to the Salesforce Amazon Web Services (AWS) servers. This connection prompts the Salesforce AWS servers to make network calls to Google, Microsoft Exchange, or Office 365. Within the AWS data centers, our application uses keys and IDs to ensure that we serve the relevant data to the relevant customers.

For information about the security and architecture of the Einstein Platform that Inbox uses, see the Einstein Platform Trust and compliance documentation.

Network Connections

The Inbox mobile app makes network calls to the Salesforce AWS servers. Then, the servers make direct network calls to Microsoft Exchange, Office 365, and Google.



Outlook and Gmail integrations with an Inbox License to Salesforce AWS Servers
An HTTPS TLS 1.2 connection with AES-128 cipher. This connection is used for login and for performing Inbox-specific tasks.

Salesforce AWS Servers to Google
An HTTPS TLS 1.2 connection with AES-256 cipher. This connection uses the Gmail API protocol with OAuth 2.0 authentication.

Salesforce AWS Servers to Office 365 (Exchange Online)
An HTTPS TLS 1.2 connection with AES-256 cipher. This connection uses the EWS protocol with OAuth 2.0 authentication.

Salesforce AWS Servers to On-Premises Exchange (2013, 2016, and 2019)
An HTTPS TLS 1.1 or TLS 1.2 connection (the Exchange server decides the TLS version and cipher). This connection uses the EWS protocol with username and password basic authentication. If an IP or VPN restricts the EWS endpoint,whitelist the following addresses.

- 54.200.130.205
- 54.218.59.121

See the What are the Salesforce IP Addresses and Domains to White List article for information about Salesforce IP addresses that are not specific to Inbox.

iOS and Android Inbox App to Google
An HTTPS TLS 1.2 connection with AES-256 cipher. This connection uses the Gmail API protocol with OAuth 2.0 authentication.

iOS and Android Inbox App to Office 365 (Exchange Online)
An HTTPS TLS 1.2 connection with AES-256 cipher. This connection uses the Exchange ActiveSync (EAS) protocol or the Office 365 API protocol, each with OAuth 2.0 authentication.

iOS and Android Inbox App to On-Premises Exchange (2013, 2016, and 2019)
An HTTPS TLS 1.1 or TLS 1.2 connection. The Exchange server decides the TLS version and cipher. This connection uses the Exchange ActiveSync (EAS) protocol with username and password basic authentication.

See More About the OAuth Protocol for more details about how Inbox uses OAuth.

Salesforce AWS Servers Storage

The Salesforce AWS servers and databases are hosted in Amazon Web Services behind an AWS Virtual Private Cloud (VPC). The Salesforce AWS servers store the following information:

What	Details
Inbox User Settings	User's personal settings.
Inbox Email Accounts	Details about connected email accounts, including email address, server, and domain.
Email Headers and Metadata	The Salesforce AWS servers download and store email messages from your email accounts. The subject line including From, To, CC, Subject, and Sent Date are stored in a database in AWS. Einstein Activity Capture uses these emails for email logging, Einstein Email Insights' classification, and Recommended Connections suggestions.
Email HTML Bodies	The Salesforce AWS servers download and store email message HTML bodies. The HTML body is stored in Amazon S3 and is encrypted at rest, using AES-256 encryption. Einstein Activity Capture and Einstein Email Insights use the email body.
Email Attachments	The Salesforce AWS servers download and store email attachment metadata in a database. Einstein Activity Capture uses the attachment metadata. During the email send action, attachments themselves can be dynamically fetched from the Google or Exchange server by passing the email message ID. For the Send

What	Details
	Later feature, email attachments for emails to send are temporarily stored in Amazon S3 and are encrypted at rest using AES-256 encryption. After the email is sent, the email attachments are removed from Amazon S3.
Calendar Events	The Salesforce AWS servers download and store calendar events from your email accounts to the Salesforce database. Calendar events are not encrypted. Einstein Activity Capture and the Insert Availability feature use the stored events.
Passwords and OAuth Tokens	Inbox connects to your Salesforce, Google, and Office 365 accounts using the OAuth 2.0 protocol. The OAuth refresh and access tokens are stored in a database in AWS. They are encrypted at rest, using AES-256 encryption. For on-premises Exchange email accounts that use password authentication, the Exchange password is also stored in a database in AWS. The password is encrypted at rest, using AES-256 encryption.
Sales Cloud Records (Inbox mobile only)	To improve performance, Sales Cloud metadata (permissions, fields, and page layouts) for records such as contacts, leads, and opportunities are cached in the Salesforce AWS servers for up to 24 hours.
Contact Details	To improve performance, contact information displayed in the Contact Profile (from Gmail, Exchange, and Sales Cloud) is cached in the Salesforce AWS servers for seven days. The contact data is also used to create Einstein Email Insights.
Data Amount and Retention	When a new email account is connected, the Salesforce AWS Servers download six months or 180 days into the past. From that point on, the servers use notification subscriptions from Gmail, Exchange, or Office 365 to trigger downloading new email messages. By default, email messages are retained for two years before being deleted. The retention period can be configured from 30 days up to five years.

Encryption Key Management

The Salesforce AWS instance doesn't support customer-specific keys for encrypting data. It uses a single AWS-managed key for encryption.

The email metadata and calendar data in the database isn't encrypted. Only the HTML email bodies are stored as encrypted data.

Inbox Storage for Mobile

The Inbox mobile app stores the following information:

What	Details
Inbox User Settings	Personal user settings are stored in an SQLite database on the device and in the app preference. • NSUserDefaults on iOS • SharePreferences in Android The user settings stored include: • Salesforce settings (prompt to log email) • A user's set work hours • Email Settings, including swipe direction, number of lines to show in a message, organize by thread, and badge count type • Calendar settings, including declined events and number of days to display

What	Details
	.
Email Messages	The Inbox mobile apps store recent email messages, including the email body. Email messages are stored in an SQLite database. On iOS, the database is encrypted using SQLCipher, which uses AES-256 encryption. On Android, the database is not encrypted.
Calendar Events	The Inbox mobile apps store calendar events in an SQLite database. On iOS, the database is encrypted using SQLCipher, which uses AES-256 encryption. On Android, the calendar events are stored in the default shared calendar provider storage. See Android developer documentation for information on the calendar.
Passwords and OAuth Tokens	To fetch email and calendar events directly from Google, Office 365, and Exchange, the Inbox mobile apps store OAuth access tokens from Google and Office 365. They also store passwords for Exchange. These tokens or passwords are stored in the iOS keychain or Android Account Manager, which are default encrypted areas for the respective devices.
The iOS Keychain	See the Apple developer guide for their Keychain Services Concept.
The Android Account Manager	See the Android developer guide for their Account Manager provider.

Subsequent Logins for Inbox-Licensed Users

After the initial login, users with an Inbox license continue to authenticate to connect their mailbox with the Salesforce AWS server. This authentication obtains and stores an OAuth refresh token and access token, to make requests to Salesforce. This OAuth token is tied to the user's Salesforce account.

The OAuth refresh token and access token are stored on the Salesforce AWS servers. The connection makes API calls to obtain an OAuth access token. The access token expires in one hour, by default. A Salesforce admin can configure the expiration time. For example, the admin can set the OAuth refresh token to expire every seven days. To change the default token expire time:

1. From Salesforce Setup, enter `Connected Apps` in the Quick Find box, and then select **Managed Connected Apps**.
2. Select **SalesforceIQ** > **Edit** > **Refresh Token Policy**.

A user's Salesforce password is never stored in the Outlook or Gmail integration or in the Salesforce AWS server. A separate token is obtained for authenticating with the Salesforce AWS servers. With the Outlook and Gmail integrations, this token is an HttpOnly cookie. For the mobile apps, this token is a separate Bearer token that is passed in the Authorization HTTPS header.

All network calls to the Salesforce AWS servers are done over TLS 1.2.

The Salesforce admin can block OAuth access at any time. From Salesforce Setup, enter `Connected Apps` in the Quick Find box, and then select **Connected Apps OAuth Usage**.

An individual user can revoke their own Salesforce OAuth token. The Salesforce admin can also revoke OAuth tokens for any user. From your personal settings, view the OAuth tokens in Connections.

Gmail Guidelines

When a user has an Inbox license, the Gmail integration connects their Gmail account using the OAuth 2.0 protocol. The Chrome extension and mobile apps open a page for logging in to your Salesforce account. After this initial login, the Salesforce AWS server obtains and stores an OAuth refresh token and access token for making requests to Google. This token is a single-user token that provides access to that user's Gmail account. The Chrome extension and the Salesforce AWS servers don't use and never store the user's Google password.

Because the Inbox mobile app comprises email and calendar apps, they request full read and write access to the Google email and calendar. They also request read-only access to Google Drive, to support attaching files from Google Drive to emails. Inbox mobile apps request the following Google OAuth permissions.

- https://mail.google.com/
- https://www.google.com/m8/feeds/
- https://www.googleapis.com/auth/userinfo.profile
- https://www.googleapis.com/auth/userinfo.email
- https://www.googleapis.com/auth/calendar
- https://www.googleapis.com/auth/drive.readonly

A user can revoke the Google OAuth token at any time by navigating to the third-party app in the Google account user settings. See Google help for more information about removing third-party apps.

All network calls to Google are made using TLS 1.2 AES-256 cipher. Inbox uses two protocols with Google.

- Gmail API - See the Gmail API reference
- Google Calendar API - See the Google Calendar API reference

Exchange Online (Office 365) Guidelines

When users have an Inbox license, the Outlook integration connects the Office 365 account using the OAuth 2.0 protocol outlined in this guide. The Outlook add-in and mobile apps open a page for logging in to your Salesforce account. After this initial login, the Salesforce AWS server obtains and stores an OAuth refresh token and access token for making requests to Office 365. This token is a single-user token that provides access to that user's Office 365 account. Salesforce never uses or stores the user's Office 365 password.

Because the Inbox mobile apps are email and calendar apps, they request full read and write access to the Office 365 email and calendar. The Inbox mobile apps request the following Office 365 permissions.

- Send mail as a user
- Read and write user calendars
- Read and write user contacts
- Read user profiles
- Access mailboxes as the signed-in user via EWS
- Read and write use mail

A user can revoke the Office 365 OAuth token at any time. Go to Office 365, and then select **My Account** and **App Permissions**.

A Microsoft Azure administrator can change the Office 365 OAuth refresh token expiration time by setting it to MaxAgeSingleFactor and MaxAgeMultiFactor in the Microsoft Azure Active Directory.

All network calls between the Outlook add-in or Inbox mobile apps and Office 365 are performed over TLS1.2.

Salesforce uses these protocols to access the email and calendar from Office 365:

What	Protocol	Authentication	Notes
Salesforce AWS servers	EWS	Legacy authentication, which uses a username, domain, and password.	EWS is a SOAP-based protocol that Microsoft Outlook desktop uses on Windows and Mac. The Salesforce AWS server polls data based on notifications, rather than polling at a regular interval. For more information, see Notification subscriptions, mailbox events, and EWS in Exchange in the Office Dev Center.

What	Protocol	Authentication	Notes
Inbox mobile apps	Exchange ActiveSync (EAS) or Office 365 REST API	EAS supports authentication using a username, domain, and password. The Office 365 REST API uses an Oauth 2.0 access token.	Exchange ActiveSync (EAS) is a binary XML-based protocol targeted for mobile apps. The native iOS Mail app and Android Exchange mail app use the EAS protocol. To learn more about Exchange ActiveSync for iOS, view this Apple support topic. Microsoft also has a newer API, called the Microsoft Graph API or Office 365 REST API. The Microsoft Outlook app for iOS and Android can now use the newer Microsoft Graph API for reading and writing email and calendar events. The Inbox mobile apps don't use the IMAP protocol.

Microsoft Exchange On-Premises Guidelines

Connections with on-premises Exchange instances (2013, 2016, 2019) use legacy authentication, based on username, password, and domain. In this case, the Salesforce AWS server and the Inbox mobile app store the Exchange password. Passwords are encrypted at rest using AES-256 encryption.

Network calls with Exchange are performed using TLS 1.1 or TLS 1.2. The TLS protocol is determined by the highest TLS protocol supported by the Exchange server.

Salesforce uses two protocols to access email and calendar from Exchange:

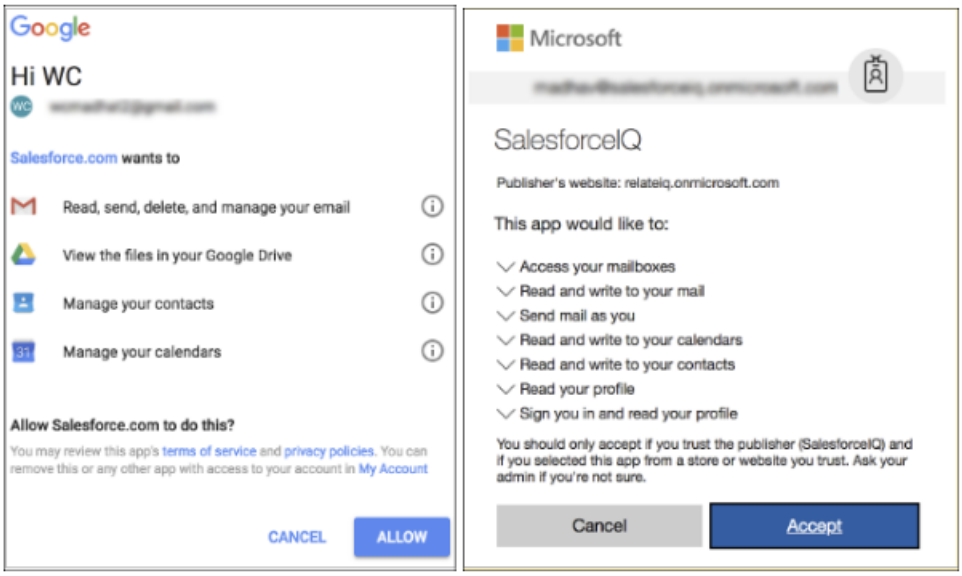
What	Protocol	Notes
Salesforce AWS servers	EWS	EWS is a SOAP-based protocol used in Microsoft Outlook desktop.
Inbox mobile apps	Exchange ActiveSync (EAS) or Office 365 REST API	Exchange ActiveSync (EAS) is a binary XML-based protocol targeted for mobile apps. The native iOS Mail app and Android Exchange mail app use the EAS protocol.

On-premise servers don't support the OAuth 2.0 protocol. Instead, the connection supports NT LAN Manager (NTLMv2) and basic authentication, which requires the username, domain, and password for authentication, unlike with OAuth 2.0 on Office 365.

Microsoft recently introduced a newer API, called the Microsoft Graph API or Office 365 REST API. Exchange servers that aren't on Office 365 don't support the newer Office 365 REST API or the Microsoft Graph API.

See Microsoft documentation for information about Office 365 APIs.

More About the OAuth Protocol



When provisioned with an Inbox license, users connect to the Salesforce platform and to their email provider. Mobile users with a Gmail or Office 365 email account connect using the OAuth 2.0 protocol. After this initial login, the Salesforce AWS server obtains and stores an OAuth refresh token and access token for making requests to the server.

OAuth 2.0 allows third-party applications, such as Salesforce, to obtain access to a service such as Gmail and Office 365 while delegating the login process to that service.

When the user connects a service, Salesforce opens a new page to the service's login page. This page is the service's own login web page such as:

- https://login.salesforce.com/
- https://accounts.google.com
- https://login.microsoftonline.com

After users log in, they're prompted to grant permissions to the third-party app, in this case, Salesforce. These permissions include:

- Full read and write access to your email.
- Full read and write access to your calendar.

When the user allows permission, the service redirects back to the Salesforce AWS server. The Salesforce AWS server connects to the service and obtains the following:

- An OAuth refresh token
- An OAuth access token

The OAuth access token is used to make authenticated requests to the service, such as fetching email or calendar events. These tokens typically expire after one hour. To obtain another access token, the OAuth refresh token is used. The refresh token typically has a long expiration time or no expiration time.

Salesforce AWS Server Operations

The Salesforce AWS Server performs the following email and calendar operations.

- Fetch and store email headers and HTML body from your mailbox.
- Fetch email headers and HTML body from your Sent Mail.
- Fetch and store calendar events.
- Send email messages with the Send Later feature.
- Create a calendar event with attendees with Insert Availability.

The Inbox mobile apps have full read and write access to email and calendar. They perform the following operations on email and calendar.

- Reading, composing, and sending email, including attachments
- Marking as read, flagging, moving an email to a folder, archive, trash
- Viewing, creating, and editing calendar events
- RSVPing to calendar events

Mobile Device and Application Management and Inbox

Inbox mobile app has limited support for Mobile Device Management (MDM) and Mobile Application Management (MAM).

Inbox mobile supports MDM Client Certificates for the following:

- A user's Salesforce login.
- When connecting an Office 365 email account.
- When connecting a Google email account.

Inbox mobile doesn't support MDM Client Certificates for Exchange 2013 or 2016 email accounts.

For specific vendors:

- Inbox mobile supports Microsoft InTune MDM only with Office 365 email accounts. Inbox mobile doesn't support other MDM solutions, such as MobileIron, VMWare Airwatch, and others.
- Inbox mobile doesn't support Microsoft InTune Mobile Application Management (MAM).

Mobile App Data Removal

A Salesforce admin can remove a Salesforce or Salesforce Inbox license for a specific user at any time. After the license is removed, when a user opens the app, the app logs out the user and deletes all data stored locally for that user.

An Exchange admin can also initiate an Exchange ActiveSync Remote Wipe. An Exchange admin can also initiate an Exchange Active Sync Remote Wipe.

- In Office 365, navigate to Office 365 settings to wipe data.
- For on-premises Exchange servers, navigate to the Exchange Admin Center and wipe data.

After wiping data, if the app is opened, the app connects to Exchange, receives the remote wipe response. It marks the email account as disabled, logs the user out of the app, and deletes all the data stored locally for that user.