

---

# Salesforce1 Mobile Security Guide

Version 1, 1





# CONTENTS

- Chapter 1: Introduction** ..... 1
- Chapter 2: Salesforce1 Architecture Overview** ..... 2
- Chapter 3: Authorizations and Permissions** ..... 3
- Chapter 4: Communication Security** ..... 4
- Chapter 5: Authentication** ..... 5
  - OAuth Pairing ..... 6
  - Single Sign On (SSO) ..... 6
  - Certificates and Keys ..... 7
  - Identity Providers and Service Providers ..... 7
  - Inactivity Lock ..... 8
  - Session Token ..... 8
  - Restrict Device Platforms ..... 8
- Chapter 6: Storage Security** ..... 9
  - Local Data Protection ..... 10
  - Remote Wipe ..... 11
- Chapter 7: Mobile Device Management (MDM)** ..... 12
- Chapter 8: Notes** ..... 14



# CHAPTER 1 Introduction

This document describes the Salesforce1 mobile application (Android and iOS downloadable application, and the mobile browser application), and addresses security concerns an enterprise may have when evaluating Salesforce1 for their organization.

This document does not cover Salesforce Classic, Mobile Dashboards, Touch applications, or BlackBerry.

## CHAPTER 2 Salesforce1 Architecture Overview

Salesforce1 uses the Force.com platform, with all application logic and database storage provided by salesforce.com's hosted application servers. The Salesforce1 solution consists of the Salesforce application server, and either the client application or mobile browser on the handheld mobile device. Supported operating systems are Apple iOS and Android.

The Salesforce1 client application communicates across the wireless network to display a subset of the user's data on the handheld device. The client application or browser on the handheld device pulls feed data on demand to the device. This architecture provides a very high quality of service and a productive working experience for the end user.

Salesforce1 provides a sandboxed environment for a user to access Salesforce data from a mobile device, while an org administrator can manage user access, even if the mobile device belongs to the user.

## CHAPTER 3 Authorizations and Permissions

Access to Salesforce1 is “default on” and does not require an administrator to grant permission to use the application. Administrators can edit profile and permission sets to revoke Salesforce1 access to any user through the administration console . The Salesforce1 application provides access to data and functions based upon the core permissions and rights defined for each user by their Salesforce administrator. Mobile users are never able to view or access more than their permissions allow.

### Installation

---

When the Salesforce1 application is installed on a mobile device, the permissions requested vary for each OS.

- **Android Downloadable App:** At the time of installation, Android requires permission for:
  - Device and App History
  - Identity
  - Calendar
  - Contacts
  - Location
  - Phone
  - Photos/Media/Files
  - Wi-Fi Connection Information
  - Device ID and Call Information
- **iOS Downloadable App:** After installation, iOS requests permission for each item listed above on an individual basis, and the user can approve or deny the request.

## CHAPTER 4 Communication Security

Salesforce1 uses SSL/TLS for Over-The-Air (OTA) communication encryption. All Salesforce OAuth authorization endpoints are HTTPS only. Communication requests over HTTP are denied by Salesforce servers, unless the org administrator opts out of and unchecks “Require secure connections (HTTPS)” in the administration console.



# CHAPTER 5 Authentication

## In this chapter ...

- OAuth Pairing
- Single Sign On (SSO)
- Certificates and Keys
- Identity Providers and Service Providers
- Inactivity Lock
- Session Token
- Restrict Device Platforms

All components of Salesforce1 require user authentication at the point and time of access. Salesforce1 utilizes OAuth2.0 for authentication through username/password or SSO (single sign-on) credentials.

## OAuth Pairing

---

During the initial login, the device is uniquely identified and paired with the mobile user's account using the OAuth 2.0 protocol (<http://tools.ietf.org/html/rfc6749>). All requests to the Salesforce service are made using the OAuth token established through the pairing created during activation.

After initial login, there is no exchange of a password in the communication between the mobile client and the Salesforce server. For this reason, the Salesforce password is not stored on the device and is not required even when the password is changed or has expired.

A user obtains an access and refresh token after successfully completing the OAuth 2.0 web server authentication. A user can use the refresh token to get a new access token (session ID). Upon logout, the OAuth access and refresh tokens are revoked, and the user set passcode is wiped (if passcode is enabled by org administrator). The user is re-prompted to enter the username/password and reset the passcode.

The org administrator can revoke a refresh token the first time a user uses the app, every time a user uses the app, or on set a schedule (hourly, daily, or monthly) to force a user to re-enter the username/password and reset the passcode. The default token expiration schedule is set at 2 hours, but can be as short as 15 minutes.

## OAuth Access Token Storage

- **iOS Downloadable App:** AES-128 with a 256 bit key consisting of a SHA-256 hashed concatenation of a generated RFC 4122 Universally Unique Identifier (persisted to the encrypted keychain) and a Base64 encoded SHA-256 hash of the device passcode (this 4 to 8 digit non-alphanumeric passcode requirement is enabled by the org administrator and is set by the client). Token is stored in the keychain using `kSecAttrAccessibleWhenUnlockedThisDeviceOnly` to preserve the session for the user, because iOS may terminate the application.
- **Android Downloadable App:** PBKDF2 produced AES-256 encrypted key derived from device unique Android ID and randomly generated string. Token is stored in Android's AccountManager. The SQLCipher-encrypted key is derived from the passcode, if enabled by org administrator, or UUID (universally unique identifier) if the passcode isn't enabled.
- **Mobile Browser App:** Access token is never stored on the mobile device. The mobile browser app requires a user to re-enter the username/password to obtain a new access token.

## OAuth Refresh Token Storage

- **iOS Downloadable App:** AES-128 with a 256 bit key consisting of a SHA-256 hashed concatenation of a generated RFC 4122 Universally Unique Identifier (persisted to the encrypted keychain) and a Base64 encoded SHA-256 hash of the device passcode (this 4 to 8 digit non-alphanumeric passcode requirement is enabled by the org administrator and is set by the client). Token is stored in the keychain using `kSecAttrAccessibleWhenUnlockedThisDeviceOnly`.
- **Android Downloadable App:** PBKDF2 produced AES-256 encrypted key derived from device unique Android ID and randomly generated string. Token is stored in Android's AccountManager. The SQLCipher-encrypted key is derived from the passcode, if enabled by org administrator, or UUID (universally unique identifier) if the passcode isn't enabled.
- **Mobile Browser App:** The web server authentication flow for the mobile browser app doesn't use or store a refresh token on the device. The mobile browser app requires a user to re-enter the username/password to obtain a new access token.

## Single Sign On (SSO)

---

Single sign-on is a process that allows network users to access all authorized network resources without having to log in separately to each resource. Single sign-on allows orgs to validate username/password against their user database or other client applications rather than having separate username/password managed by Salesforce.

## Federated Authentication Support

When federated authentication is enabled, Salesforce doesn't validate a user's password. Instead, Salesforce verifies an assertion in the HTTP POST request, and allows single sign-on if the assertion is true. This is the default form of single sign-on.

See "[Single Sign-On for Desktop and Mobile Applications using SAML and OAuth](#)" for more information.

## Delegated Authentication Support

When delegated authentication is enabled, Salesforce does not validate a user's password. Instead, Salesforce makes a Web services call to a customer org to establish authentication credentials for the user. Administrators must request delegated authentication support be enabled by Salesforce.

See "[Understanding Delegated Authentication Single Sign-On](#)" for more information.

## Certificates and Keys

---

Salesforce certificates and key pairs are used for signatures that verify a request is coming from a customer org. They are used for authenticated SSL communications with an external web site, or when using a customer org as an Identity Provider. Customers only need to generate a Salesforce certificate and key pair if they're working with an external website that wants verification that a request is coming from a Salesforce org.

Salesforce offers two types of certificates:

- **Self-Signed:** A self-signed certificate is signed by Salesforce. Not all external websites accept self-signed certificates.
- **CA-Signed:** A CA-signed certificate is signed by an external certificate authority (CA). Most external websites accept CA-signed certificates. Customers must first generate the certificate signing request to send to a CA, and then import the signed version of the certificate before they can use it.

See "[About Salesforce Certificates and Keys](#)" for more information.

## Identity Providers and Service Providers

---

An identity provider is a trusted provider that enables a customer to use single sign-on to access other websites. A service provider is a website that hosts applications. Customers can enable Salesforce as an identity provider, then define one or more service providers, so their users can access other applications directly from Salesforce using single sign-on. This can be a great help to users: instead of having to remember many passwords, they will only have to remember one.

Salesforce is automatically enabled as an identity provider when a [domain is created](#). After a domain is deployed, administrators can add or change identity providers and increase security for their organization by customizing their domain's login policy.

Enabling Salesforce as an identity provider requires a [Salesforce certificate and key pair that is signed by an external certificate authority \(CA-signed\) or self-signed](#). If customers haven't generated a Salesforce certificate and key pair, one is automatically created for them when they enable Salesforce as an identity provider. They also have the option of picking an already generated certificate, or creating one.

Salesforce uses the SAML 2.0 standard for single sign-on and generates SAML assertions when configured as an identity provider.

See "[About Identity Providers and Service Providers](#)" for more information.

## Inactivity Lock

---

Upon initial activation, Salesforce1 prompts the user to create an arbitrary passcode (if required by the org administrator), which is used to unlock the application after reboot, or an administrator defined period of inactivity (1, 5, 10, or 30 minutes).

The passcode lock protects lost or stolen devices that may have their wireless connection disabled, and can't have their OAuth token revoked.

## Passcode Strength and Storage

- **iOS Downloadable App:** 4 to 8 digits (non-alphanumeric). A Base64 encoded SHA-256 hash of the passcode is stored in the secure keychain using `kSecAttrAccessibleWhenUnlockedThisDeviceOnly` for passcode validation and a component of the AES-128 encryption key for various encryption processes.
- **Android Downloadable App:** 5 or more alphanumeric characters. PBKDF2 produced AES-256 encrypted key derived from device unique Android ID and randomly generated string. Token is stored in Android's AccountManager. The SQLCipher-encrypted key is derived from the passcode, if enabled by org administrator, or UUID (universally unique identifier) if the passcode isn't enabled.
- **Mobile Browser App:** Users are prompted to re-enter username/password after 30 minutes of inactivity, or if they navigate to a different site or close the mobile browser.

Salesforce1 guards against brute force attacks by erasing all locally stored data after 10 failed attempts at entering the passcode. Reactivation is required to continue using the application.

## Session Token

---

Session token is only used for Visualforce pages. It is derived from the OAuth Access Token and is scoped to the Visualforce page. The UIWebView/WebView stores it in the cache.

## Restrict Device Platforms

---

Administrators can restrict Salesforce1 app access through the administration console by blocking the Salesforce1 Connected App for either platform (iOS or Android).

Administrators can also enable/disable the mobile browser app through administration console. If the mobile browser app is disabled, the user is taken to the full Salesforce site from the mobile browser.

## CHAPTER 6 Storage Security

### In this chapter ...

- [Local Data Protection](#)
- [Remote Wipe](#)

A mobile device may be lost or stolen at any time. Since mobile devices are small and designed to be highly portable, they may not remain under the physical control of a trusted person. Therefore, Salesforce1 provides methods to secure the device data if it passes out of control of the user or the user's organization.

Salesforce1 has multiple levels of security at the handheld device level. First, device vendors provide the ability to enforce OS-level password access restrictions on any device applications or data. Users must be required to use the device protection in accordance with the owning enterprise's security policy. If the device is locked by a strong password, it is difficult for unauthorized persons to do anything with it.

## Local Data Protection

---

Salesforce1 does not currently support any external memory. The data stored locally on the device is saved in the device's embedded memory and never on an external memory card.

Mobile platforms don't generally allow data extraction from a local database. In order to make the system more secure, Salesforce1 does provide encryption on the device database.

## Feed Database Encryption

Feeds are made up of feed items. A feed item is a piece of information posted by a user (for example, a poll) or by an automated process (for example, when a tracked field is updated on a record).

- **iOS Downloadable App:** Feed data is stored using Core Data, and encrypted using NSFileProtectionComplete. NSFileProtectionComplete dictates how passcodes are exposed internally to access the feed data. The passcode for the feed data is removed from the local keychain when Salesforce1 is closed or running in the background. Salesforce1 feed data is only accessible when the app is open and in the foreground. If an OS passcode isn't set or if the device screen is unlocked, the encrypted feed data can be compromised by brute force attacks.

Additionally, the feed data storage is time-based. The feed cache purges items if the time last viewed is greater than one week, unless the remainder of feed items after purging is less than 25 items. Also feeds that have more than 500 items will have their excess items removed.

- **Android Downloadable App:** Feed data is stored in a SQLCipher-encrypted SQLite database with PBKDF2 produced AES-256 encrypted in CBC (cipher-block chaining) mode with appropriate IV (initialization vector) and PKCS #5 padding.
- **Mobile Browser App:** The feed cache is invalidated on client logout, navigation to a different site, or when closing the mobile browser.

## Files and Attachments

A file or attachment is any file that a user uploads, shares, or attaches to posts, comments, or records. All file types are supported: documents, presentations, spreadsheets, PDFs, images, audio files, and video files.

- **iOS Downloadable App:** Files and attachments are stored on the device's file system in a double-encrypted format. We use the device's hardware encryption capability to encrypt the files while the device is locked and in addition we perform our own encryption using AES algorithm (128 bit block size and 256 bit key size). When the file is being viewed, there's a temporary unencrypted copy kept on the file system (removed when the 'viewing' operation is complete).
- **Android Downloadable App:** To store files offline, we require the user to enable device encryption and utilize the OS's file encryption system. This allows the application to securely store local files.
- **Mobile Browser App:** The files cache is invalidated on client logout, navigation to a different site, or when closing the mobile browser.

## Offline Sync

If Salesforce1 users lose their wireless connection, they can enable offline sync to navigate within the app and view most recent items.

- **iOS Downloadable App:** Offline data is stored using Core Data, and encrypted using NSFileProtectionComplete. NSFileProtectionComplete dictates how passcodes are exposed internally to access the offline data. The passcode for the offline data is removed from the local keychain when Salesforce1 is closed or running in the background. Salesforce1 offline data is only accessible when the app is open and in the foreground. If an OS passcode isn't set or if the device is unlocked, the encrypted offline data can be compromised by brute force attacks.

- **Android Downloadable App:** Offline data is stored in the Salesforce Mobile SDK SmartStore, which is a SQLCipher-encrypted SQLite database with PBKDF2 produced AES-256 encrypted key in CBC (cipher-block chaining) mode with appropriate IV (initialization vector) and PKCS #5 padding.
- **Mobile Browser App:** Offline sync functionality isn't available in the mobile browser app.

## Remote Wipe


---

To minimize the risk of information loss when a device is compromised, an org administrator can:

1. Disable a user completely (e.g. termination of an employee) to remove access and wipe the data from the application.
2. View the Connected Apps OAuth Usage report in the administration console to revoke the OAuth refresh token and associated access tokens. This wipes the application, which forces the user to re-authenticate (e.g. employee loses a phone).

## CHAPTER 7 Mobile Device Management (MDM)

With version 8.0 or later of the Salesforce1 downloadable app for Android, Salesforce provides an extra level of security compliance through interoperation with the most popular MDM (mobile device management) suites. The Salesforce1 downloadable app for Android, with an MDM, give you enhanced functionality for distribution and control over your users' devices. The enhanced security functions when you combine Salesforce1 with an MDM are certificate-based authentication and automatic custom host provisioning.

 **Note:** SAML 2.0 (security assertion markup language) must be enabled and configured for your organization.

There are prerequisites to implement enhanced security for Salesforce1 for Android.

- First, configure Android for Work for your org. Android for Work is a program that supports enterprise use of Android devices. See [Android for Work](#) to learn more about the program and [Android for Work Help](#) for setup information.
- Once Android for Work is set up, the next step is to configure your Mobile Device Management (MDM) suite. There are a multitude of MDM solutions in the market place. When you decide on the right product, work with your MDM provider to complete the configuration for your org.
- After you have Android for Work and your MDM suite up and running in your org, you're ready to implement the enhanced security features of the Salesforce1 for Android app.

### Certificate-Based Authentication

Using certificates to authenticate simplifies provisioning your mobile users, and your day-to-day mobile administration tasks by eliminating usernames and passwords. Salesforce uses X.509 certificates to authenticate users more efficiently, or as a second factor in the login process.


#### MDM Settings for Certificate-Based Authentication

To enable certificate-based authentication for your mobile users, you need to configure key-value pair assignments through your MDM suite. Here are the supported keys:

Key	Data Type	Platform	Description
RequireCertAuth	Boolean	Android	If true, the certificate-based authentication flow initiates.  <b>Android:</b> Uses the user certificate on the device for authentication inside a webview.



Key	Data Type	Platform	Description
ManagedAppCertAlias	String	Android	Alias of the certificate deployed on the device picked by the application for user authentication. Required for Android only.

 **Note:** There's a minimum device OS version requirement to use certificate-based authentication. For Android, the minimum supported version is 5.0. For iOS, the minimum supported version is 7.0.

Once you save your key-value pair assignments, you can push the mobile app with the updated certificate-based authentication flow to your users via your MDM suite.

## Automatic Custom Host Provisioning

You can now push custom login host settings to your mobile users. This spares your mobile users from having to manually type long URLs for login hosts—typically a frustrating and error-prone activity. You can configure key-value pair assignments through your MDM to define multiple custom login hosts for your mobile users.

### MDM Settings for Automatic Custom Host Provisioning

To push custom login host configurations to your mobile users, you need to configure key-value pair assignments through your MDM suite. Here are the supported keys:

Key	Data Type	Platform	Description
AppServiceHosts	String, String Array	Android	Login hosts. First value in the array is the default host.  <b>Android:</b> Requires https:// in the host URL.
AppServiceHostLabels	String, String Array	Android	Labels for the hosts.  The number of <code>AppServiceHostLabels</code> entries must match the number of <code>AppServiceHosts</code> entries.

## CHAPTER 8 Notes

- iOS: Prior to entering `applicationDidEnterBackground`, a benign splash screen is displayed to protect sensitive data from automatic iOS snapshotting (iOS uses automatic snapshotting for transition animations). The application prevents any snapshots of customer data during backgrounding.
- Security is not a binary (on/off), but implemented at different levels.
- Salesforce1 provides multiple levels of security; however, there's no application that can guarantee a completely secure system.