Mobile API Connector Installation Guide

SurePassID Authentication Server 23.1

SurePassID, Corp.

360 Central Avenue

First Central Tower

Suite 800

St. Petersburg, FL 33701

USA

+1 (888) 200-8144

www.surepassid.com

Table of Contents

About the SurePassID Mobile API Connector

This guide explains how install and configure the SurePassID Mobile API Connector. The purpose of this guide is to provide a reference for system administrators.

This guide provides information on the following topics:

- **What is SurePassID Mobile API Connector?**
  - A brief introduction to the Mobile API Connector
- **Installing and Configuring Mobile API Connector**
  - Detailed explanations for installing the Mobile API Connector

Other SurePassID Guides

The Mobile API Connector has the following companion guides that provide additional detail on specific topics for SurePassID:

- Server API Guide
- Fido U2F Mobile API Guide
- System Administration Guide
- Local Agent Guide
  - High performance Radius Server
  - Windows Event Log Integration
  - Active Directory Synchronization
- Desktop OTP Authenticator Guide
- Google Authenticator Guide
- Windows Login Authentication Guide

What is the SurePassID Mobile API Connector?

The SurePassID Mobile API Connector is a server designed to act as an intermediary (proxy) between mobile apps that request authentication services from SurePassID authentication server that runs deep behind the company firewall. The mobile API connector supports the same API as the SurePassID authentication server so that apps that are built for “trusted zone” can be moved into the ”unstrusted zone” of the mobile world and remain secure. The system is distributed as a Windows set-up (msi) install that you run on an existing Windows server in the DMZ and operates on 32-bit or 64-bit Windows server.

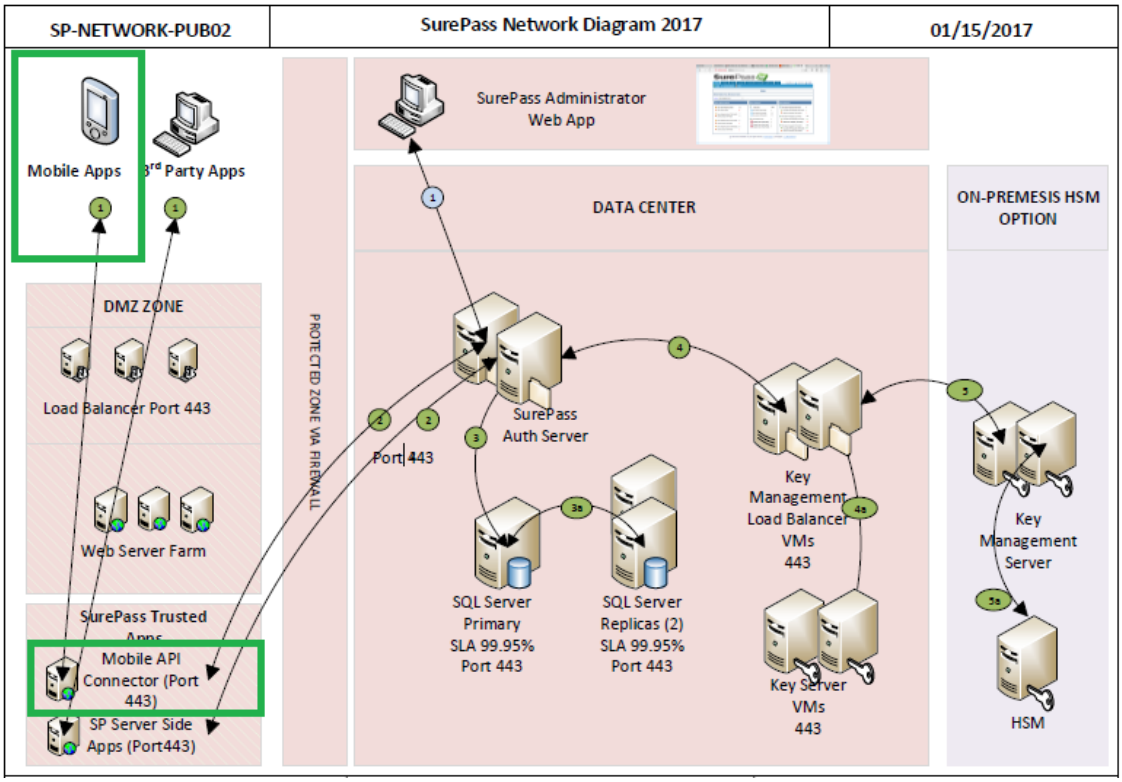
The Mobile API Connector supports on-premises and hybrid, private and public cloud installations of SurePassID Authentication Server.

The Mobile API connector supports both traditional OATH OTP password authentication, authentication push technologies, as well as new technologies such as wearables, biometrics, and native FIDO U2F authentication on iOS, Android and other mobile operating systems.

You need to use the SurePassID Mobile API Connector if you using the following SurePassID technologies.

- SurePassID Mobile Authenticator
- SurePassID U2F & OATH Native Mobile SDK
- SurePassID ServicePass Self-service Portal (SurePassID Authentication Server on-premises; cloud optional)
- Developing third-party apps that require SurePassID authentication services

The high level architecture of the system is shown below:



This document focuses on the area's that are boxed in green.

Prerequisites

SurePassID Server can be installed on the following Windows versions:

- Windows Server 2008 – All versions
- Windows Server 2012 – All versions
- Windows Server 2016 – All versions
- Windows 7 – Professional & Ultimate
- Windows 8 – Professional & Ultimate
- Windows 10

Security

SurePassID Mobile API Connector does not install with any certificates for SSL. It is recommended that you configure the SurePassID Mobile API Connector (IIS web app) for SSL using corporate certificates for production. Or create self-signed certificates for testing.

It is recommended that SurePassID Mobile API Connector is configured to communicate to the SurePassID Authentication Server using transport level security (https) on a specific port (see Customizing the System section).

IMPORTANT Security note: The firewall should be configured to only allow communications on that port from the SurePassID Mobile API Connector server IP.

The SurePassID Mobile API Connector configuration file can limit the types of REST API requests that will be permitted. It is important that you only allow the requests that your mobile app supports.

For ultra-secure operations the SurePassID Mobile API Connector can be daisy-chained across many servers.

It is recommended that you follow security best practices for deploying mobile applications.

Database

The system does not require any database access.

Request Logging

The system does support request logging. Logging captures the payload and IP of the requesting application. By default the logs will be persisted in the local file system. Alternatively, they can be sent to the Windows Event Log for persistence, trouble shooting or further analysis.

Internet Information Server

The Windows server must have the IIS feature enabled.

Post Configuration Steps

It is HIGHLY recommended that you proceed with the following steps after installation:

- Set up TLS for the MobileAPI Connector IIS virtual directory
- Rename the MobileAPI Connector IIS virtual directory to something that conforms to your standards.
- Update DNS (internal or external depending on the use) to allow for access to the MobileAPI Connector via A record or CNAME.
- Customize the web.config file as per the Customizing the System section.

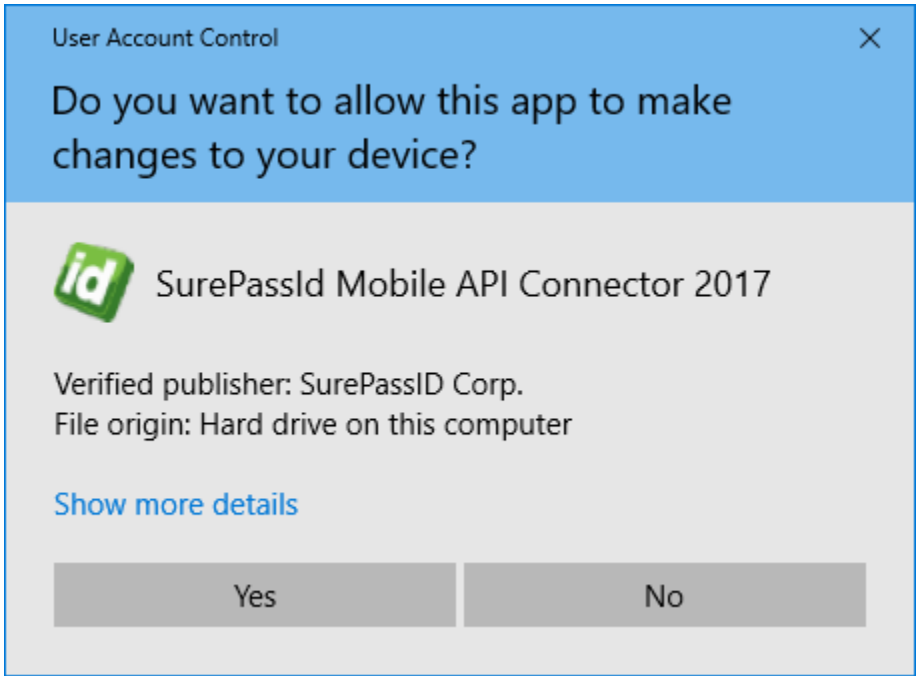
- Protect the **web.config** file in the root folder of the SurePassId configuration by encrypting it using Aspnet_regiis utility. Detail procedures on how to do this can be found here:

  https://msdn.microsoft.com/en-us/library/zhhddkxy(v=vs.140).aspx

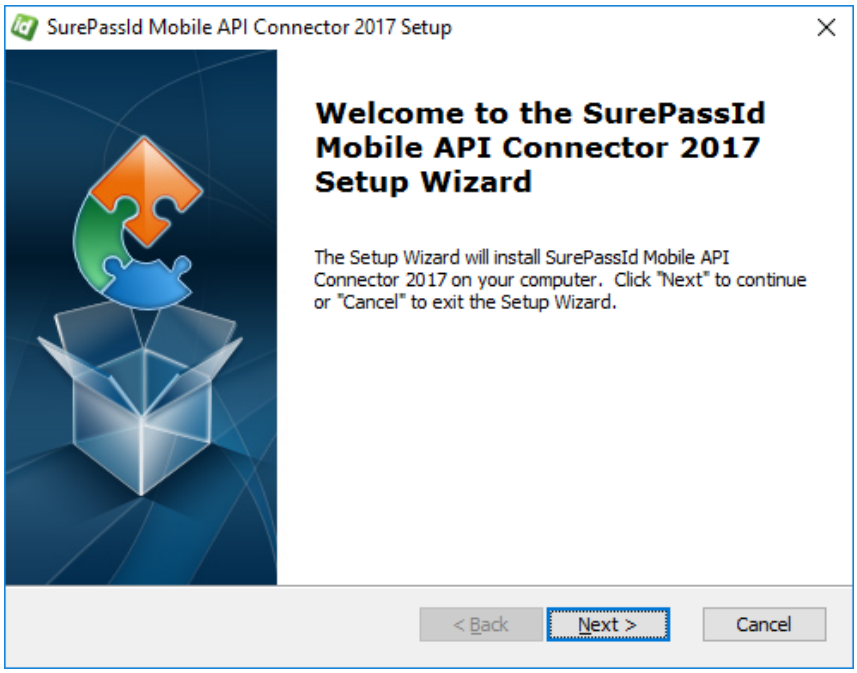
Installing and Configuring Mobile API Connector

The SurePassID Mobile API Connector is distributed as a Windows msi installer file (**SurePassIDMobileAPIConn2019.exe**) located in a zip file (**SPMACONN.ZIP**).

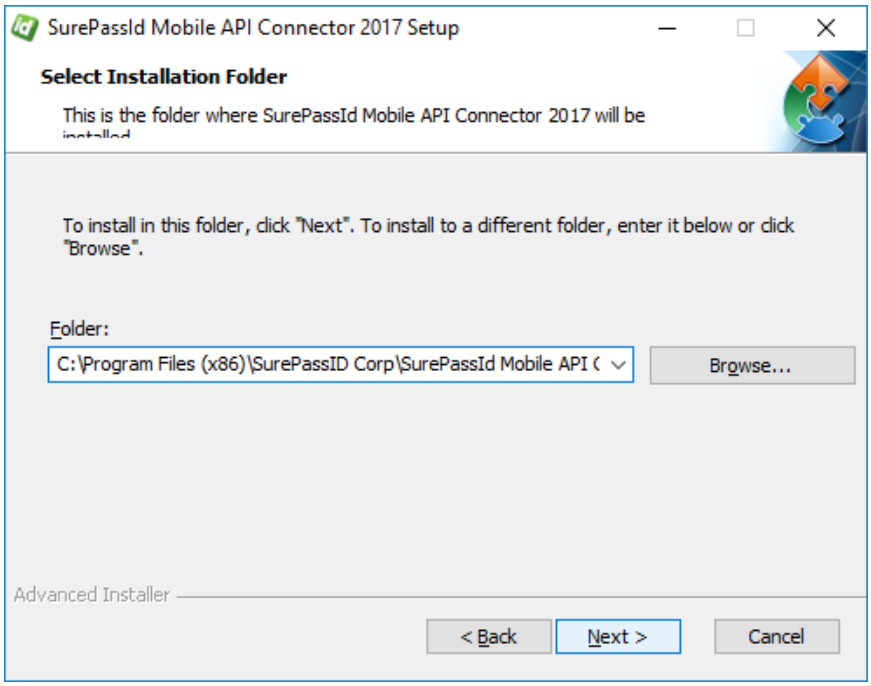
After downloading and unzipping **SPMACONN.ZIP**, locate the file **SurePassIDMobileAPIConn2019.exe** file, copy the file to the appropriate Windows server (if not already there) and run **SurePassIDMobileAPIConn2019.exe** to install the system. The following window will be displayed.



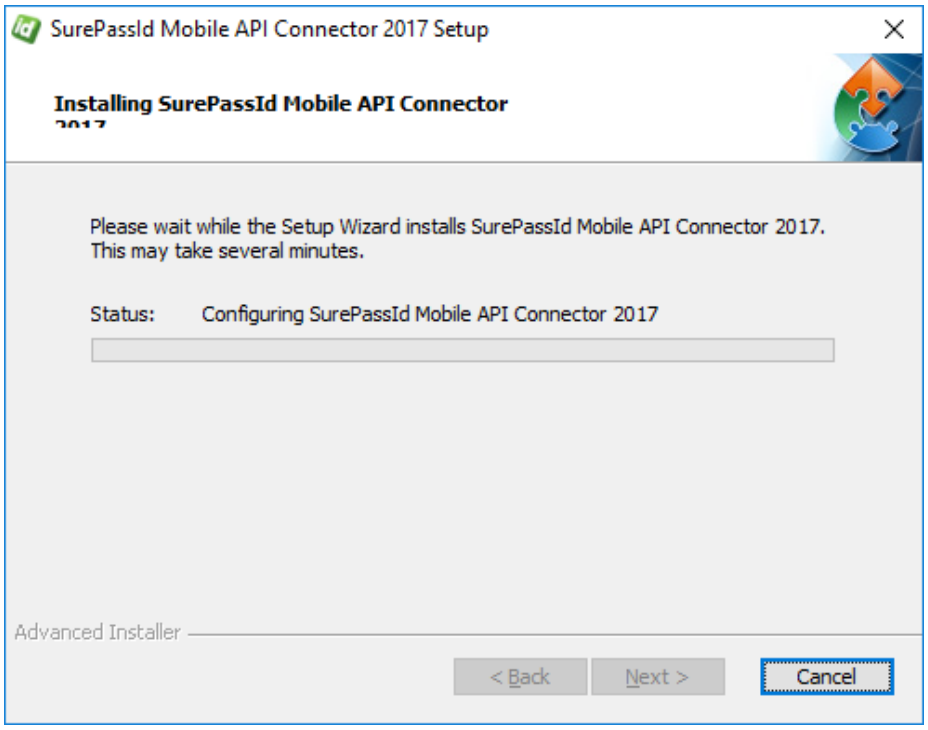
Click Yes.



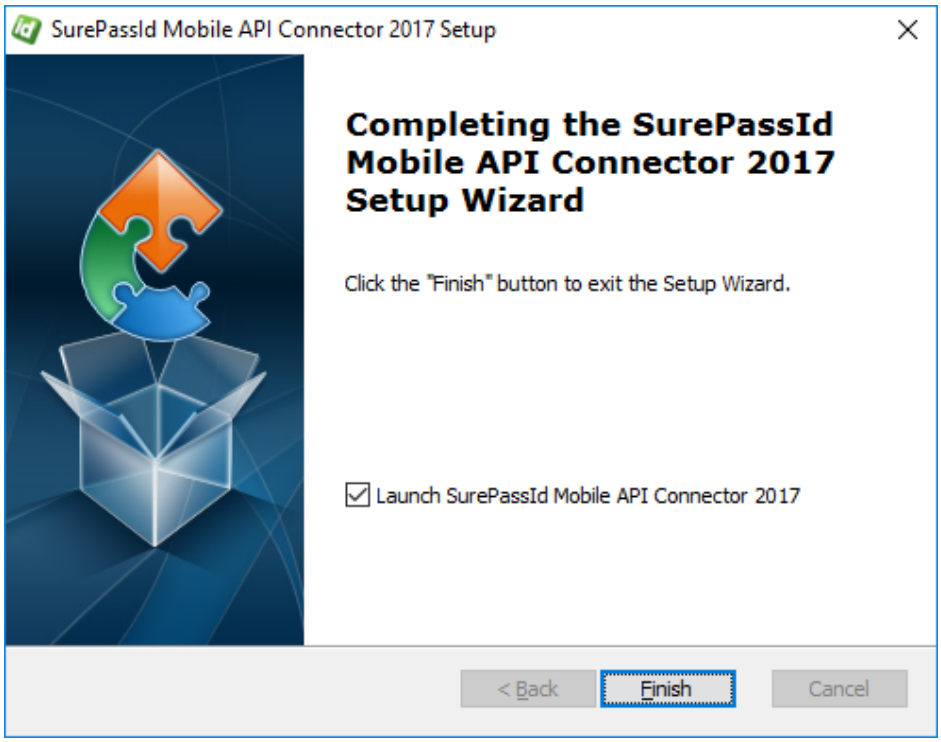
Click **Next**.



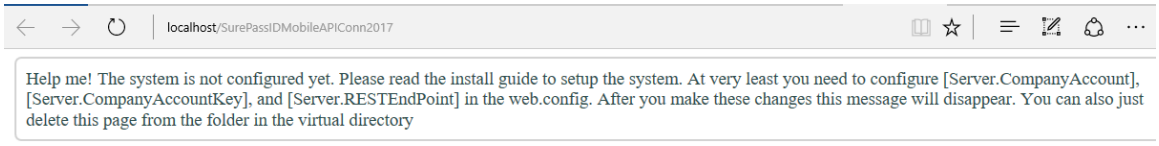
Click **Next**.



You will see files being installed and IIS configured. When complete you will see:



Click **Finish**. The MobileApiConnector will be started and you will see the following screen:



This screen informs you that the system is installed and ready to be customized before use. Once you customize the system, this screen will go away. The next section will provide you with the information you need to customize the system to use your instance of the SurePassID Authentication Server endpoint and set allowable access.

Customizing the System

When installation is completed, you will have a fully functioning system, however, it cannot properly service requests. There are certain customizations required to tailor the system to your company's requirements.

Customizations are made in the **web.config** file located in the root folder of the SurePass installation. Local customizations are made by each tenant using the SurePassID Admin portal.

Web.config

The web.config file is an XML file and is part of the .Net Framework. The file contains global customization settings. Some of the settings are SurePassID specific (<configuration>**<appsettings>)** and you should change them to suite your needs. Other settings effect the way that ASP .Net operates and you should not change these settings unless you have experience in this area. Some settings you can change and others you should not. If you make a change to web.config that violates the rules of xml syntax, the system will not run and you will receive an error. The table below describes the most notable SurePassID specific settings:

<configuration>**<appsettings> keys**

Parameter	Description	Used By
`Server.RESTEndPoint`	The server endpoint that will process all mobile requests.	SP REST API SP Authenticator SP Desktop Authenticator SP U2F Mobile SDK
`Server.CompanyAccount`	Your company account in the server	SP REST API SP Authenticator SP Desktop Authenticator SP U2F Mobile SDK
`Server.CompanyAccountKey`	Your company account key in the server	SP REST API SP Authenticator SP Desktop Authenticator SP U2F Mobile SDK
`Access.AllowU2FEnroll`	Allow apps to enroll U2F keys with the server	SP REST API SP Authenticator SP Desktop Authenticator SP U2F Mobile SDK
`Access.AllowU2FSign`	Allow apps to verify U2F keys with the server	SP REST API SP Authenticator SP Desktop Authenticator SP U2F Mobile SDK
`Access.AllowU2FDeleteKey`	Allow apps to delete a U2F key on the server	SP REST API SP Authenticator

		SP Desktop Authenticator SP U2F Mobile SDK
Access.AllowU2FDeleteKeys	Allow apps to delete all U2F keys on the server	SP REST API SP Authenticator SP Desktop Authenticator SP U2F Mobile SDK
Access.AllowValidatePasscode	Allow apps to verify a One Time Passcode on the server.	SP REST API SP U2F Mobile SDK
Access.AllowSendPasscode	Allow apps to send a One Time Passcode to the user via email or sms.	SP REST API SP U2F Mobile SDK
Access.AllowActivateSoftToken	Allow SP Authenticator or SP Desktop Authenticator to provision a mobile token over the air.	SP REST API SP Authenticator SP Desktop Authenticator
Access.AllowValidateUser	Allow an app to verify a user via username and password.	SP REST API SP U2F Mobile SDK
Access.AllowAddUser	Allow app to add a new user (and optionally a soft token) to the SurePassID directory.	SP REST API SP U2F Mobile SDK
Access.AllowPushResponse	Allow the server to respond to push responses from the SurePassID Authenticator.	SP REST API SP Authenticator
Server.Trace	Log all activity that passes through the server. The output is provided in the **Trace** subfolder of the installation folder.	All

Default Language

The system ships with a default language file that is based on US English culture (en-US). The system is Unicode based so it can support every possible language including double byte and right to left character sets.

There are only a few instances when the system will provide the user with a web page. To make these pages culturally friendly, you will need to update the language file settings.

The system will automatically change language to the culture of the user (which is usually set by the underlying operating system) if the appropriate culture (language file) exists for their culture. This has two important uses:

1. Provide a language-centric experience to your users across cultural boundaries.
2. Change any constant field/message in the system to the appropriate language.

If you would like to change the English text of the messages you can modify the **settings.resx** file located in **App_GlobalResources** folder located under the root installation folder. This is the language file for the default culture. To add additional cultures you will need copy the existing **settings.resx** to **settings.xx-yy.resx** where xx is the language and yy is the dialect. For example, is en-us is for the United States en-gb is for Great Britain, fr-fr is france and fr-ca is Canada, etc.). The system will automatically change to the match the default language that is currently selected in the user's browser.