

# Creating Safari Web Push Certificate

Before we begin, ensure you have an **active paid Apple Developer subscription** to create a unique Push ID for your website. The entire process can be divided into four sections:

1. **Creating a Certificate Request (CSR)**
2. **Setting up a Website Push ID**
3. **Generating the Web Push Certificate**
4. **Hosting the push package and integrating it into your app**

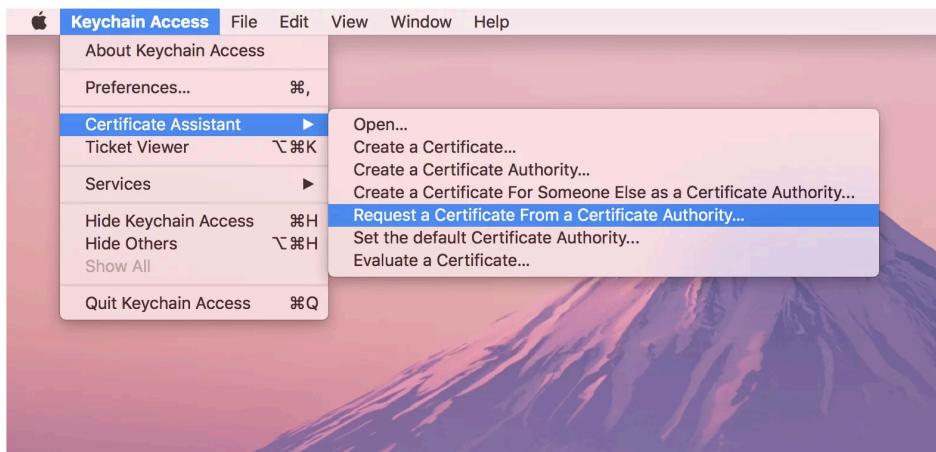
**Note:** With Safari 16 running on macOS Ventura (13) or later, the Web Push Certificate is **no longer required**. Apple now uses the **standard Web Push API**, aligning Safari with browsers like Chrome and Firefox. For Safari 15 or earlier, the web push certificate is still required for permissions.

---

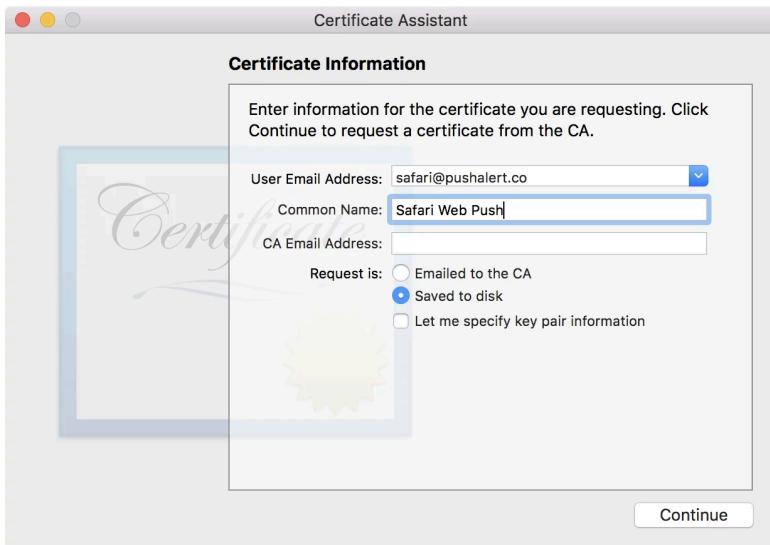
## Step 1: Create a Certificate Request (CSR)

We will create a **Certificate Signing Request (CSR)** using the Keychain Access app.

1. Launch **Keychain Access**.
2. Navigate to: Keychain Access > Certificate Assistant > Request a Certificate From a Certificate Authority.



3. Enter your email address and name.



4. Select "**Saved to disk**", then click **Continue**.
  5. Save the generated .certSigningRequest file. You will need this in later steps.
- 

## Step 2: Set Up Website Push ID for APNS

Now we create a unique Website Push ID for Apple Push Notification Service (APNS).

1. Log in to the **Apple Developer Console**, and go to [Certificates List](#)
2. Go to **Identifiers** → **Register an App ID**.
3. Select **Website Push IDs** from the "Register a New Identifier" list, then click **Continue**.

**Certificates, Identifiers & Profiles**

[< All Identifiers](#)

### Register a New Identifier

1.  **Website Push IDs**  
2. [Continue](#)

**Services IDs**  
For each website that uses Sign In with Apple, register a services identifier (Services ID), configure your domain and return URL, and create an associated private key.

**Pass Type IDs**  
Register a pass type identifier (Pass Type ID) for each kind of pass you create (i.e. gift cards). Registering your Pass Type IDs lets you generate Apple-issued certificates which are used to digitally sign and send updates to your passes, and allow your passes to be recognized by Wallet.

**Website Push IDs**  
Register a Website Push Identifier (Website Push ID). Registering your Website Push IDs lets you generate Apple-issued certificates which are used to digitally sign and send push notifications from your website to macOS.

**iCloud Containers**  
Registering your iCloud Container lets you use the iCloud Storage APIs to enable your apps to store data and documents in iCloud, keeping your apps up to date automatically.

4. Enter a meaningful **Description**.
5. For **Identifier**, follow reverse-domain naming, e.g., `web.com.mywebsite`. > Note: The prefix `web` is added automatically by Apple.
6. Click **Continue**, verify the details, and then click **Register**.

**Certificates, Identifiers & Profiles**

[< All Identifiers](#)

### Register a Website Push ID

1. Identifier: web.com.mywebsite  
2. [Continue](#)

## Step 3: Generate Web Push Certificate

1. Go to **Certificates** and click the **+** icon.

The screenshot shows the 'Certificates, Identifiers & Profiles' section of the Apple Developer portal. On the left, there's a sidebar with links for Certificates, Identifiers, Devices, Profiles, Keys, and More. The 'Certificates' link is highlighted with a blue box. The main area has a heading 'Certificates' with a plus sign and a '2.' next to it. A search bar says 'All Types'. Below is a table with columns: NAME, TYPE, PLATFORM, CREATED BY, and EXPIRATION. Two entries are listed:

NAME	TYPE	PLATFORM	CREATED BY	EXPIRATION
[REDACTED]	Website Push	iOS	Mohit Kuldeep	2020/03/18
[REDACTED]	Website Push	iOS	Mohit Kuldeep	2020/03/09

2. Select **Website Push ID Certificate**, then click **Continue**.
3. Choose the **Website Push ID** created in Step 2, then click **Continue**.

The screenshot shows the 'Create a New Certificate' page. At the top, there's a back link '[All Certificates](#)' and a 'Back' button. The main title is 'Create a New Certificate' with a 'Continue' button to its right. Below is a note: 'Select a Website Push ID for your Website Push Certificate' followed by a detailed explanation: 'Each Website Push ID you register will require its own individual Website Push Certificate. This certificate allows your notification server to connect to the Website Push service.' A 'Website Push ID:' label is followed by a dropdown menu containing the value '.web.com.mywebsite'.

4. Upload the **CSR file** from Step 1 and click **Continue**.

The screenshot shows a web browser window with the URL [developer.apple.com/account/resources/certificates/add](https://developer.apple.com/account/resources/certificates/add). The title bar says "Apple Developer". The main content area is titled "Certificates, Identifiers & Profiles". Below it, a sub-section titled "Create a New Certificate" is shown. A note says "Upload a Certificate Signing Request" and "To manually generate a Certificate, you need a Certificate Signing Request (CSR) file from your Mac." There is a "Choose File" button and a file input field containing "CertificateSigningRequest.certSigningRequest". At the bottom right are "Back" and "Continue" buttons.

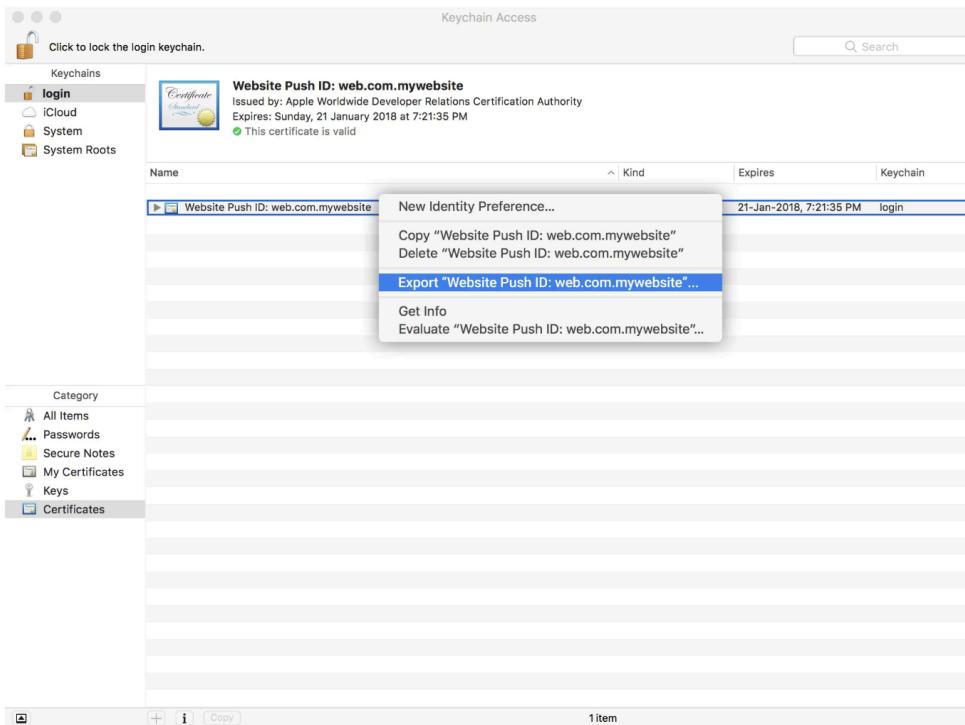
## 5. Download the generated **SSL Certificate**.

The screenshot shows a web browser window with the URL [developer.apple.com/account/resources/certificates/add/d](https://developer.apple.com/account/resources/certificates/add/d). The title bar says "Apple Developer". The main content area is titled "Certificates, Identifiers & Profiles". Below it, a sub-section titled "Download Your Certificate" is shown. A "Download" button is visible. Under "Certificate Details", there are two columns of information: "Certificate Name" (web.com.mywebsite) and "Certificate Type" (Website Push); "Expiration Date" (2020/06/05) and "Created By" (Mohit Kuldeep). A note on the right says "Download your certificate to your Mac, then double click it in Keychain Access. Make sure to save a backup copy of your keys somewhere secure."

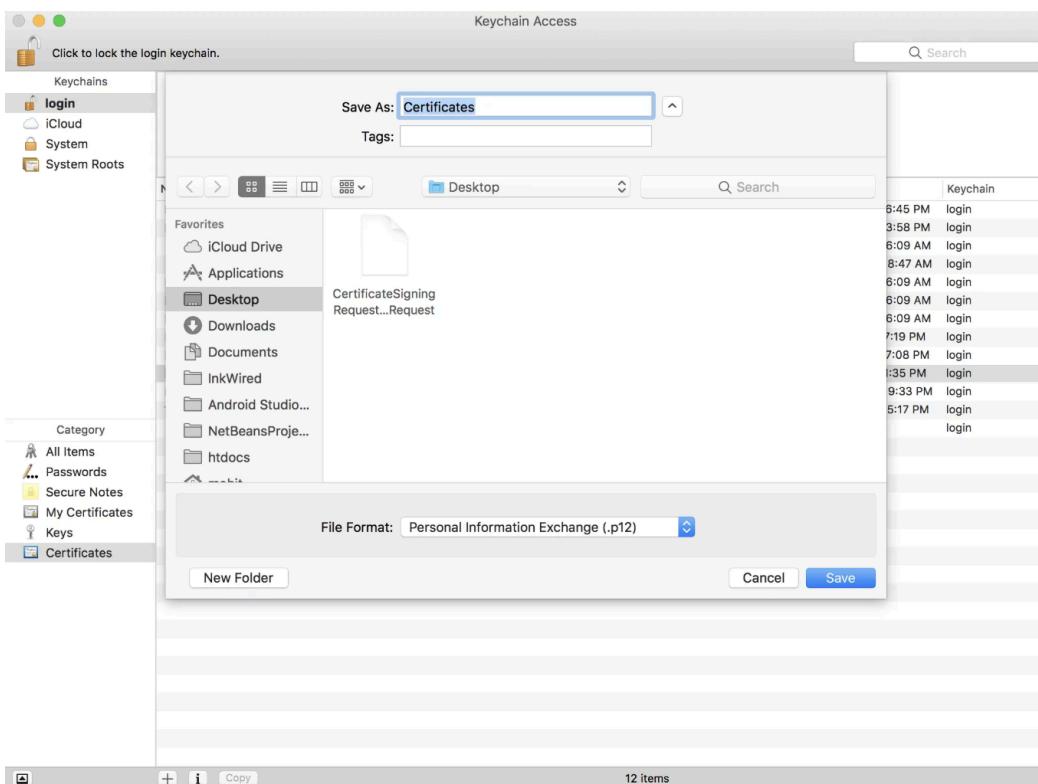
This certificate will be used to send push notifications to subscribers.

## Export the Certificate (.p12)

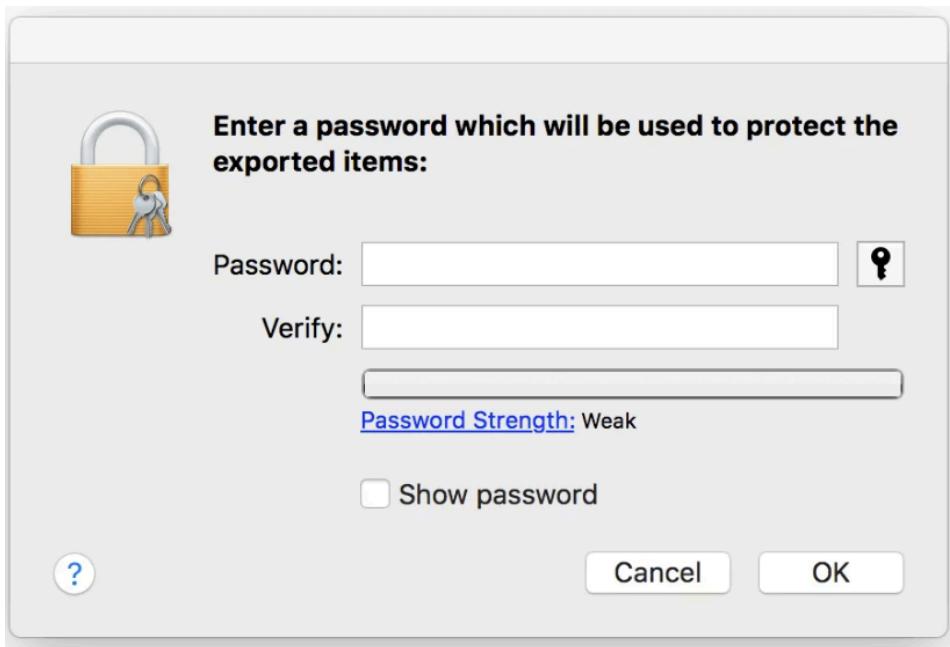
1. Double-click the downloaded certificate to install it in **Keychain Access**.
2. In Keychain Access, select **Certificates** under the Category section.
3. Right-click the certificate associated with your Website Push ID and choose **Export**.



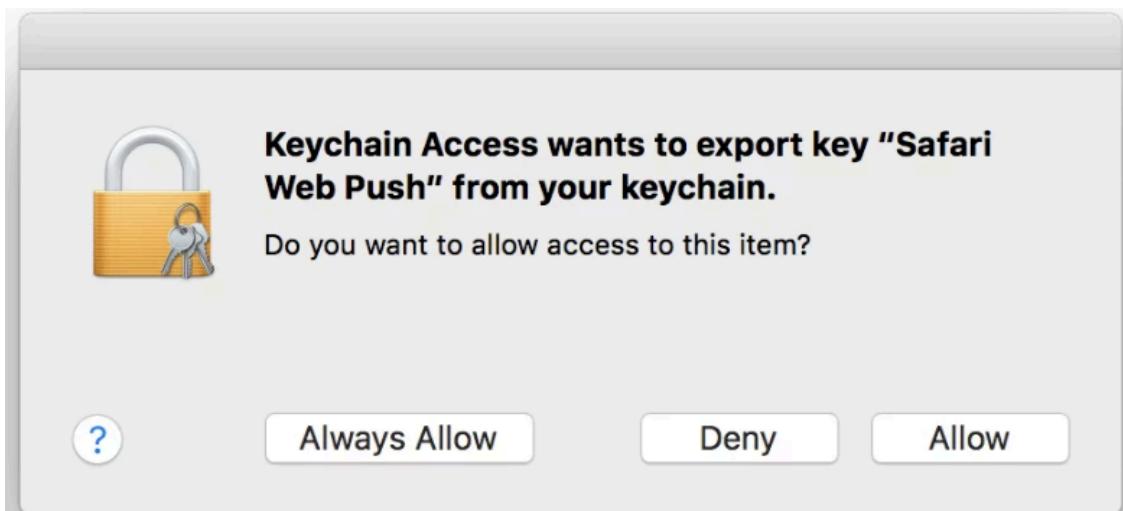
#### 4. Save it as a Personal Information Exchange (.p12) file.



#### 5. When prompted for a password, leave it blank and click OK.



6. Click **Allow** and enter your Mac password to complete the export.



You now have your .p12 certificate ready to upload or use.

---

## Step 4: Create and Host the Push Package

You can now proceed with creating a **push package** using Apple's official PHP companion file.

### Requirements

- PHP 7.4.33 installed locally
- OpenSSL (built-in on macOS)

### Installing PHP 7.4 via Homebrew

```
brew tap shivammathur/php  
brew install shivammathur/php/php@7.4
```

Verify installation:

```
brew list | grep php  
php -v
```

If `php -v` doesn't work, add it to your PATH:

```
export PATH="$(brew --prefix php@7.4)/bin:$PATH"
```

### Download the Companion Script

From Apple's documentation, download the **companion PHP file** (link available at the top right of the page).

This package includes `createPushPackage.php`, which builds the `.zip` push package. However, use the companion file in the code repo to generate the push package as there is a difference in `create_signature` function compared to what is provided by the apple developer script.

### Re-export or Repackage Your .p12 Certificate

If your `.p12` certificate uses **RC2 encryption**, you must re-export it using AES instead.

#### *Convert Using OpenSSL*

```
# Step 1: Extract using Legacy mode  
openssl pkcs12 -legacy -in ReelQA.p12 -out temp.pem  
  
# Step 2: Repackage with AES encryption  
openssl pkcs12 -export -in temp.pem -out ReelQA_AES.p12 \  
-name "Website Push ID: web.com.reelqa" \  
-certpbe AES-256-CBC -keypbe AES-256-CBC
```

## Create the Push Package

```
php createPushPackage.php > pushPackage.zip  
unzip -l pushPackage.zip
```

---

## Step 5: Integrate Push Permission in Your App

Finally, enable Safari Push in your web app. Add a **Call-To-Action (CTA)** to prompt users for notification permission.

Here's an example **React component**:

```
export default function PushNotification() {  
  const { t } = useTranslation('translation', { keyPrefix: 'Common' });  
  const [anchorEl, setAnchorEl] = React.useState<HTMLButtonElement |  
null>(null);  
  const [permissionData, setPermissionData] = useState<any>(  
    window.safari.pushNotification.permission(` ${process.env.WEB_PUSH_ID}`),  
  );  
  const [showIcon, setShowIcon] = useState<boolean>(  
  
    window.safari.pushNotification.permission(` ${process.env.WEB_PUSH_ID}`).permi  
ssion === 'default',  
  );  
  
  const handleClick = (event: React.MouseEvent<HTMLButtonElement>) =>  
    setAnchorEl(event.currentTarget);  
  const handleClose = () => {  
    setAnchorEl(null);  
    setShowIcon(false);  
  };  
  
  const handleSafariPermission = () => {  
    checkRemotePermission(permissionData);  
  };  
  
  const checkRemotePermission = (permissionD: any) => {  
    if (permissionD.permission === 'default') {  
      window.safari.pushNotification.requestPermission(  
        `${process.env.API_URL}notification-service/reel/api/v1/apns`,  
        `${process.env.WEB_PUSH_ID}`,  
        {},  
        checkRemotePermission,  
      );  
    } else if (permissionD.permission === 'denied') {  
    }  
  };  
}
```

```

        setShowIcon(false);
        setPermissionData(permissionD);
        handleClose();
    } else if (permissionD.permission === 'granted') {
        setShowIcon(false);
        setPermissionData(permissionD);
        updateDeviceTokenForSafari(permissionD.deviceToken);
        handleClose();
    }
};

const open = Boolean(anchorEl);
const id = open ? 'notification-popover' : undefined;

return (
    <div className="pushNotification">
        {showIcon && (
            <IconButton aria-describedby={id} onClick={handleClick}>
                <img src={Images.PUSH_NOTIFICATION} alt={t('manage_push_notifications')} />
            </IconButton>
        )}
    <Popover
        id={id}
        open={open}
        anchorEl={anchorEl}
        onClose={handleClose}
        anchorOrigin={{ vertical: 'bottom', horizontal: 'right' }}
        transformOrigin={{ vertical: 280, horizontal: 'right' }}
        className="notification_popover"
    >
        <div className="push_head">
            <h3>{t('manage_push_notifications')}</h3>
            <IconButton onClick={handleClose}>
                <img src={Images.CLOSE} alt={t('manage_push_notifications')} />
            </IconButton>
        </div>
        <div className="push_body">
            <p>{t('push_notifications_message')}</p>
        </div>
        <div className="push_bottom">
            <CustomButton
                onClick={handleClose}
                size="large"
                variant="outlined"
                text={t('reject')}
                width="100%">
        
```

```
        />
      <CustomButton
        onClick={handleSafariPermission}
        size="large"
        variant="contained"
        text={t('allow')}
        width="100%"
      />
    </div>
  </Popover>
</div>
);
}
```

---

## Summary

You've now completed the Safari Web Push setup:

1. Generated and exported a .p12 certificate.
2. Created a Website Push ID and certificate in Apple Developer.
3. Repackaged your certificate with AES encryption.
4. Built and hosted the push package.
5. Integrated the request permission flow in your React app.

You're ready to send push notifications to Safari users on macOS and iOS!