# **Fortanix**<sup>®</sup>

# **User Guide**

# FORTANIX - CUSTODIAL WARM WALLET SOLUTION

**VERSION 1.0** 



# TABLE OF CONTENTS

1.0	INTRODUCTION	2
1.1	Fortanix Custodial Warm Wallet Plugin	2
2.0	DEFINITIONS	3
3.0	SETUP	4
3.1	Create a Fortanix DSM Group	4
3.2	Configure a Quorum Policy for the Group	5
3.3	Create an App in Fortanix DSM	6
3.4	Generate a Master Key	7
3.5	Import the Master Key in Fortanix DSM	8
4.0	PLUGIN OPERATIONS	9
4.1	User Registration	9
4.2	Derive THE PUBLIC KEY	10
4.3	Sign DATA OR Ethereum Transaction	11
5.0	USING NODE.JS BLOCKCHAIN SDK	11
5.1	User Registration	11
5.2	Derive THE ACCOUNT ADDRESS	11
5.3	Sign DATA OR Ethereum Transaction	12
6.0	DOCUMENT INFORMATION	13
6.1	Document Location	13
6.2	Document Updates	13
6.3	Revision HistoryError!	Bookmark not defined.

## 1.0 INTRODUCTION

This document describes how the **Fortanix "Custodial Warm Wallet"** solution provides an additional layer of security to crypto-currency wallets by incorporating a second factor of authentication (2FA) for transaction signing using Time-based One-Time Passwords (TOTP). This solution comprises a plugin that is securely deployed inside Fortanix Data Security Manager (DSM) Software-as-a-Service (SaaS). This solution also comprises a Node.js SDK that makes it easy for wallet backends to interact with Fortanix DSM.

#### 1.1 FORTANIX CUSTODIAL WARM WALLET PLUGIN

The Fortanix "Custodial Warm Wallet" solution implements a Warm Wallet as a Fortanix DSM plugin. The warm wallet supports secure-second factor authentication (2FA) using TOTP, secure key management, and secure transaction signing that enables B2C crypto-currency businesses to ensure that customers' assets are not transferred without their explicit consent.

The plugin is protected by a quorum policy that involves multiple admin users. Once deployed, the plugin code cannot be modified without explicit permissions from multiple administrators.

The plugin performs the following operations:

- Registers users for 2FA using TOTP
- Derives the public key
- Signs data or Ethereum transaction



- The name of the plugin used in the Fortanix "Custodial Warm Wallet" solution is "TOTP ETH Signer".
- The name of the Node.js SDK used in the Fortanix "Custodial Warm Wallet" solution is "fortanix-web3-eth-accounts".

## 2.0 **DEFINITIONS**

#### • Fortanix Data Security Manager

Fortanix DSM is the cloud solution secured with Intel® SGX. With Fortanix DSM, you can securely generate, store, and use cryptographic keys and certificates, as well as secrets, such as passwords, API keys, tokens, or any blob of data.

#### • Accounts

A Fortanix DSM account is the top-level container for security objects managed by the Fortanix DSM. An account is generally associated with an organization, rather than an individual. Security objects, groups, and applications belong to exactly one account. Different accounts are fully isolated from each other. *See <u>support</u> for more information.* 

#### • Fortanix Data Security Manager Security Objects

A security object is any datum stored in Fortanix DSM (for example a key, a certificate, a password, or other security objects). Each security object is assigned to exactly one group. Users and applications assigned to the group have permission to see the security object and to perform operations on it. *See <u>support</u> for more information.* 

## 3.0 SETUP

# 3.1 CREATE A FORTANIX DSM GROUP

 To use the Fortanix "TOTP ETH Signer" plugin in Fortanix DSM, you must first create a Fortanix DSM group, and add the Plugin to this group from the Fortanix DSM Plugin Library.

Fortanix Data Security Manager	
Ø Dashboard Groups	器 Vault (No description)
Apps     Security Objects     Keys, Secrets, Certs	UUID: b5c405db-85ab-4212-9614-57043e914e5e
(2) Users (2) Plugins	INFO USERS (2) APPS (1) SECURITY OBJECTS PLUGINS (2) EXTERNAL ROLES (0)
<ul><li>Tasks</li><li>Audit Log</li></ul>	ADD PLUGIN REMOVE SELECTED Q Search
ී Settings	Plugin title      Description

FIGURE 1: IMPORT PLUGIN

*Refer to the User's Guide: Plugin Library for steps to <u>access</u> and <u>install</u> the plugin from the Fortanix DSM Plugin Library.* 

Fortanix Data Security Manager  Dashboard  Croups  Apps Security Objects	PLUGINS PLUGIN LIBRARY Plugin Library Runtime Encryption® Plugins GitHub Plugin Library	ò					1 + F	
<ul> <li>Keys, Secres, Cetts</li> <li>Users</li> <li>Plugins</li> <li>Tasks</li> <li>Audit Log</li> <li>Settings</li> </ul>	HD Wallet V 1.0 This plugin implements hierarchical deterministic wallets (or "HD Wallets") BIP0032 protocol. By Fortanix Inc.	ψi	SSH CA V 1.0 Issue certificates for SSH authentication By Fortanik Inc.	N)	DUKPT V 1.0 Plugin for importing DUKPT BDKs and for encrypting and decrypting data using the DUKPT procedure.	N/F	X.509 CA V 1.0 Issue X.509 certificates for keys stored in SDKMS. By Fortanik inc.	Ð
	TOTP Ethereum Signer V 1.0 * TOTP Ethereum Signer This plugin implements an Ethereum Signer, Each Ethereum Signer is paired with a "MASTER, KEY." Several wallets can be added to each Ethereum Signer, Several	₩P	Key/Value Pair V 1.0 Seamlessle extend the functionality of Fortanix Self Defending MMS Secrets. Leverage applications generate and manage key-value (N0) pairs through Self Defending KMS Secrets.	lψ.	JWS-JWE Encrypt V 1.0 This plaging enerates a **JSON Web Encryption (IWE)** from the **JSON Web Signature (IWE)** which is constructed using the user's input payload.	N()	JWS-JWE Decrypt V 1.0 Generates the user's payload from a **fSON Web Encryption (JWE)**. By Fortanik Inc.	ų.

FIGURE 2: INSTALL PLUGIN FROM THE PLUGIN LIBRARY



 Copy the UUID of the plugin. When using the "fortanix-web3-eth-accounts" SDK, the environment variable signerId is assigned this UUID.

Fortanix Data Security Manager	
	PD / Plugins / View plugin
🖉 Dashboard	
品 Groups	🐡 TOTP ETH Signer 📖
🗁 Apps	Updated: May 25, 2022, 10:35 pm
Security Objects Keys, Secrets, Certs	UUID: bfd5f265-ca00-40f9-a51d-69ba197110a6
ନ୍ତି Users	
🐡 Plugins	INFO CODE SECURITY OBJECTS
ලි Tasks	■ Enabled ⑦ (No description) Ø
🔄 Audit Log	
ැ <sup>ම්</sup> Settings	Groups
	品 Vault Default group 印
	EDIT GROUPS

FIGURE 3: PLUGIN UUID

## 3.2 CONFIGURE A QUORUM POLICY FOR THE GROUP

After creating the Fortanix DSM group and adding the "TOTP ETH Signer" plugin to this group, configure a Quorum Policy for the group to protect the plugin. This will ensure that the plugin code cannot be modified without the approval of the Group Administrator.

- 1. Go to the detailed view of the group, and click the **INFO** tab.
- 2. In the **Quorum approval policy** section, click **ADD POLICY** to add a new quorum policy.
- 3. Configure the Quorum approval policy and click **SAVE POLICY**.



Fortanix Data Security Manager	磊 Vault (No description)	
② Dashboard 聶 Groups	UUID: b5c405db-85ab-4212-9614-57043e914e5e	
Apps	INFO USERS (2) APPS (1) SECURITY OBJECTS PLUGINS (2) EXTERNAL ROLES (0)	
Security Objects Keys, Secrets, Certs  Users  Plugins	Quorum approval policy Sensitive operations with keys and plugins within the group require approval from:	
⑦ Tasks 贤 Audit Log	1 of the following administrators	
© Settings	User Fortanix ×	
	Profile password re-entry is required to approve request	
	Operations that require Quorum approval	
	Image: State Program     Image: State Program       Image: State Program     Ima	
	SAVE POLICY CANCEL	

FIGURE 4: CONFIGURE QUORUM APPROVAL POLICY

## 3.3 CREATE AN APP IN FORTANIX DSM

Create an app in Fortanix DSM for the Ethereum Signer and copy the app's API KEY. When using "fortanix-web3-eth-accounts" SDK, the API Key of this application is used as the value of the environment variable signerAccessToken to interact with Fortanix DSM for signing the crypto transactions.

#### CUSTODIAL WARM WALLET SOLUTION



FIGURE 5: CREATE AN APP AND COPY THE API KEY

## 3.4 GENERATE A MASTER KEY

The "TOTP ETH Signer" plugin is paired with a MASTER\_KEY. To use the plugin in Fortanix DSM, you must first manually generate this MASTER KEY and initialize the plugin.

You may use the following JavaScript code snippet to generate a master key:

```
const bip39 = require('bip39')
const bitcore = require('bitcore-lib')
const bitcoin = require('bitcoinjs-lib')
const bip32utils = require('bip32-utils')
let mnemonic = bip39.generateMnemonic()
let seed = bip39.mnemonicToSeedSync(mnemonic)
var xprv = bitcore.HDPrivateKey.fromSeed(seed);
console.log("MASTER_KEY: " + xprv.xprivkey)
```



Here is a sample master key:

MASTER\_KEY =

xprv9s21ZrQH143K31xYSDQpPDxsXRTUcvj2iNHm5NUtrGiGG5e2DtALGdso3pGz6ssrdK4PFmM8 NSpSBHNqPqm55Qn3LqFtT2emdEXVYsCzC2U

#### 3.5 IMPORT THE MASTER KEY IN FORTANIX DSM

After manually generating the master key, import this key into Fortanix DSM SaaS as a Secret Raw key in the group created in *Section 3.1*.



	MASTER_KEY			+ ADD DESCRIPTION				
② Dashboard				_				
Integrations	This is an HSM/external KMS object ⑦							
🚠 Groups	磊 Vault		~					
🕞 Apps								
Security Objects Keys, Secrets, Certs		⊖ generate ⊙						
(2) Users	Import Key from Components ⑦							
Plugins	Choose a type							
ලි Tasks	Certain types may be disabled due to the cryptographic po	icy.						
🖺 Audit Log	O AES O DES3	O HMAC	O OPAQUE					
් Settings	O RSA O DSA	O DES	O EC	SECRET				
	O Certificate 🗈 🔿 BIP32		O EC-KCDSA	⊖ KCDSA				
	SEED 💿							
	The key has been encrypted To import an encrypted key in a file or as a blob, select the corresponding KEK that was previously used to encrypt your target key.							
	Place value here or import from fi	le:						
	Attribute: Example: Password							
	Choose value format: O Text (UTF-8)	🔿 Hex 🔿 Base64 💿 Ra	w					
	MASTER_KEY.txt DELETE							

FIGURE 6: IMPORT MASTER KEY



Fortanix Data Security Manager	■ *	۵ ۵	<u>(</u>	User Fortanix 🗸
<ul> <li>Ø Dashboard</li> <li>Groups</li> <li>Apps</li> <li>Security Objects key, Sores Carts</li> <li>Users</li> </ul>	PD / Groups / View group <b>E Vault</b> (No description) UUID: b5c405db-85ab-4212-9614-57043e914e5e Activity chart			
Plugins	INFO USERS (2) APPS (1) SECURITY OBJECTS PLUGINS (2) EXTERNAL ROLES (3)			
🕐 Tasks	Q. Search			
Audit Log	ADD SECURITY OBJECT 📋 DELETE SELECTED 🛞 DESTROY SELECTED 🛞 DISABLE LOGGING 🗋 ENABLE LOGGING			🕁 DOWNLOAD CSV
i <del>Je</del> ttings	NAME A KCV KEYOPS CREATED BY CREATED TYPE	SIZE/CURVE	EXPIRES	0
	AASTER_KEY	888	Never	÷

#### FIGURE 7: MASTER KEY IMPORTED

## 4.0 PLUGIN OPERATIONS

#### 4.1 USER REGISTRATION

At user registration time, the TOTP authentication system generates a token.

The plugin must be invoked using walletName (Label) as the input for deriving the shared token during registration. For example: walletName= `alice@acme.com'

After the registration is successful:

• A security object of type HMAC is created in Fortanix DSM for the wallet name provided. For example: "totp/alice@acme.com"



Fortanix Data Security Manager			1 🛕 @	User Fortanix 🗸
Croups Croups Apps Security Objects Keys, Secure, Cerrs Users Users	PD / Plugins / Wew plugin TOTP ETH Signer Lua Updated: May 25, 2022, 10:35 pm UUID: bfd5f265-ca00-40f9-a51d-69ba197110a6 C COPY UUID & COPY URL © INVOKE PLUCIN			
<ul> <li>Plugins</li> <li>Tasks</li> </ul>	INFO CODE SECURITY OBJECTS			
Audit Log Settings	C Search			
	NAME A KCV KEYOPS CROUP CREATED TYPE	SIZE/CURVE	EXPIRES	÷
	■ totp/alice@acme.com - 👬 Za Vault 14 hours ago HMAC 🖸	120	Never	:
	ltems per page: 10 ▼	<b>«</b> (<	PREVIOUS	NEXT >

FIGURE 8: SECURITY OBJECT CREATED

• A TOTP path containing the Secret, Label, and Issuer is returned. For example:

otpauth://totp/alice%40acme.com?secret=DZKFQ3J4DRUOL7MY2HR7BJ7M&issuer=Forta
nix%20DSM

where,

- Secret: DZKFQ3J4DRUOL7MY2HR7BJ7M
- o Label: <u>alice@acme.com</u>
- o **Issuer:** Fortanix DSM

The Secret, Label, and Issuer information can be used by a QR code generator to generate a QR code. For a quick check, try generating a QR code with the QR code generator such as <a href="https://stefansundin.github.io/2fa-qr/">https://stefansundin.github.io/2fa-qr/</a>.

#### 4.2 DERIVE THE PUBLIC KEY

Given a walletName and a keyIndex, the plugin can be used to retrieve the corresponding public key. For example, "walletName": alice@acme.com, "keyIndex": "0". The retrieved public key is used by the "fortanix-web3-eth-accounts" SDK to retrieve the account address.

## 4.3 SIGN DATA OR ETHEREUM TRANSACTION

The plugin can be used to sign Ethereum transactions or arbitrary data. If a user has been registered for 2FA, a TOTP code must be provided along with the signing request. The transaction is signed using the following input payload:

```
{
    "operation": "sign",
    "walletName": "string",
    "keyIndex": "number as string",
    "msgHash": "<32-Byte-Message-Hash>"
    "code": "number as string" // code to be provided only if wallet is
    registered for 2FA.
}
```

Where, msgHash is a 32-byte message hash that is signed by the "TOTP ETH Signer" plugin. The message hash is generated by the Node.js blockchain SDK.

## 5.0 USING NODE.JS BLOCKCHAIN SDK

#### 5.1 USER REGISTRATION

During user registration time, the TOTP authentication system generates a token.

The SDK must be invoked using walletName (Label) as the input for deriving the shared token during registration. For example: walletName= `alice@acme.com'. See the example code here.

After the registration is successful:

• A TOTP path containing the Secret, Label, and Issuer is returned.

#### 5.2 DERIVE THE ACCOUNT ADDRESS

Given a walletName and a keyIndex, the SDK can be used to retrieve the corresponding account address. See the example code <u>here</u>.

# Fortanix<sup>®</sup>

## 5.3 SIGN DATA OR ETHEREUM TRANSACTION

The SDK can be used to sign Ethereum transactions or arbitrary data. If a user has been registered for 2FA, a TOTP code has to be provided along with the signing request. See the example code <u>here</u>.

# 6.0 DOCUMENT INFORMATION

#### 6.1 DOCUMENT LOCATION

The latest published version of this document is located at the URL:

https://support.fortanix.com/hc/en-us/articles/6677033204756-Using-Fortanix-Custodial-Warm-

Wallet-Solution

#### 6.2 DOCUMENT UPDATES

This document will typically be updated on a periodic review and update cycle.

For any urgent document updates, please send an email to: <a href="mailto:support@fortanix.com">support@fortanix.com</a>

© 2016 – 2023 Fortanix, Inc. All Rights Reserved.

Fortanix® and the Fortanix logo are registered trademarks or trade names of Fortanix, Inc. All other trademarks are the property of their respective owners.

**NOTICE**: This document was produced by Fortanix, Inc. (Fortanix) and contains information which is proprietary and confidential to Fortanix. The document contains information that may be protected by patents, copyrights, and/or other IP laws. If you are not the intended recipient of this material, please destroy this document and inform <u>info@fortanix.com</u> immediately.