StrongKey FIDO Server (SKFS) 4.4.1 Release Notes

1.1—New Features in SKFS 4.4.1

1.1.1—Cross-domain Authentication

StrongKey has created the *changeusername* web service to enable administrators to change the username associated with registered FIDO keys. This is useful in the following use case.

When enterprises use *Active Directory (AD)* to manage users, a user (johndoe@strongkey.net) may be registered under a specific domain with a few FIDO keys associated to that UPN. With *changeusername* enabled, if that user moves to a different domain (strongkey.local) in the company, the UPN changes to johndoe@strongkey.local. Instead of re-registering FIDO keys for the new UPN, an administrator can use this web service to change the username associated with registered FIDO keys from the old to new (johndoe@strongkey.net → johndoe@strongkey.local). In AD the pre-Windows 2000 username attribute is *sAMAccountName*; other systems may have different nomenclature for the pre-Windows 2000 username value.

Though we have described a use case for AD here, administrators and developers may find other scenarios where this functionality will be beneficial.

1.2—Fixes and Changes in SKFS 4.4.1

#	Explanation
DEV-1956	**Update the tutorial to accept JWT authentication response. Update the demo to only use username or email at initial registration; update basic demo design.** *Tutorial:* - Updated the authentication response check to handle the return of a JWT from SKFS - Added the useragent to *strongkeymetadata* during authentication request to SKFS *Demo:* - Changed the name from basic server to basic demo - Changed the registration process to only ask for username or email - Changed the layout to display longer usernames when logged in - Added a notification when clicking on 'register' to tell the user if their device/ browser combination supports platform keys or if they need a **security key**

#	Explanation
DEV-1957	**Update install and upgrade scripts for 4.4.1.** ➢ Added change username configuration ➢ Added Java downgrade until a fix is found for the latest java-1.8.0-openjdk and OpenDJ-3.0.0 ➢ Added StrongKey FIDO Server (SKFS) version to distribution for install/upgrade ➢ Tested the upgrade from all previous SKFS releases to 4.4.1
DEV-1959	**Update username web service.** To accommodate UPN/AD domain switching, we have added a web service that changes a user's username associated with registered FIDO keys. A server property now enables the web service. For more details, check out ***Chapter 3—Administration*** in the *SKFS Administration Guide*.
DEV-1960	**Update the *usersession* flush property.** There is a small discrepancy in the name of the property that determines the age of the user session object in memory. KA and SKFS use different names for the same property; this needs to be changed so that they are consistent with each other. The property name has been updated in the code. If this property was previously overridden by the `/usr/local/strongkey/skce/etc/skce-configuration.properties` file then the name of the property should be changed. ➢ Old name: `skce.cfg.property.usersession.flush.cutofftime.seconds` ➢ New name: `skfe.cfg.property.usersession.flush.cutofftime.seconds`

1.3—New Features in SKFS 4.4

1.1.2—StrongKey Android Client Library (SACL) Preview Release 1

The StrongKey Android Client Library is an open source, native Android library providing support for the FIDO2 protocol for native Android apps. It provides the following features:

- It is supported on Android 9 (API 28) "Pie" or greater
- It supports the Java programming language, and does not require the use of JavaScript or the WebView component to deliver FIDO capability. Note that it does not support the use of external *Security Keys*—only platform keys
- It uses the *AndroidKeystore*—taking advantage of the *Trusted Execution Environment (TEE)* or a *Secure Element (SE)*, whichever is present—for key generation, storage and usage. It is always used as a *user verifying platform authenticator (UVPA)*. Devices without the TEE or SE cannot install apps using the SACL
- It supports *registration, authentication* and *transaction authorization* using "dynamic linking"—a core requirement of the European Union's Payment Services Directive 2 regulation for *Strong Customer Authentication (SCA)*
- Supports Android *BiometricPrompt* API for verifying users before enabling use of FIDO keys
- It has out-of-the-box integration with the open-source FIDO® Certified *StrongKey FIDO Server (SKFS)*—just add your mobile app to the flow

- It includes a sample e-Commerce web application—the *Sample FIDO App for e-Commerce (SFAECO)*—to demonstrate 4 basic functions:
  - User enrollment
  - FIDO registration
  - FIDO authentication and
  - User confirmation of business transactions with the user's registered FIDO key
- The server side components of the SFAECO app are available as a *Java Enterprise Edition (JEE)* application to support the mobile sample app. This JEE application makes web service requests of the SKFS
- It includes a sample browser based web application—*Back Office Application (BOA)*—to work in concert with the SFAECO app to perform sample back-office business functions. But, the primary purpose is to demonstrate the use of FIDO for strong authentication and to review business transactions performed by app users, as well as see data collected by the app when performing *transaction confirmation (TXC)*
- It includes a second sample browser based web application—FIDO Key Management—to demonstrate the newly announced *single sign-on (SSO)* capability of the SKFS with *JSON Web Token (JWT)* using X.509 based *JSON Web Signatures (JWS)*
- These three web applications have been installed on a demo server on the internet (https://psd2demo.strongkey.com) against which the mobile app makes REST web service calls

SACL has been tested using Essential PH-1, Google Pixel 3a, and Google Pixel 4a phones—the first two running Android 9 (Pie) with API 28, and the Pixel 4a with Android R (API 30). The device must have a fingerprint enrolled to support the use of SACL. While it is likely to work on most Android devices with biometric capability, your mileage may vary.

1.1.3—JSON Web Tokens (JWTs)

SKFS now returns a *JSON web token (JWT)* upon authentication. A JWT is a self contained token that can be used between parties to provide trust. When a user requests an authentication with a *relying party (RP)*, the RP sends a request to the FIDO server. The FIDO server then verifies the user's identity based on the FIDO2 protocol. If the user identity is authenticated then the FIDO server creates a JWT and returns it to the RP as part of the response. Then the RP returns the JWT to the user. This user is then able to use the JWT for a predetermined length of time (default 30 minutes). Also, there is a independent program used for verifying the JWTs. This program is used by any service that wants to accept the FIDO server JWT as proof of identity. Upon receiving the request the service will pass the JWT, along with other data about the user, to the JWT verification program. That will then return whether or not the JWT is valid for the user. The advantage of using the JWT is it allows the user to only require verifying their identity with FIDO once per session while allowing them to access any service that has been configured to accept and verify the FIDO server's JWTs.

1.4—Fixes and Changes in SKFS 4.4

#	Explanation
DEV-1941	**Update to Payara 5 and MariaDB 10.5.8** Upgraded from Payara 4.1 and MariaDB 10.2.3. Hardware dependencies and documentation have been updated accordingly: ▸ New configurations; new configurations table ▸ Changes made to the FIDO policy in the database ▸ Signing keystore changes ▸ Generates JWT signing certs ▸ Updated `fido_keys` and removed *khdigest* and *khddigestalgo* columns and added *signature key type* columns
DEV-1932	**Load JSON web Token (JWT) certificates into memory.** JWT keys are now created on install and are loaded in memory for the purpose of generating a JWT for an authenticate response. This is similar to an earlier feature which allowed for signing keys to be loaded into memory for the cryptomodule.
DEV-1931	**Create JSON web Token (JWT) signing certificate generator.** The application now generates a self-signed CA certificate that is then used to issue JWT signing certificates. All generated certificates are created and imported into `jwtsigningkeystore.bcfks` and `jwtsigningtruststore.bcfks`.
DEV-1930	**Add get all policy web service.** Add get all policies metadata web service call to populate a policy list.
DEV-1928	**Implement FIDO policy JSON system change.** System element is now a peer element of algorithms, attestation, registration, authentication, authorization, and rp; not sub-elements of system.
DEV-1925	**Add a log for API calls recieved from username for policy web service.** Added info in the log that specifies the username passed by the policy web service call.