

NIST SPECIAL PUBLICATION 1800-13C

Mobile Application Single Sign-On

Improving Authentication for Public Safety First Responders

Volume C: How-To Guides

Paul Grassi

Applied Cybersecurity Division
Information Technology Laboratory

Bill Fisher

National Cybersecurity Center of Excellence
Information Technology Laboratory

Santos Jha

William Kim

Taylor McCorkill

Joseph Portner

Mark Russell

Sudhi Umarji

The MITRE Corporation
McLean, VA

April 2018

DRAFT

This publication is available free of charge from:
<https://www.nccoe.nist.gov/projects/use-cases/mobile-sso>

DISCLAIMER

Certain commercial entities, equipment, products, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST or NCCoE, nor is it intended to imply that the entities, equipment, products, or materials are necessarily the best available for the purpose.

National Institute of Standards and Technology Special Publication 1800-13C, Natl. Inst. Stand. Technol. Spec. Publ. 1800-13C, 163 pages, (April 2018), CODEN: NSPUE2

FEEDBACK

You can improve this guide by contributing feedback. As you review and adopt this solution for your own organization, we ask you and your colleagues to share your experience and advice with us.

Comments on this publication may be submitted to: psfr-nccoe@nist.gov.

Public comment period: April 16, 2018 through June 18, 2018

All comments are subject to release under the Freedom of Information Act (FOIA).

National Cybersecurity Center of Excellence
National Institute of Standards and Technology
100 Bureau Drive
Mailstop 2002
Gaithersburg, MD 20899
Email: nccoe@nist.gov

NATIONAL CYBERSECURITY CENTER OF EXCELLENCE

The National Cybersecurity Center of Excellence (NCCoE), a part of the National Institute of Standards and Technology (NIST), is a collaborative hub where industry organizations, government agencies, and academic institutions work together to address businesses' most pressing cybersecurity issues. This public-private partnership enables the creation of practical cybersecurity solutions for specific industries, as well as for broad, cross-sector technology challenges. Through consortia under Cooperative Research and Development Agreements (CRADAs), including technology partners—from Fortune 50 market leaders to smaller companies specializing in IT security—the NCCoE applies standards and best practices to develop modular, easily adaptable example cybersecurity solutions using commercially available technology. The NCCoE documents these example solutions in the NIST Special Publication 1800 series, which maps capabilities to the NIST Cyber Security Framework and details the steps needed for another entity to re-create the example solution. The NCCoE was established in 2012 by NIST in partnership with the State of Maryland and Montgomery County, Md.

To learn more about the NCCoE, visit <https://nccoe.nist.gov>. To learn more about NIST, visit <https://www.nist.gov>.

NIST CYBERSECURITY PRACTICE GUIDES

NIST Cybersecurity Practice Guides (Special Publication Series 1800) target specific cybersecurity challenges in the public and private sectors. They are practical, user-friendly guides that facilitate the adoption of standards-based approaches to cybersecurity. They show members of the information security community how to implement example solutions that help them align more easily with relevant standards and best practices and provide users with the materials lists, configuration files, and other information they need to implement a similar approach.

The documents in this series describe example implementations of cybersecurity practices that businesses and other organizations may voluntarily adopt. These documents do not describe regulations or mandatory practices, nor do they carry statutory authority.

ABSTRACT

On-demand access to public safety data is critical to ensuring that public safety and first responder (PSFR) personnel can deliver the proper care and support during an emergency. This requirement necessitates heavy reliance on mobile platforms while in the field, which may be used to access sensitive information, such as personally identifiable information (PII), law enforcement sensitive (LES) information, or protected health information (PHI). However, complex authentication requirements can hinder the process of providing emergency services, and any delay—even seconds—can become a matter of life or death.

In collaboration with NIST'S Public Safety Communications Research lab (PSCR) and industry stakeholders, the NCCoE aims to help PSFR personnel to efficiently and securely gain access to mission data via mobile devices and applications (apps). This practice guide describes a reference design for multifactor authentication (MFA) and mobile single sign-on (MSSO) for native and web apps, while improving interoperability between mobile platforms, apps, and identity providers, irrespective of the app development platform used in their construction. This NCCoE practice guide details a collaborative effort between the NCCoE and technology providers to demonstrate a standards-based approach using commercially available and open-source products.

This guide discusses potential security risks facing organizations, benefits that may result from the implementation of an MFA/MSSO system, and the approach that the NCCoE took in developing a reference architecture and build. This guide includes a discussion of major architecture design considerations, an explanation of the security characteristics achieved by the reference design, and a mapping of the security characteristics to applicable standards and security control families.

For parties interested in adopting all or part of the NCCoE reference architecture, this guide includes a detailed description of the installation, configuration, and integration of all components.

KEYWORDS

access control; authentication; authorization; identity; identity management; identity provider; single sign-on; relying party

ACKNOWLEDGMENTS

We are grateful to the following individuals for their generous contributions of expertise and time.

Name	Organization
Donna Dodson	NIST NCCoE
Tim McBride	NIST NCCoE
Jeff Vettraino	FirstNet
FNU Rajan	FirstNet
John Beltz	NIST Public Safety Communications Research Lab
Chris Leggett	Ping Identity

Name	Organization
Paul Madsen	Ping Identity
John Bradley	Yubico
Adam Migus	Yubico
Derek Hanson	Yubico
Adam Lewis	Motorola Solutions
Mike Korus	Motorola Solutions
Dan Griesmann	Motorola Solutions
Arshad Noor	StrongAuth
Pushkar Marathe	StrongAuth
Max Smyth	StrongAuth
Scott Wong	StrongAuth
Akhilesh Sah	Nok Nok Labs
Avinash Umap	Nok Nok Labs

The Technology Partners/Collaborators who participated in this build submitted their capabilities in response to a notice in the Federal Register. Respondents with relevant capabilities or product components were invited to sign a Cooperative Research and Development Agreement (CRADA) with NIST, allowing them to participate in a consortium to build this example solution. We worked with:

Technology Partner/Collaborator	Build Involvement
Ping Identity	Federation Server

Technology Partner/Collaborator	Build Involvement
Motorola Solutions	Mobile Apps
Yubico	External Authenticators
Nok Nok Labs	Fast Identity Online (FIDO) Universal Authentication Framework (UAF) Server
StrongAuth	FIDO Universal Second Factor (U2F) Server

1 Contents

2	1 Introduction	1
3	1.1 Practice Guide Structure	1
4	1.2 Build Overview	1
5	1.2.1 Usage Scenarios	2
6	1.2.2 Architectural Overview	3
7	1.2.3 General Infrastructure Requirements.....	5
8	1.3 Typographic Conventions.....	5
9	2 How to Install and Configure the Mobile Device.....	6
10	2.1 Platform and System Requirements	6
11	2.1.1 Supporting SSO	7
12	2.1.2 Supporting FIDO U2F	8
13	2.1.3 Supporting FIDO UAF	8
14	2.2 How to Install and Configure the Mobile Apps	9
15	2.2.1 How to Install and Configure SSO-Enabled Apps.....	9
16	2.2.2 How to Install and Configure a FIDO U2F Authenticator.....	20
17	2.2.3 How to Install and Configure a FIDO UAF Client.....	22
18	2.3 How App Developers Must Integrate AppAuth for SSO.....	30
19	2.3.1 Adding the Library Dependency	31
20	2.3.2 Adding Activities to the Manifest	31
21	2.3.3 Create Activities to Handle Authorization Responses	32
22	2.3.4 Executing the OAuth 2 Authorization Flow	36
23	2.3.5 Fetching and Using the Access Token.....	38
24	3 How to Install and Configure the OAuth 2 AS	39
25	3.1 Platform and System Requirements	39
26	3.1.1 Software Requirements	39
27	3.1.2 Hardware Requirements.....	39
28	3.1.3 Network Requirements.....	40
29	3.2 How to Install the OAuth 2 AS.....	41

30	3.2.1 Java Installation.....	41
31	3.2.2 Java Post Installation.....	41
32	3.2.3 PingFederate Installation	43
33	3.2.4 Certificate Installation.....	43
34	3.3 How to Configure the OAuth 2 AS.....	43
35	3.4 How to Configure the OAuth 2 AS for Authentication.....	57
36	3.4.1 How to Configure Direct Authentication	58
37	3.4.2 How to Configure SAML Authentication.....	67
38	3.4.3 How to Configure OIDC Authentication.....	74
39	3.4.4 How to Configure the Authentication Policy	81
40	4 How to Install and Configure the Identity Providers	87
41	4.1 How to Configure the User Store	87
42	4.2 How to Install and Configure the SAML Identity Provider	91
43	4.2.1 Configuring Authentication to the IdP	93
44	4.2.2 Configure the SP Connection	103
45	4.3 How to Install and Configure the OIDC Identity Provider	110
46	4.3.1 Configuring Authentication to the OIDC IdP.....	111
47	4.3.2 Configuring the OIDC Client Connection.....	123
48	5 How to Install and Configure the FIDO UAF Authentication Server ...	125
49	5.1 Platform and System Requirements	126
50	5.1.1 Hardware Requirements.....	126
51	5.1.2 Software Requirements	126
52	5.2 How to Install and Configure the FIDO UAF Authentication Server	127
53	5.3 How to Install and Configure the FIDO UAF Gateway Server	128
54	5.4 How to Install and Configure the FIDO UAF Adapter for the OAuth 2 AS	128
55	6 How to Install and Configure the FIDO U2F Authentication Server ...	129
56	6.1 Platform and System Requirements	129
57	6.1.1 Software Requirements	129
58	6.1.2 Hardware Requirements.....	130

59	6.1.3 Network Requirements.....	130
60	6.2 How to Install and Configure the FIDO U2F Authentication Server.....	131
61	6.3 How to Install and Configure the FIDO U2F Adapter for the IdP	135
62	6.3.1 FIDO U2F Registration in Production.....	136
63	7 Functional Tests.....	136
64	7.1 Testing FIDO Authenticators	136
65	7.2 Testing FIDO Servers.....	137
66	7.3 Testing IdPs.....	137
67	7.4 Testing the AS.....	143
68	7.5 Testing the Application.....	145
69	Appendix A Abbreviations and Acronyms.....	146
70	Appendix B References.....	149

71 **List of Figures**

72	Figure 1-1 Lab Build Architecture.....	3
73	Figure 2-1 Comparison of UAF and U2F Standards.....	7
74	Figure 2-2 FIDO UAF Architectural Overview	9
75	Figure 2-3 PSX Cockpit Setup	10
76	Figure 2-4 PSX Cockpit Setup, Continued.....	11
77	Figure 2-5 PSX Cockpit Group List Selection.....	12
78	Figure 2-6 PSX Cockpit Groups	13
79	Figure 2-7 PSX Cockpit Group List Setup Complete	14
80	Figure 2-8 PSX Cockpit User Interface	15
81	Figure 2-9 PSX Mapping User Interface	16
82	Figure 2-10 PSX Mapping Group Member Information	17
83	Figure 2-11 PSX Messenger User Interface	18
84	Figure 2-12 PSX Messenger Messages.....	19

85	Figure 2-13 FIDO U2F Registration	21
86	Figure 2-14 FIDO U2F Authentication.....	22
87	Figure 2-15 Nok Nok Labs Tutorial App Authentication.....	24
88	Figure 2-16 Nok Nok Labs Tutorial App Login	25
89	Figure 2-17 FIDO UAF Registration Interface	26
90	Figure 2-18 FIDO UAF Registration QR Code	27
91	Figure 2-19 FIDO UAF Registration Device Flow.....	28
92	Figure 2-20 FIDO UAF Fingerprint Authenticator	29
93	Figure 2-21 FIDO UAF Registration Success	30
94	Figure 3-1 Access Token Attribute Mapping Framework.....	44
95	Figure 3-2 Server Roles for AS.....	47
96	Figure 3-3 Federation Info	48
97	Figure 3-4 AS Settings.....	49
98	Figure 3-5 Scopes	51
99	Figure 3-6 Access Token Management Instance	52
100	Figure 3-7 Access Token Manager Instance Configuration	53
101	Figure 3-8 Access Token Manager Attribute Contract	54
102	Figure 3-9 OAuth Client Registration, Part 1.....	55
103	Figure 3-10 OAuth Client Registration, Part 2	56
104	Figure 3-11 Create Adapter Instance.....	59
105	Figure 3-12 FIDO Adapter Settings	60
106	Figure 3-13 FIDO Adapter Contract	61
107	Figure 3-14 FIDO Adapter Instance Summary	62
108	Figure 3-15 Policy Contract Information.....	63
109	Figure 3-16 Policy Contract Attributes.....	63
110	Figure 3-17 Create Authentication Policy Contract Mapping	64
111	Figure 3-18 Authentication Policy Contract Fulfillment.....	65
112	Figure 3-19 Create Access Token Attribute Mapping	66

113	Figure 3-20 Access Token Mapping Contract Fulfillment	66
114	Figure 3-21 Create IdP Connection	68
115	Figure 3-22 IdP Connection Options	68
116	Figure 3-23 IdP Connection General Info	69
117	Figure 3-24 IdP Connection – User-Session Creation	70
118	Figure 3-25 IdP Connection OAuth Attribute Mapping	71
119	Figure 3-26 IdP Connection – Protocol Settings	72
120	Figure 3-27 Policy Contract for SAML RP	73
121	Figure 3-28 Contract Mapping for SAML RP	74
122	Figure 3-29 IdP Connection Type	75
123	Figure 3-30 IdP Connection Options	75
124	Figure 3-31 IdP Connection General Info	76
125	Figure 3-32 IdP Connection Authentication Policy Contract	77
126	Figure 3-33 IdP Connection Policy Contract Mapping	78
127	Figure 3-34 IdP Connection OAuth Attribute Mapping	79
128	Figure 3-35 IdP Connection Protocol Settings	80
129	Figure 3-36 IdP Connection Activation and Summary	81
130	Figure 3-37 Authentication Selector Instance	82
131	Figure 3-38 Authentication Selector Details	83
132	Figure 3-39 Selector Result Values	84
133	Figure 3-40 Policy Settings	84
134	Figure 3-41 Authentication Policy	85
135	Figure 3-42 Policy Contract Mapping for IdP Connections	86
136	Figure 3-43 Policy Contract Mapping for Local Authentication	87
137	Figure 4-1 Active Directory Users and Computers	88
138	Figure 4-2 Server Configuration	89
139	Figure 4-3 Data Store Type	90
140	Figure 4-4 LDAP Data Store Configuration	91

141	Figure 4-5 Server Roles for SAML IdP	92
142	Figure 4-6 SAML IdP Federation Info	93
143	Figure 4-7 Create Password Credential Validator.....	94
144	Figure 4-8 Credential Validator Configuration	95
145	Figure 4-9 Password Credential Validator Extended Contract	96
146	Figure 4-10 Password Validator Summary.....	97
147	Figure 4-11 HTML Form Adapter Instance	98
148	Figure 4-12 Form Adapter Settings.....	99
149	Figure 4-13 Form Adapter Extended Contract	100
150	Figure 4-14 Create U2F Adapter Instance	101
151	Figure 4-15 U2F Adapter Settings.....	102
152	Figure 4-16 IdP Authentication Policy	103
153	Figure 4-17 SP Connection Type.....	104
154	Figure 4-18 SP Connection General Info	105
155	Figure 4-19 SP Browser SSO Profiles	106
156	Figure 4-20 Assertion Identity Mapping	107
157	Figure 4-21 Assertion Attribute Contract.....	107
158	Figure 4-22 Assertion Attribute Contract Fulfillment	108
159	Figure 4-23 Browser SSO Protocol Settings.....	109
160	Figure 4-24 OIDC IdP Roles	110
161	Figure 4-25 Create Access Token Manager	112
162	Figure 4-26 Access Token Manager Configuration	113
163	Figure 4-27 Access Token Attribute Contract.....	114
164	Figure 4-28 Access Token Contract Fulfillment	115
165	Figure 4-29 Data Store for User Lookup	116
166	Figure 4-30 Attribute Directory Search.....	117
167	Figure 4-31 Access Token Contract Fulfillment	118
168	Figure 4-32 Access Token Issuance Criteria.....	119

169	Figure 4-33 OIDC Policy Creation	120
170	Figure 4-34 OIDC Policy Attribute Contract	121
171	Figure 4-35 OIDC Policy Contract Fulfillment	122
172	Figure 4-36 OIDC Client Configuration	124
173	Figure 6-1 Glassfish SSL Settings	134
174	Figure 7-1 Using Postman to Obtain the ID Token	142
175	Figure 7-2 Authorization Prompt	144
176	Figure 7-3 Token Introspection Request and Response	145

177 1 Introduction

178 The following guide demonstrates a standards-based example solution for efficiently and securely
179 gaining access to mission-critical data via mobile devices and applications (apps). This guide
180 demonstrates multifactor authentication (MFA) and mobile single sign-on (MSSO) solutions for native
181 and web apps using standards-based commercially available and open-source products. We cover all of
182 the products that we employed in our solution set. We do not recreate the product manufacturer's
183 documentation. Instead, we provide pointers to where this documentation is available from the
184 manufacturers. This guide shows how we incorporated the products together in our environment as a
185 reference implementation of the proposed build architecture for doing MSSO.

186 *Note: This is not a comprehensive tutorial. There are many possible service and security configurations
187 for these products that are out of scope for this reference solution set.*

188 1.1 Practice Guide Structure

189 This National Institute of Standards and Technology (NIST) Cybersecurity Practice Guide demonstrates a
190 standards-based example solution and provides users with the information they need to replicate this
191 approach to implementing our MSSO build. The example solution is modular and can be deployed in
192 whole or in parts.

193 This guide contains three volumes:

- 194 ■ NIST SP 1800-13A: *Executive Summary*
- 195 ■ NIST SP 1800-13B: *Approach, Architecture, and Security Characteristics* – what we built and why
- 196 ■ NIST SP 1800-13C: *How-To Guides* – instructions for building the example solution (**you are
197 here**)

198 See Section 2 in Volume B of this guide for a more detailed overview of the different volumes and
199 sections, and the audiences that may be interested in each.

200 1.2 Build Overview

201 The National Cybersecurity Center of Excellence (NCCoE) worked with its build team partners to create a
202 lab demonstration environment that includes all of the architectural components and functionality
203 described in Section 4 of Volume B of this build guide. This includes mobile devices with sample apps,
204 hardware and software-based authenticators to demonstrate the Fast Identity Online (FIDO) standards
205 for MFA, the authentication server and authorization server (AS) components required to demonstrate
206 the AppAuth authorization flows (detailed in Internet Engineering Task Force [IETF] Request for
207 Comments [RFC] 8252) with federated authentication to a Security Assertion Markup Language (SAML)
208 Identity Provider (IdP) and an OpenID Connect (OIDC) Provider. The complete build includes several

209 systems deployed in the NCCoE lab by StrongAuth, Yubico and Ping Identity as well as cloud-hosted
210 resources made available by Motorola Solutions and by Nok Nok Labs.

211 This section of the build guide documents the build process and specific configurations that were used in
212 the lab.

213 **1.2.1 Usage Scenarios**

214 The build architecture supports three usage scenarios. The scenarios all demonstrate single sign-on
215 (SSO) among Motorola Solutions Public Safety Experience (PSX) apps using the AppAuth pattern, but
216 differ in the details of the authentication process. The three authentication mechanisms are as follows:

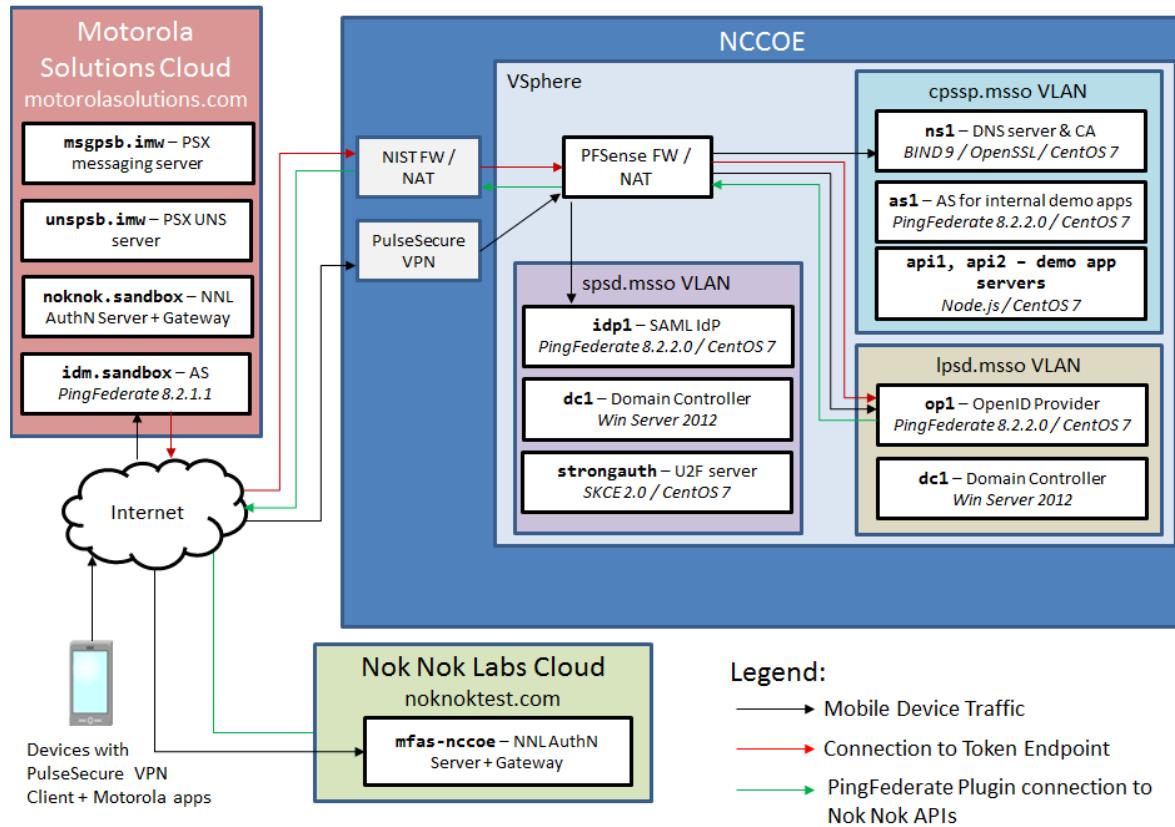
- 217 ▪ The OAuth AS directly authenticates the user with FIDO Universal Authentication Framework
218 (UAF); user accounts are managed directly by the service provider.
- 219 ▪ The OAuth AS redirects the user to a SAML IdP, which authenticates the user with a password
220 and FIDO U2F.
- 221 ▪ The OAuth AS redirects the user to an OIDC IdP, which authenticates the user with FIDO UAF.

222 In all three scenarios, once the authentication flow is completed, the user can launch multiple Motorola
223 Solutions PSX apps without additional authentication, demonstrating SSO. These three scenarios were
224 chosen to reflect different real-world implementation options that public safety and first responder
225 (PSFR) organizations might choose. Larger PSFR organizations may host (or obtain from a service
226 provider) their own IdPs, enabling them to locally manage user accounts, group memberships, and other
227 user attributes, and to provide them to multiple Relying Parties (RPs) through federation. SAML is
228 currently the most commonly used federation protocol, but OIDC might be preferred for new
229 implementations. As demonstrated in this build, RPs can support both protocols more or less
230 interchangeably. For smaller organizations, a service provider might also act in the role of “identity
231 provider of last resort,” maintaining user accounts and attributes on behalf of organizations.

232 **1.2.2 Architectural Overview**

233 Figure 1-1 shows the lab build architecture.

234 **Figure 1-1 Lab Build Architecture**



235

236 Figure 1-1 depicts the four environments that interact in the usage scenarios:

- 237 ▪ Motorola Solutions cloud – a cloud-hosted environment providing the back-end app servers for the Motorola Solutions PSX Mapping and Messaging apps, as well as an OAuth AS that the app servers use to authorize requests from mobile devices
- 238 ▪ Nok Nok Labs cloud – a cloud-hosted server running both the Nok Nok Authentication Server (NNAS) and the Nok Nok Labs Gateway
- 239 ▪ NCCoE – the NCCoE lab, including several servers hosted in a vSphere environment running the IdPs and directory services that would correspond to PSFR organizations' infrastructure to support federated authentication to a service provider, like Motorola Solutions. An additional AS and some demonstration app back-ends are also hosted in the NCCoE lab for internal testing.
- 240 ▪ mobile devices connected to public cellular networks with the required client software to authenticate to, and access, Motorola Solutions back-end apps and the NCCoE Lab systems
- 241 ▪
- 242 ▪
- 243 ▪
- 244 ▪
- 245 ▪
- 246 ▪
- 247 ▪

248 The names of the Virtual Local Area Networks (VLANs) in the NCCoE lab are meant to depict different
249 organizations participating in an MSSO scheme:

- 250 ■ SPSD – State Public Safety Department, a PSFR organization with a SAML IdP
251 ■ LPSD – Local Public Safety Department, a PSFR organization with an OIDC IdP
252 ■ CPSSP – Central Public Safety Service Provider, a Software as a Service (SaaS) provider serving
253 the PSFR organizations, analogous to Motorola Solutions

254 The fictitious *.mssso* top-level domain is simply a reference to the MSSO project. The demonstration apps
255 hosted in the CPSSP VLAN were used to initially test and validate the federation setups in the user
256 organization; this guide mainly focuses on the integration with the Motorola Solutions AS and app back-
257 end.

258 The arrows in Figure 1-1 depict traffic flows between the three different environments, to illustrate the
259 networking requirements for cross-organizational MSSO flows. This diagram does not depict traffic flows
260 within environments (e.g., between the IdPs and the Domain Controllers providing directory services).

261 The depicted traffic flows are described below:

- 262 ■ Mobile device traffic – The PSX client apps on the device connect to the publicly-routable PSX
263 app servers in the Motorola Solutions cloud. The mobile browser also connects to the Motorola
264 Solutions AS, and, in the federated authentication scenarios, the browser is redirected to the
265 IdPs in the NCCoE Lab. The mobile devices use the Pulse Secure Virtual Private Network (VPN)
266 client to access internal lab services through Network Address Translation (NAT) addresses
267 established on the pfSense firewall. This enables the use of the internal lab Domain Name
268 System (DNS) server to resolve the hostnames under the *.mssso* top-level domain, which is not
269 actually registered in public DNS. To support UAF authentication at the lab-hosted OIDC IdP, the
270 Nok Nok Passport app on the devices also connects to the publicly routable NNAS instance
271 hosted in the Nok Nok Labs cloud environment.
- 272 ■ Connection to Token Endpoint – The usage scenario where the Motorola Solutions AS redirects
273 the user to the OIDC IdP in the lab requires the AS to initiate an inbound connection to the IdP's
274 Token Endpoint. To enable this, the PingFederate run-time port, 9031, is exposed via NAT
275 through the NIST firewall. Note that no inbound connection is required in the SAML IdP
276 integration, as the SAML web browser SSO does not require direct back-channel communication
277 between the AS and the IdP. SAML authentication requests and responses are transmitted
278 through browser redirects.
- 279 ■ PingFederate plugin connection to Nok Nok Application Programming Interfaces (APIs) – To
280 support UAF authentication, the OIDC IdP includes a PingFederate adapter developed by Nok
281 Nok Labs that needs to connect to the APIs on the NNAS.

282 In a typical production deployment, the NNAS would not be directly exposed to the internet; instead,
 283 mobile client interactions with the Authentication Server APIs would traverse a reverse proxy server.
 284 Nok Nok Labs provided a cloud instance of their software as a matter of expedience in completing the
 285 lab build.

286 Additionally, the use of a VPN client on mobile devices is optional. Many organizations directly expose
 287 their IdPs to the public internet, though some organizations prefer to keep those services internal and
 288 use a VPN to access them. Organizations can decide this based on their risk tolerance, but this build
 289 architecture can function with or without a VPN client on the mobile devices.

290 1.2.3 General Infrastructure Requirements

291 Some general infrastructure elements must be in place to support the components of this build guide.
 292 These are assumed to exist in the environment prior to the installation of the architecture components
 293 in this guide. The details of how these services are implemented are not directly relevant to the build.

- 294 ■ DNS – All server names are expected to be resolvable in DNS. This is especially important for
 295 FIDO functionality, as the application identification (App ID) associated with cryptographic keys
 296 is derived from the hostname used in app Uniform Resource Locators (URLs).
- 297 ■ Network Time Protocol (NTP) – Time synchronization among servers is important. A clock
 298 difference of five minutes or more is sufficient to cause JavaScript Object Notation (JSON) Web
 299 Token (JWT) validation, for example, to fail. All servers should be configured to synchronize time
 300 with a reliable NTP source.
- 301 ■ Certificate Authority (CA) – Hypertext Transfer Protocol Secure (HTTPS) connections should be
 302 used throughout the architecture. Transport Layer Security (TLS) certificates are required for all
 303 servers in the build. If an in-house CA is used to issue certificates, the root and any intermediate
 304 certificates must be provisioned to the trust stores in client mobile devices and servers.

305 1.3 Typographic Conventions

306 The following table presents typographic conventions used in this volume.

Typeface/ Symbol	Meaning	Example
<i>Italics</i>	filenames and pathnames references to documents that are not hyperlinks, new terms, and placeholders	For detailed definitions of terms, see the <i>NCCoE Glossary</i> .
Bold	names of menus, options, command buttons and fields	Choose File > Edit .

Typeface/ Symbol	Meaning	Example
Monospace	command-line input, on-screen computer output, sample code examples, status codes	<code>mkdir</code>
Monospace Bold	command-line user input contrasted with computer output	<code>service sshd start</code>
blue text	link to other parts of the document, a web URL, or an email address	All publications from NIST's National Cybersecurity Center of Excellence are available at https://nccoe.nist.gov

2 How to Install and Configure the Mobile Device

This section covers all of the different aspects of installing and configuring the mobile device. There are several prerequisites and different components that need to work in tandem for the entire SSO architecture to work.

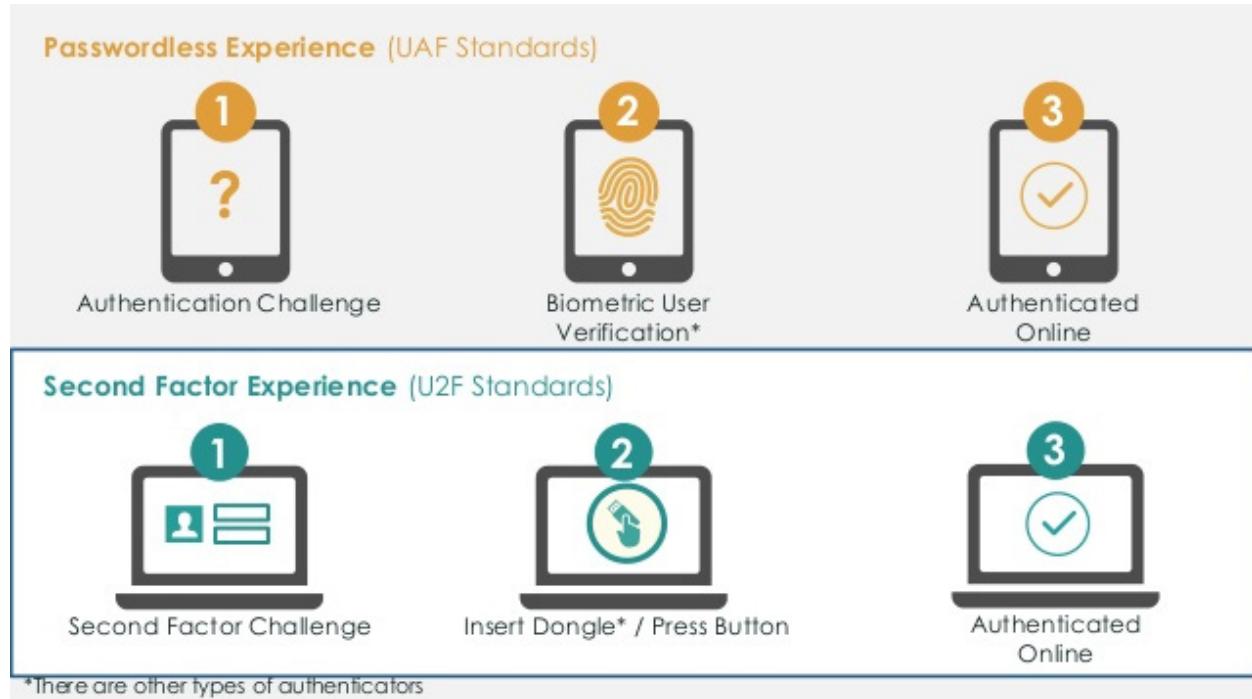
2.1 Platform and System Requirements

This section covers requirements for mobile devices—both hardware and software—for the SSO and FIDO authentication components of the architecture to work properly. The two dominant mobile platforms are Google’s Android and Apple’s iPhone operating system (iOS). The NCCoE reference architecture only tested Android devices and apps, but the same core architecture could support iOS.

First, for SSO support, the NCCoE reference architecture follows the guidance of the *OAuth 2.0 for Native Apps Best Current Practice (BCP)* [1]. That guidance, also known as *AppAuth*, requires that developers use an *external user-agent* (e.g., Google’s Chrome for Android web browser) instead of an *embedded user-agent* (e.g., an Android WebView) for their OAuth authorization requests. Because of this, the mobile platform must support the use of external user-agents.

Second, for FIDO support, this architecture optionally includes two different types of authenticators: UAF and U2F. The *FIDO Specifications Overview* presentation [2] explains the difference, as shown in Figure 2-1.

324 Figure 2-1 Comparison of UAF and U2F Standards



325

326 The following subsections address Android-specific requirements to support SSO and FIDO
 327 authentication.

328 2.1.1 Supporting SSO

329 While it is not strictly required, the BCP recommends that the device provide an external user-agent that
 330 supports “in-app browser tabs,” which Google describes as the *Android Custom Tab* feature. The
 331 following excerpt is from the AppAuth Android-specific guidance in Appendix B.2 of RFC 8252:

332 *Apps can initiate an authorization request in the browser without the user leaving the app,*
 333 *through the Android Custom Tab feature which implements the in-app browser tab pattern. The*
 334 *user's default browser can be used to handle requests when no browser supports Custom Tabs.*

335 *Android browser vendors should support the Custom Tabs protocol (by providing an*
 336 *implementation of the “CustomTabsService” class), to provide the in-app browser tab user*
 337 *experience optimization to their users. Chrome is one such browser that implements Custom*
 338 *Tabs.*

339 Any device manufacturer can support Custom Tabs in their Android browser. However, Google
 340 implemented this in its Chrome for Android web browser in September 2015 [3]. Because Chrome is not
 341 part of the operating system (OS) itself, but is downloaded from the Google Play Store, recent versions

342 of Chrome can be used on older versions of Android. In fact, the Chrome Developer website's page on
343 Chrome Custom Tabs [\[4\]](#) states that it can be used on Android Jelly Bean (4.1), which was released in
344 2012, and up.

345 To demonstrate SSO, the NCCoE reference architecture utilizes the Motorola Solutions PSX App Suite,
346 which requires Android Lollipop (5.0) or newer.

347 **2.1.2 Supporting FIDO U2F**

348 The device will need the following components for FIDO U2F:

- 349 ■ a web browser capable of understanding a U2F challenge request from an IdP
350 ■ a FIDO U2F client app capable of handling the challenge
351 ■ Near Field Communication (NFC) hardware support

352 Chrome for Android [\[5\]](#) is a browser that understands U2F challenge requests, and Google Authenticator
353 [\[6\]](#) (works on Android Gingerbread [2.3.3] and up) is an app capable of handling the challenge. If NFC is
354 unavailable, Bluetooth and Universal Serial Bus Type-C (USB-C) are also options for connecting U2F
355 tokens. Google has added support for both options into their Play Services framework, as of November
356 2017. However, these other methods are less widely used and are not a focus of this guide.

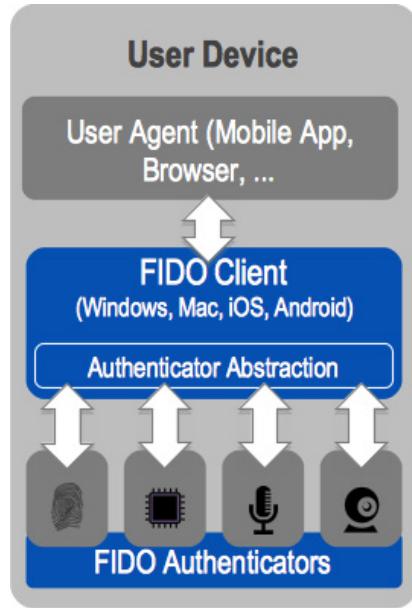
357 **2.1.3 Supporting FIDO UAF**

358 The device will need the following components for FIDO UAF:

- 359 ■ a web browser
360 ■ a FIDO UAF client app capable of handling the challenge
361 ■ a FIDO UAF authenticator

362 These components are pictured in Figure 2-2, which is from the *FIDO UAF Architectural Overview* [\[7\]](#).

363 Figure 2-2 FIDO UAF Architectural Overview



364

365 While the overview refers to the last two components (client and authenticator) as separate
 366 components, these components can—and often do—come packaged in a single app. The NCCoE
 367 reference architecture utilizes the Nok Nok Passport [8] app to provide these two components. In
 368 addition to the apps, the device will need to provide some hardware component to support the FIDO
 369 UAF authenticator. For example, for biometric-based FIDO UAF authenticators, a camera would be
 370 needed to support face or iris scanning, a microphone would be needed to support voiceprints, and a
 371 fingerprint sensor would be needed to support fingerprint biometrics. Of course, if a Personal
 372 Identification Number (PIN) authenticator is used, a specific hardware sensor is not required. Beyond
 373 the actual input method of the FIDO UAF factor, additional (optional) hardware considerations for a UAF
 374 authenticator include secure key storage for registered FIDO key pairs, storage of biometric templates,
 375 and execution of matching functions (e.g., within dedicated hardware or on processor trusted execution
 376 environments [TEE]).

377 2.2 How to Install and Configure the Mobile Apps

378 This section covers the installation and configuration of the mobile apps needed for various components
 379 of the reference architecture: SSO, FIDO U2F, and FIDO UAF.

380 2.2.1 How to Install and Configure SSO-Enabled Apps

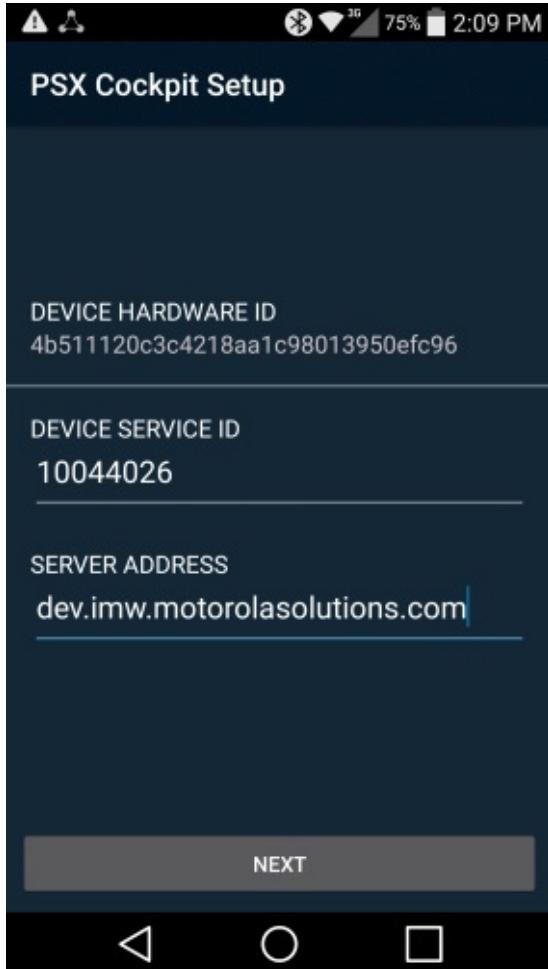
381 For SSO-enabled apps, there is no universal set of installation and configuration procedures; these will
 382 vary depending on the design choices of the app manufacturer. The NCCoE reference architecture uses
 383 the *Motorola Solutions PSX App Suite* [9] Version 5.4. This set of mobile apps provides several

384 capabilities for the public safety community. Our setup consisted of three apps: *PSX Messenger* for text,
385 photo, and video communication; *PSX Mapping* for shared location awareness; and *PSX Cockpit* to
386 centralize authentication and identity information across the other apps. These apps cannot be obtained
387 from a public venue (e.g., the Google Play Store); rather, the binaries must be obtained from Motorola
388 Solutions and installed via other means, such as a Mobile Device Management (MDM) solution or
389 private app store.

390 *2.2.1.1 Configuring the PSX Cockpit App*

391 1. Open the Cockpit app. Your screen should look like Figure 2-3.

392 Figure 2-3 PSX Cockpit Setup

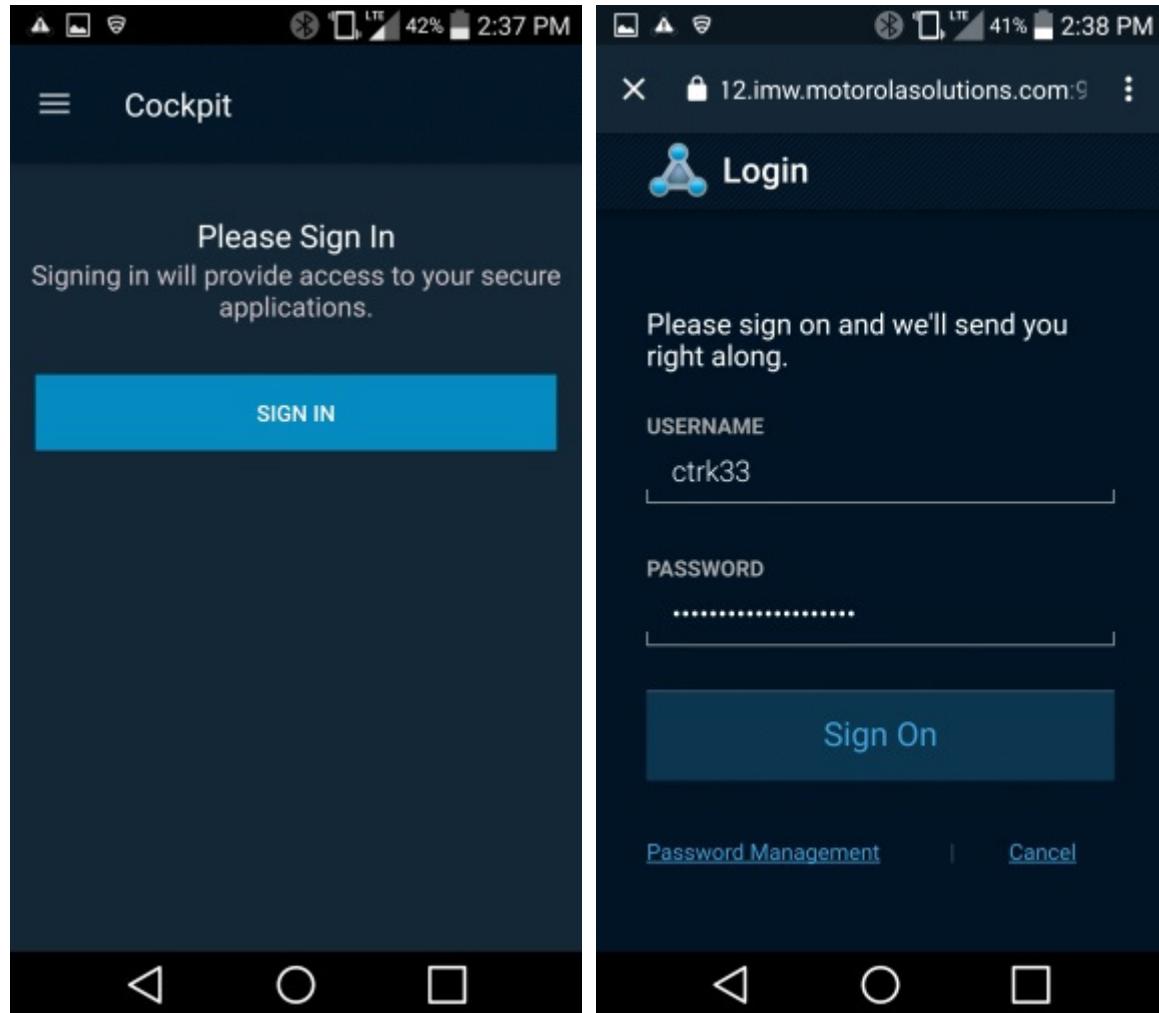


393
394 2. For **DEVICE SERVICE ID**, select a Device Service ID in the range given to you by your
395 administrator. Note that these details would be provided by Motorola Solutions if you are using

396 their service offering, or by your administrator if you are hosting the PSX app servers in your
397 own environment. Each device should be configured with a unique Device Service ID
398 corresponding to the username from the username range. For example, the NCCoE lab used a
399 Device Service ID of “22400” to correspond to a username of “2400.”

- 400 3. For **SERVER ADDRESS**, use the Server Address given to you by your administrator. For example,
401 the NCCoE lab used a Server Address of “uns5455.imw.motorolasolutions.com.”
402 4. If a **Use SUPL APN** checkbox appears, leave it unchecked.
403 5. Tap **NEXT**. Your screen should look like Figure 2-4.

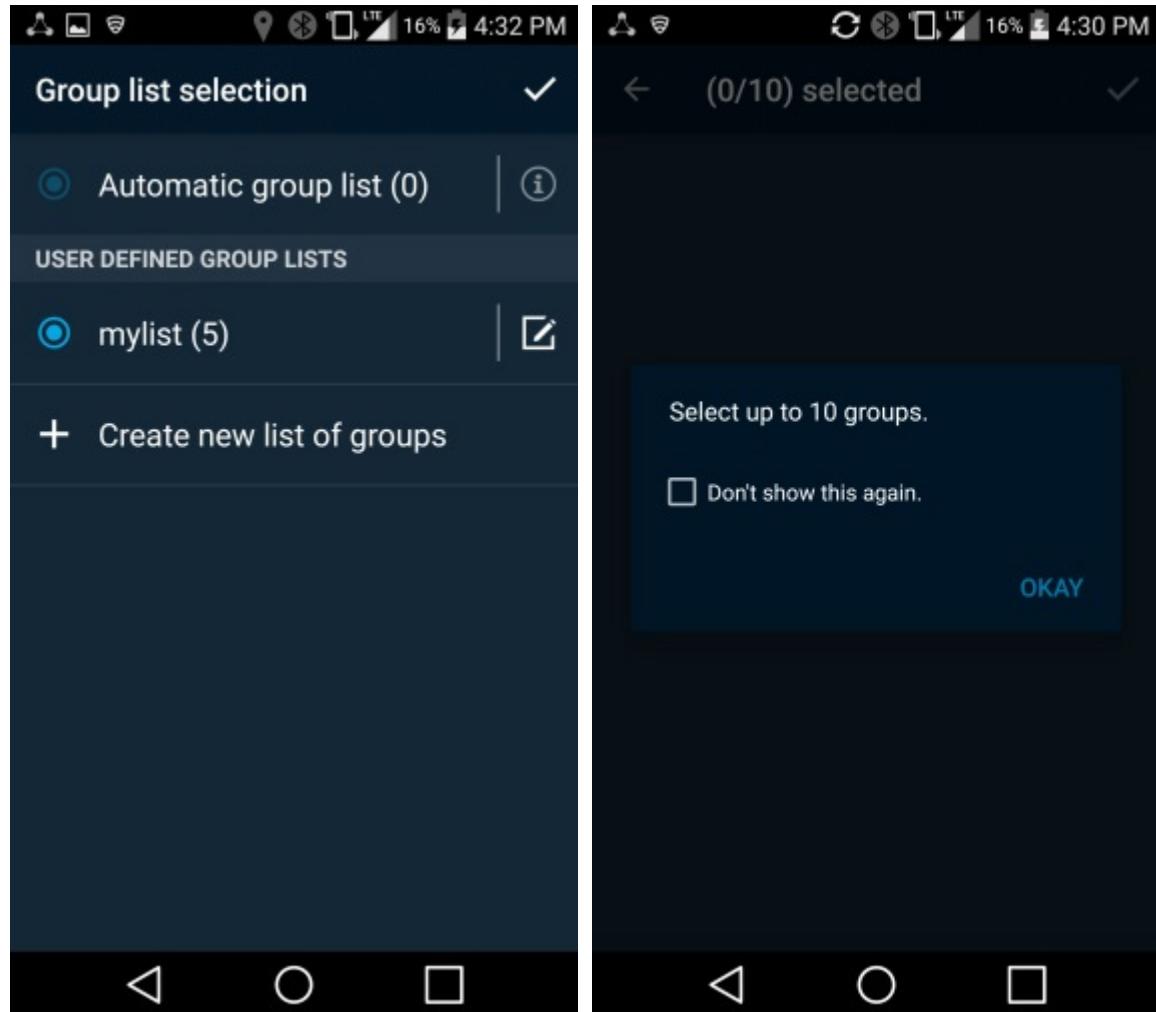
404 Figure 2-4 PSX Cockpit Setup, Continued



- 405
406 6. Tap **SIGN IN**.

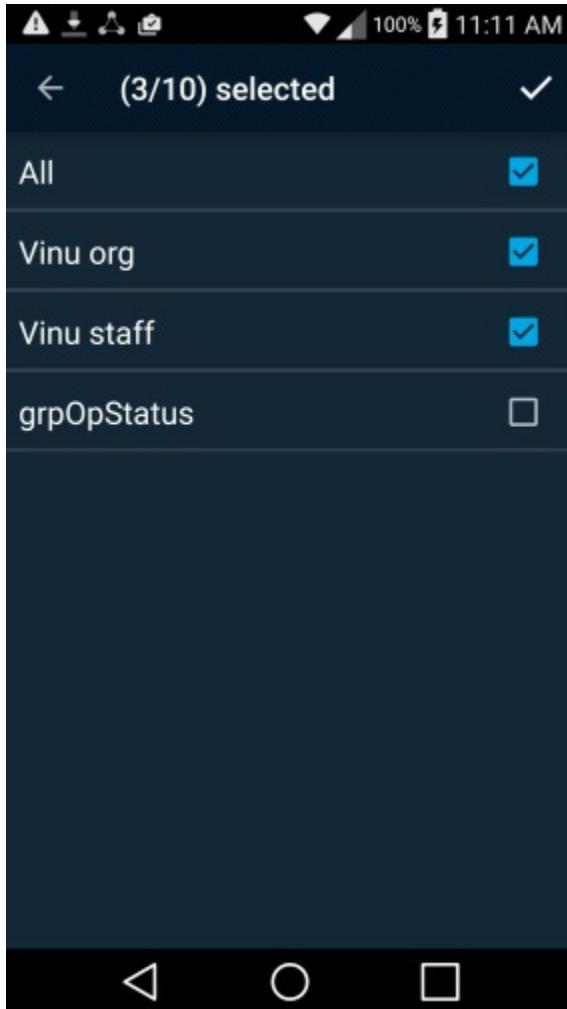
- 407 7. Log in with the authentication procedure determined by the AS and IdP policies. Note that if
408 UAF is used, a FIDO UAF authenticator must be enrolled before this step can be completed. See
409 [Section 2.2.3](#) for details on FIDO UAF enrollment. After you log in, your screen should look like
410 Figure 2-5.

411 **Figure 2-5 PSX Cockpit Group List Selection**



- 412
413 8. Tap **Create new list of groups**. This is used to select which organizationally-defined groups of
414 users you can receive data updates for in the other PSX apps.
415 9. Tap **OKAY**. Your screen should look like Figure 2-6.

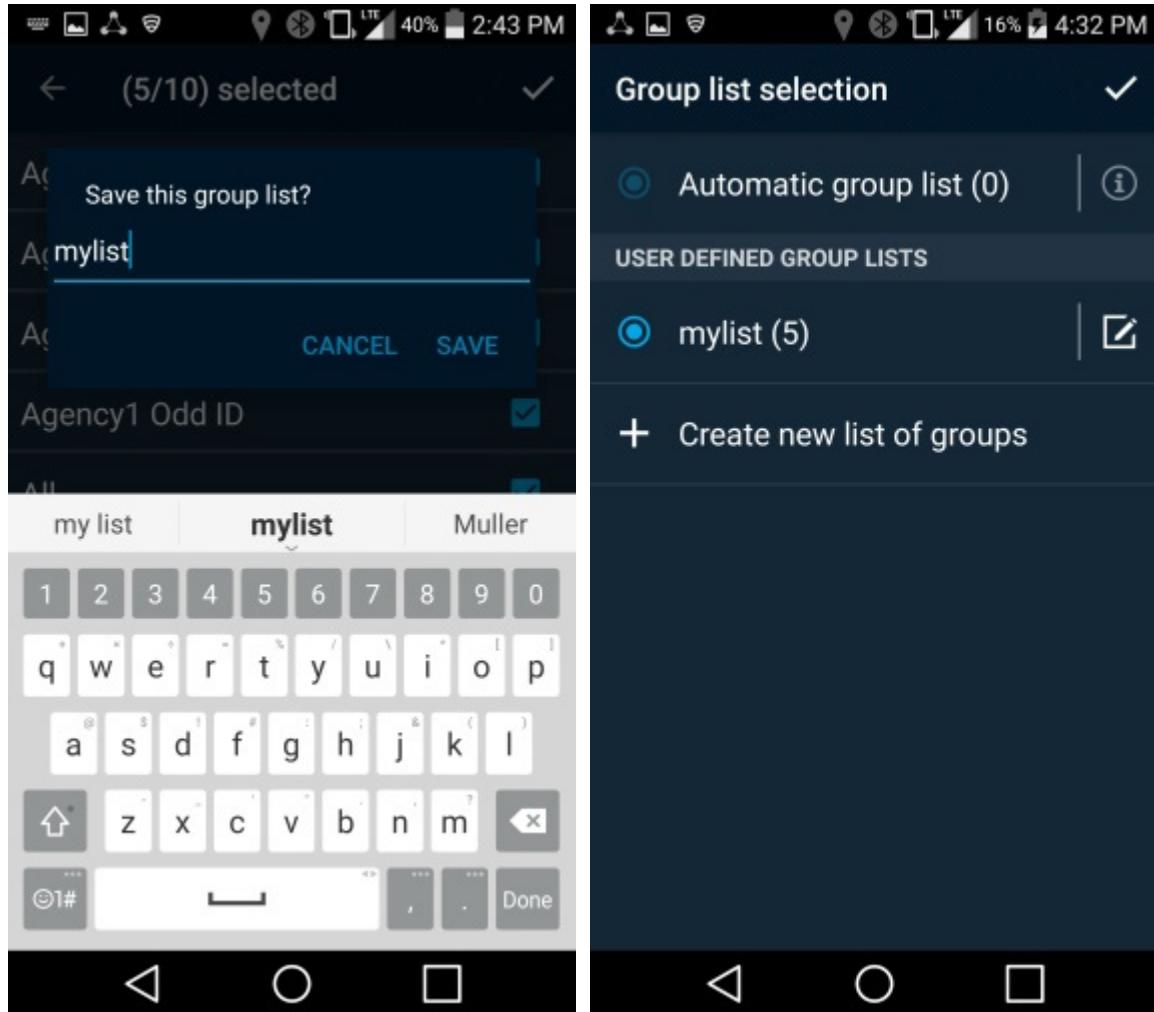
416 Figure 2-6 PSX Cockpit Groups



417

- 418 10. Check the checkboxes for the groups that you wish to use. Note that it may take a short time for
419 the groups to appear.
- 420 11. Tap on the upper-right checkmark. Your screen should look like Figure 2-7.

421 Figure 2-7 PSX Cockpit Group List Setup Complete

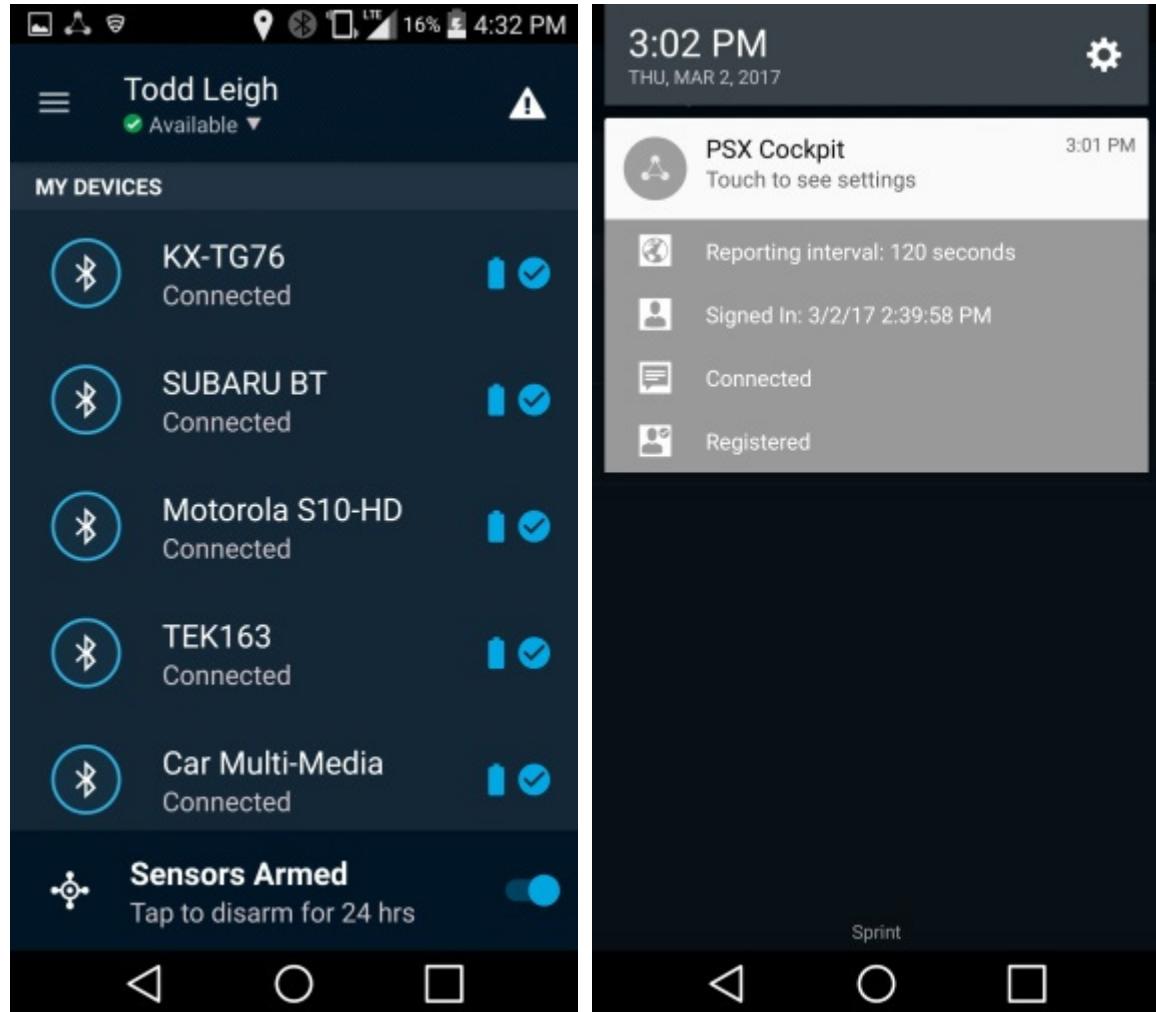


422

423 12. Enter a group list name (e.g., "mylist"), and tap **SAVE**.

424 13. Tap the upper-right checkmark to select the list. Your screen should look like Figure 2-8.

425 Figure 2-8 PSX Cockpit User Interface



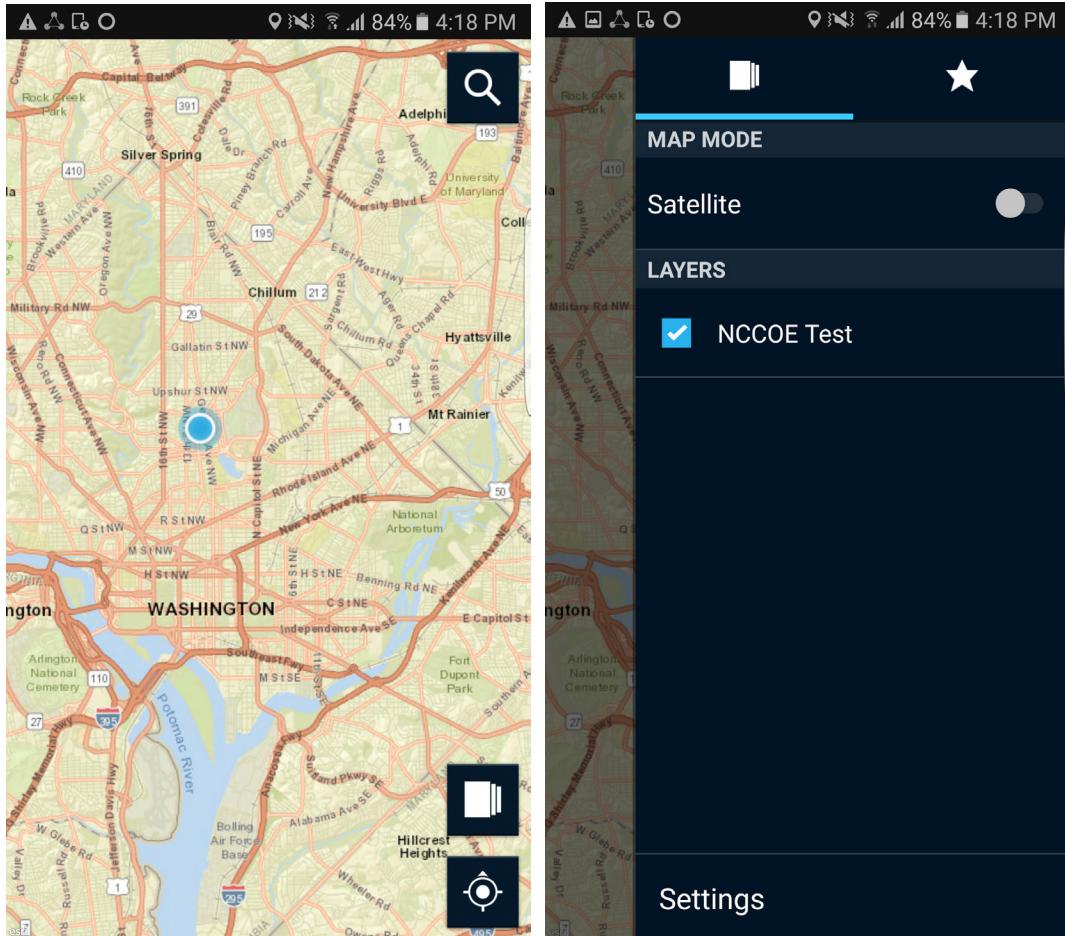
426

- 427 14. On the Cockpit screen, you can trigger an emergency (triangle icon in the upper right); set your
428 status (drop-down menu under your name); or reselect roles and groups, see configuration, and
429 sign off (hamburger menu to the left of your name, and then tap **username**).
430 15. If you pull down your notifications, you should see icons and text indicating “Reporting interval:
431 120 seconds,” “Signed In: <date> <time>,” “Connected,” and “Registered.”

432 *2.2.1.2 Configuring the PSX Mapping App*

433 1. Open the Mapping app. You should see the screen shown in Figure 2-9.

434 **Figure 2-9 PSX Mapping User Interface**

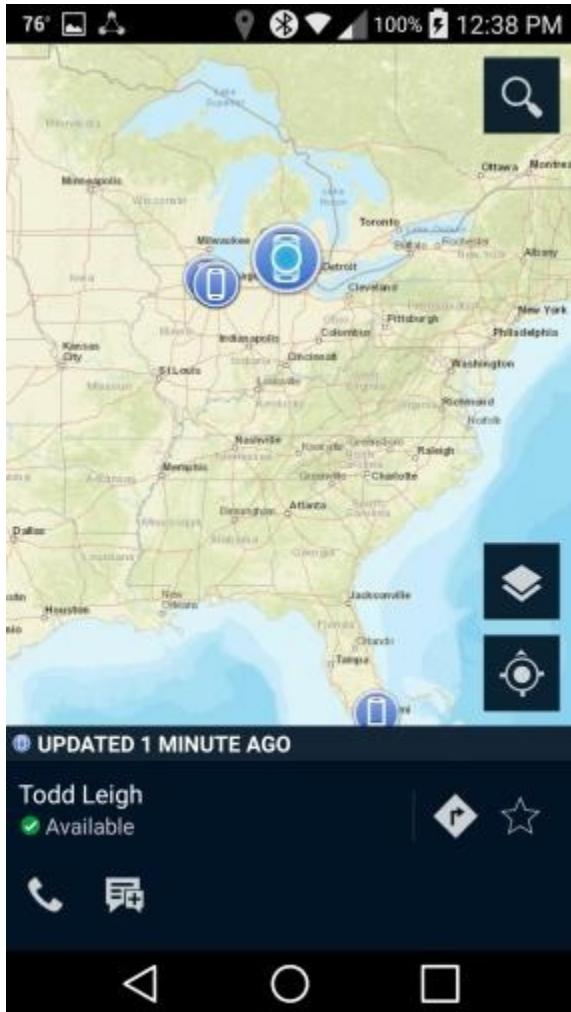


435

436 2. Select the “Layers” icon in the lower-right corner. Group names should appear under **Layers**.

437 3. Select a group. Your screen should look like Figure 2-10.

438 Figure 2-10 PSX Mapping Group Member Information



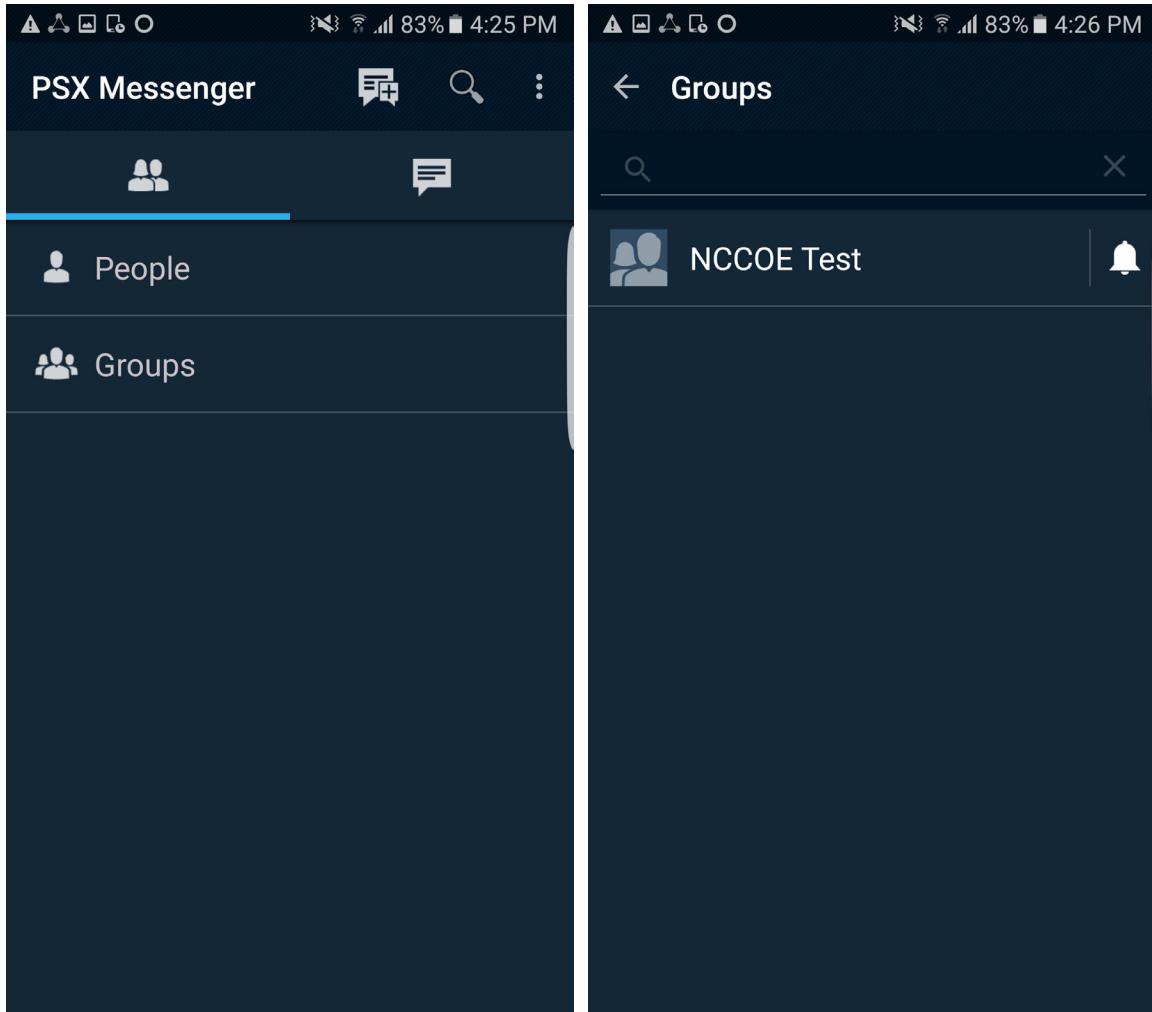
439

- 440 4. The locations of the devices that are members of that group should appear as dots on the map.
- 441 5. Select a device. A pop-up will show the user of the device, and icons for phoning and messaging
- 442 that user.
- 443 6. Selecting the “Messenger” icon for the selected user will take you to the Messenger app, where
- 444 you can send a message to the user.

445 *2.2.1.3 Configuring the PSX Messenger App*

446 1. Open the Messenger app. Your screen should look like Figure 2-11.

447 **Figure 2-11 PSX Messenger User Interface**



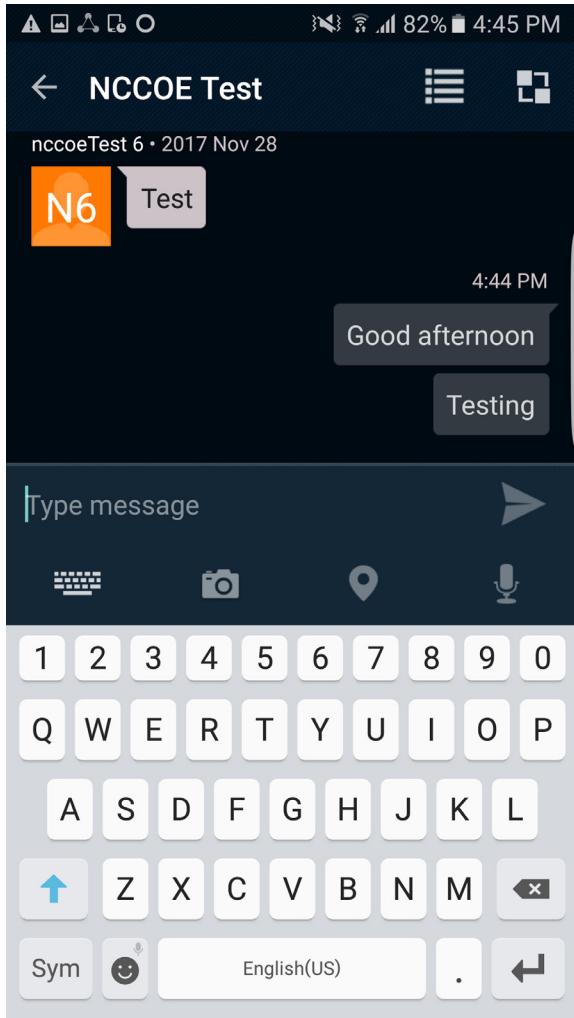
448

449 2. Your screen should show **People** and **Groups**. Select one of them.

450 3. A list of people or groups that you can send a message to should appear. Select one of them.

451 Your screen should look like Figure 2-12.

452 Figure 2-12 PSX Messenger Messages



453

- 454 4. You are now viewing the messaging window. You can type text for a message, and attach a picture, video, voice recording, or map.
- 455
- 456 5. Tap the “Send” icon. The message should appear on your screen.
- 457 6. Tap the “Pivot” icon in the upper-right corner of the message window. Select “Locate,” and you
- 458 will be taken to the Mapping app with the location of the people or group you selected.

459 **2.2.2 How to Install and Configure a FIDO U2F Authenticator**

460 This section covers the installation and usage of a FIDO U2F authenticator on the mobile device. The
461 NCCoE reference architecture utilizes the Google Authenticator app on the mobile device, and a Yubico
462 YubiKey NEO as a hardware token. The app functions as the client-side U2F authenticator and is
463 available on Google's Play Store [\[6\]](#).

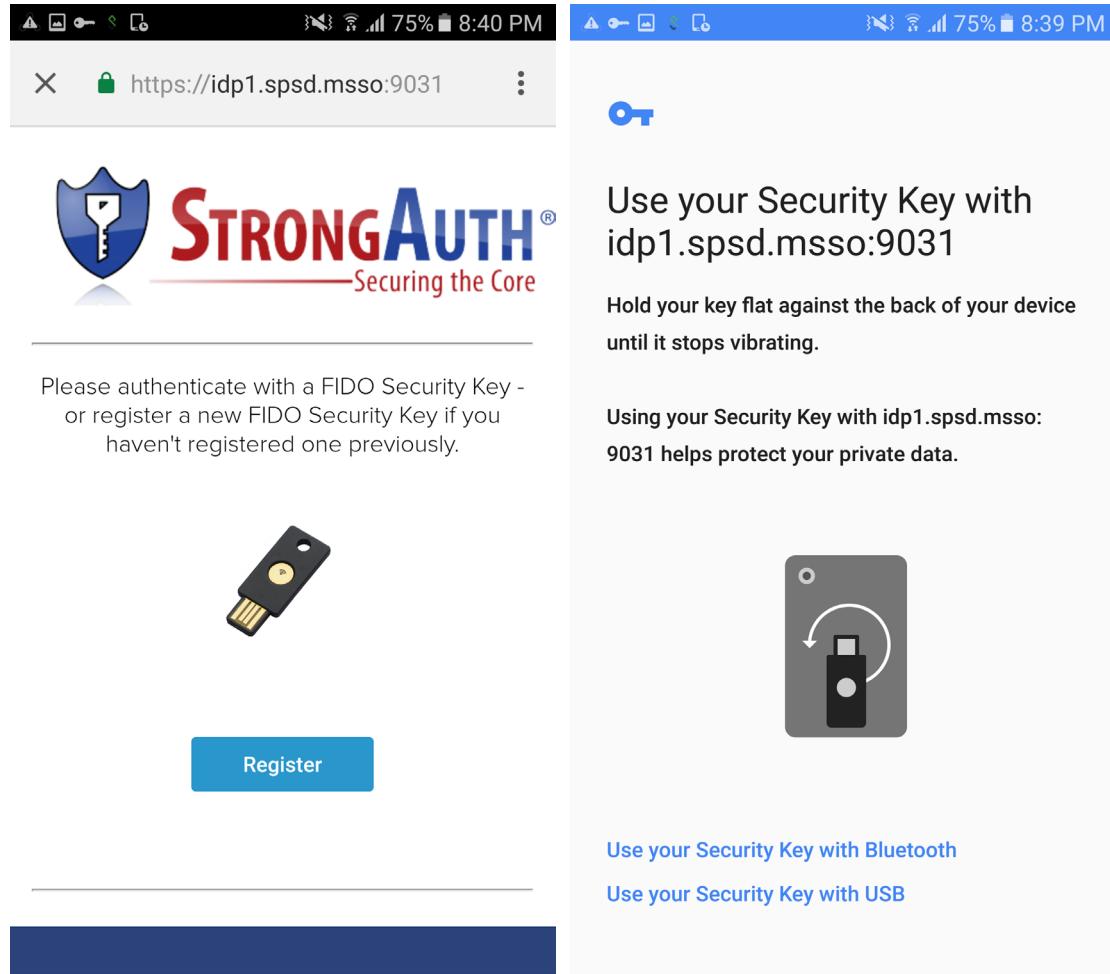
464 ***2.2.2.1 Installing Google Authenticator***

- 465 1. On your Android device, open the Play Store app.
- 466 2. Search for “Google Authenticator,” and install the app. There is no configuration needed until
467 you are ready to register a FIDO U2F token with a StrongAuth server.

468 ***2.2.2.2 Registering the Token***

469 In the architecture that is laid out in this practice guide, there is no out-of-band process to register the
470 user’s U2F token. This takes place the first time the user tries to log in with whatever SSO-enabled app
471 they are using. For instance, when using the PSX Cockpit app, once the user tries to sign into an IdP that
472 has U2F enabled and has successfully authenticated with a username and password, they will be
473 presented with the screen shown in Figure 2-13.

474 Figure 2-13 FIDO U2F Registration



475

476 Because the user has never registered a U2F token, that is the only option the user sees.

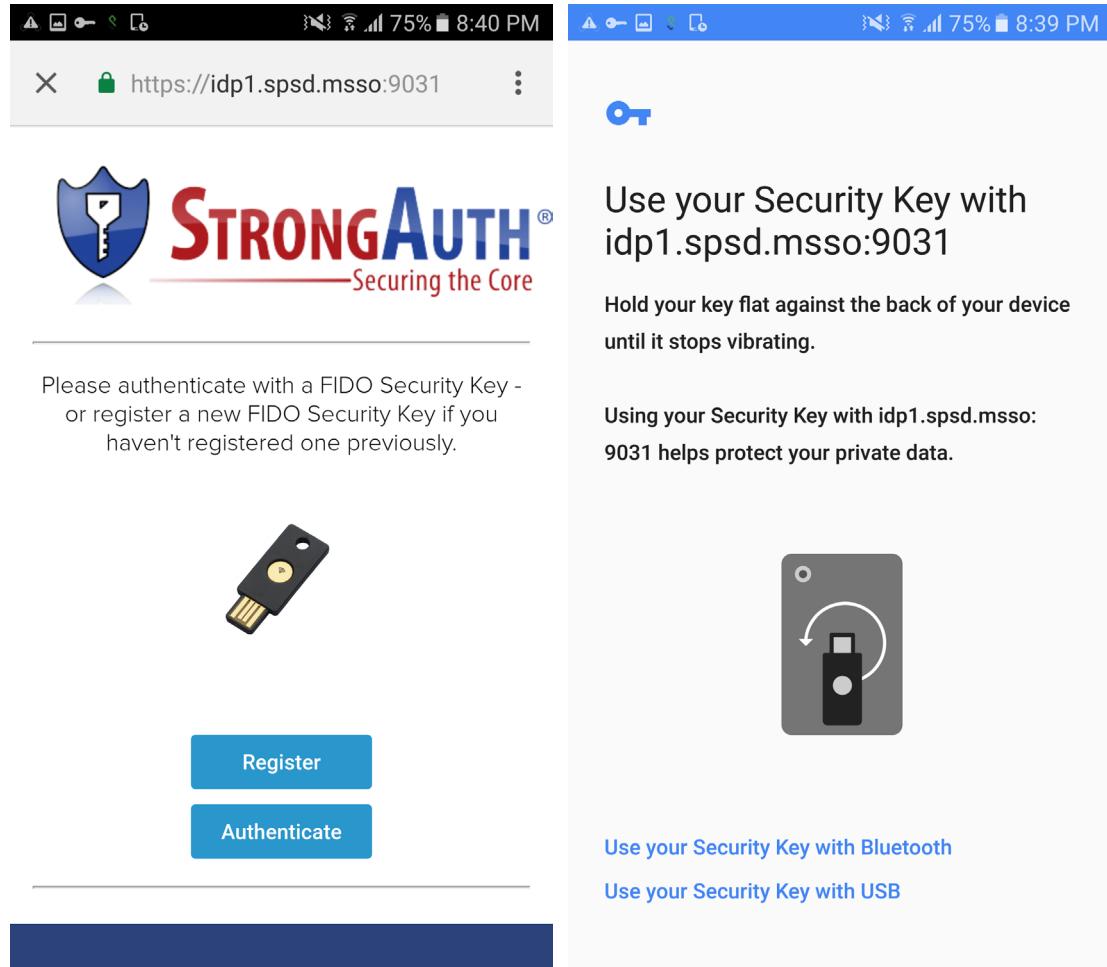
- 477 1. Click **Register**, and the web page will activate the Google Authenticator app, which asks you to use a U2F token to continue (Figure 2-13 above).
- 478 2. Hold the U2F token to your device, and then the token will be registered to your account and you will be redirected to the U2F login screen again.

481 *2.2.2.3 Authenticating with the Token*

- 482 Now, because the system has a U2F token on file for the user, the user has the option to authenticate.
- 483 1. Click **Authenticate** (Figure 2-14), and the Google Authenticator app will be activated once more.

484 2. Hold the U2F token to your device, and then the authentication will be successful and the SSO
 485 flow will continue.

486 **Figure 2-14 FIDO U2F Authentication**



487

488 2.2.3 How to Install and Configure a FIDO UAF Client

489 This section covers the installation and usage of a FIDO UAF client on the mobile device. Any FIDO UAF
 490 client can be used, but the NCCoE reference architecture utilizes the Nok Nok Passport app (hereafter
 491 referred to as "Passport"). The Passport app functions as the client-side UAF app and is available on
 492 Google's Play Store [8]. The following excerpt is from the Play Store page:

493 *Passport from Nok Nok Labs is an authentication app that supports the Universal Authentication
 494 Framework (UAF) protocol from the FIDO Alliance (www.fidoalliance.org).*

495 *Passport allows you to use out-of-band authentication to authenticate to selected websites on a
496 laptop or desktop computer. You can use the fingerprint sensor on FIDO UAF-enabled devices
497 (such as the Samsung Galaxy S® 6, Fujitsu Arrows NX, or Sharp Aquos Zeta) or enter a simple PIN
498 on non-FIDO enabled devices. You can enroll your Android device by using Passport to scan a QR
499 code displayed by the website, then touch the fingerprint sensor or enter a PIN. Once enrolled,
500 you can authenticate using a similar method. Alternatively, the website can send a push
501 notification to your Android device and trigger the authentication.*

502 *This solution lets you use your Android device to better protect your online account, without
503 requiring passwords or additional hardware tokens.*

504 In our reference architecture, we use a Quick Response (QR) code to enroll the device onto Nok Nok
505 Labs' test server.

506 2.2.3.1 *Installing Passport*

- 507 1. On your Android device, open the Play Store app.
- 508 2. Search for “Nok Nok Passport”, and install the app. There is no configuration needed until you
509 are ready to enroll the device with a Nok Nok Labs server.

510 Normally, the user will never need to open the Passport app during authentication; it will automatically
511 be invoked by the SSO-enabled app (e.g., PSX Cockpit). Instead of entering a username and password
512 into a Chrome Custom Tab, the user will be presented with the Passport screen to use the user’s UAF
513 credential.

514 2.2.3.2 *Enrolling the Device*

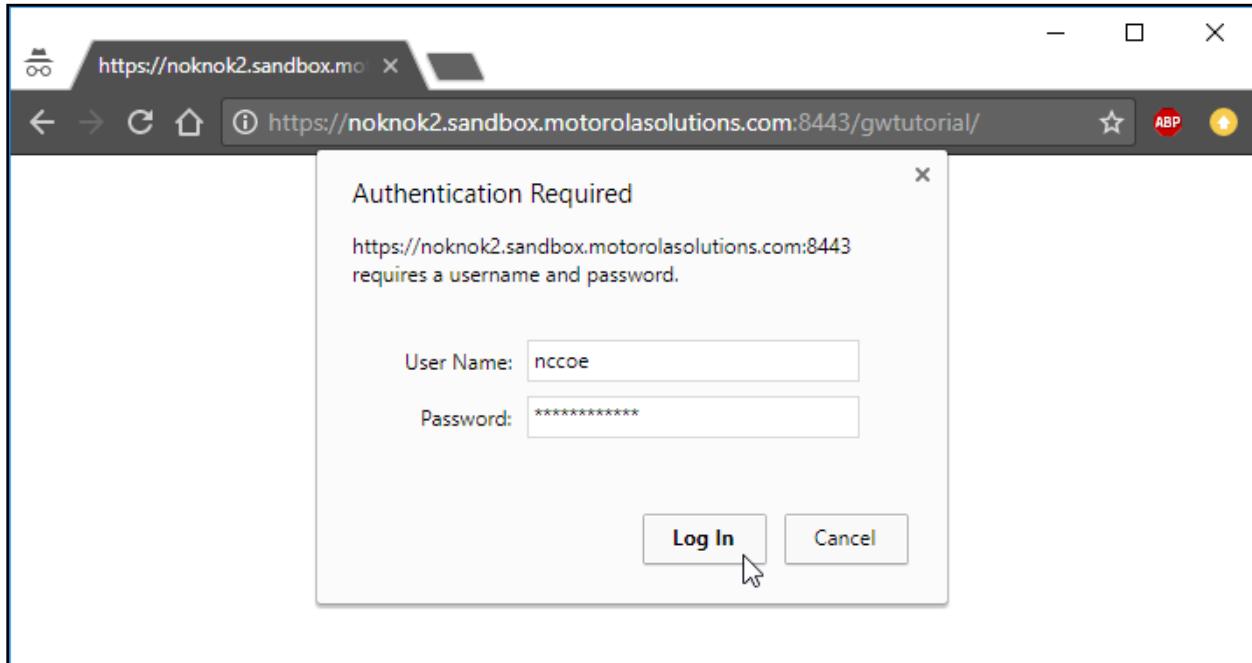
515 This section details the steps to enroll a device to an NNAS. First, you need a device that has Passport
516 installed. Second, you need to use another computer (preferably a desktop or laptop) to interact with
517 your NNAS web interface.

518 *Note: Users are not authenticated during registration. We are using the “tutorial” app provided with the
519 NNAS. This sample implementation does not meet the FIDO requirement of authentication prior to
520 registration. The production version of the NNAS may require additional steps and may have a different
521 interface.*

522 Screenshots that demonstrate the enrollment process are shown in Figure 2-15 through Figure 2-21.

- 523 1. First, use your computer to navigate to the NNAS web interface. You will be prompted for a
524 username and password; enter your administrator credentials, and click **Log In** (Figure 2-15).

525 Figure 2-15 Nok Nok Labs Tutorial App Authentication

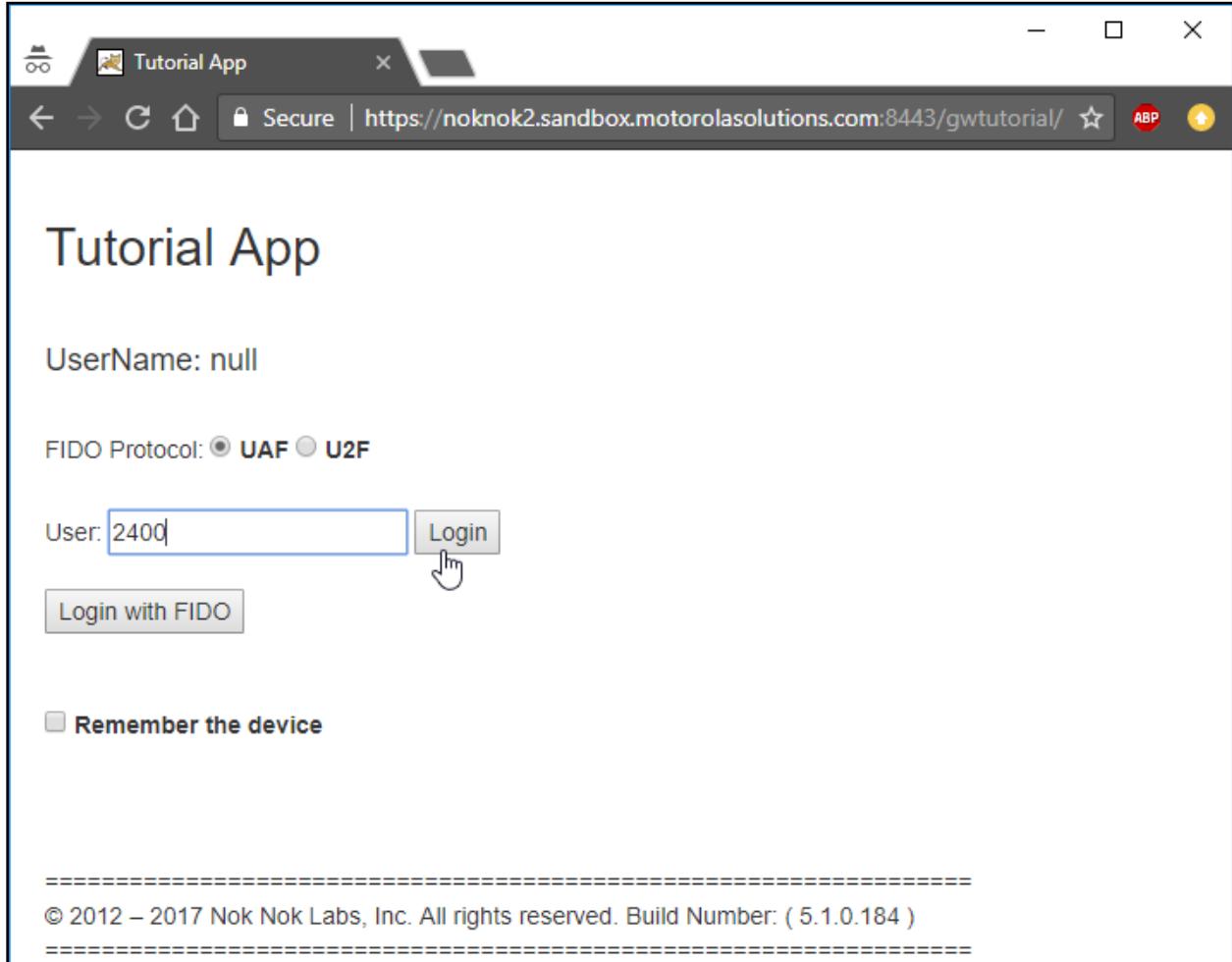


526

527 2. Once you have logged into the NNAS as an administrator, you need to identify which user you
528 want to manage. Enter the username, and click **Login with FIDO** (Figure 2-16).

529 *Note: As stated above, this is the tutorial app, so it only prompts for a username, not a
530 password. A production environment would require user authentication.*

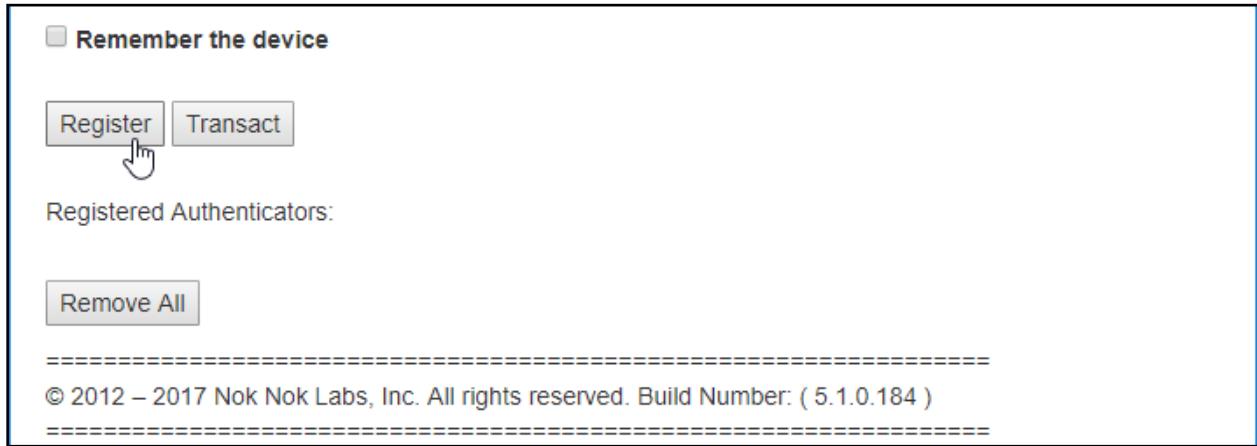
531 Figure 2-16 Nok Nok Labs Tutorial App Login



532

- 533 3. Once you have selected the user, you will need to start the FIDO UAF registration process. To
534 begin, click **Register** (Figure 2-17).

535 Figure 2-17 FIDO UAF Registration Interface

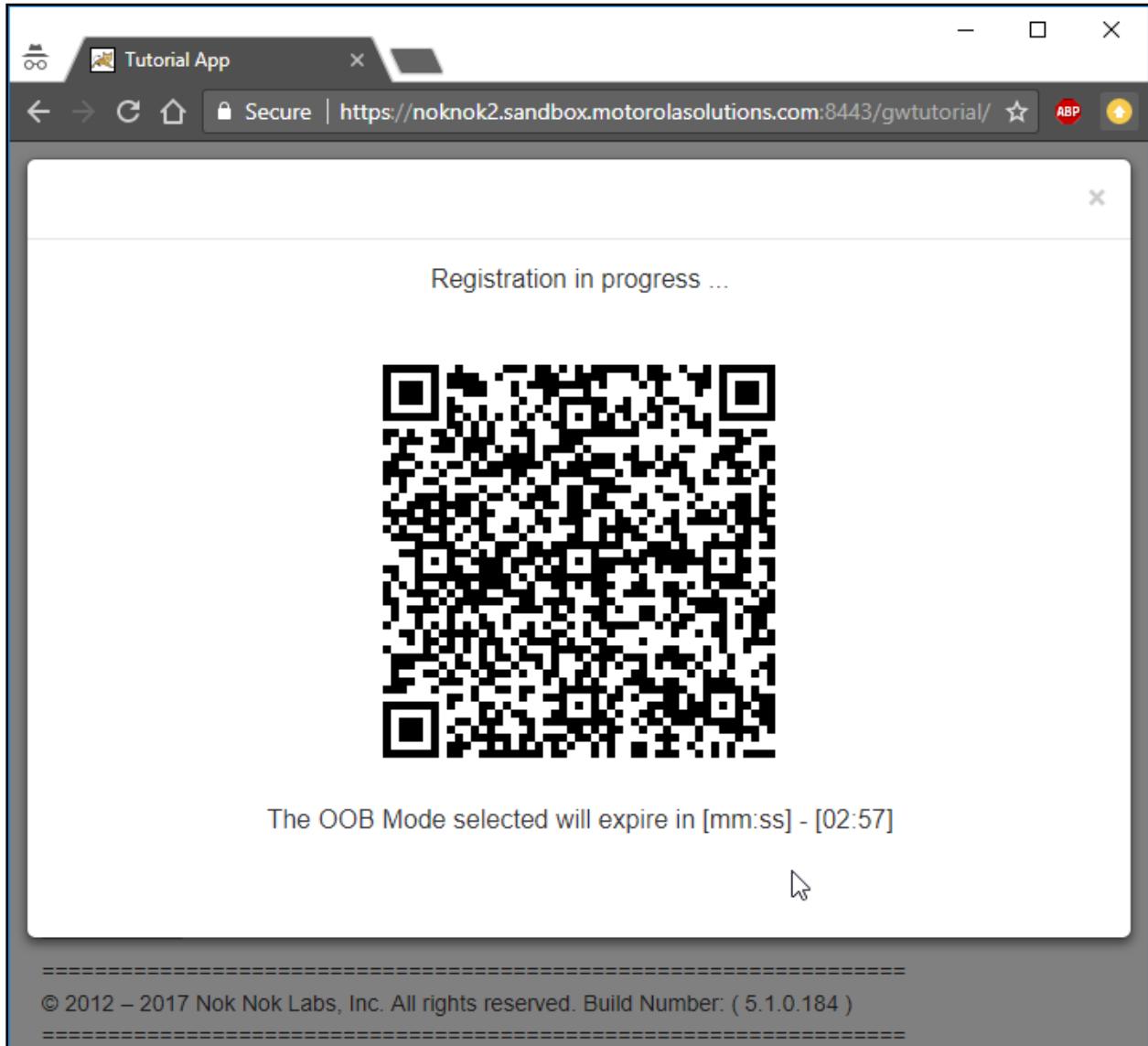


536

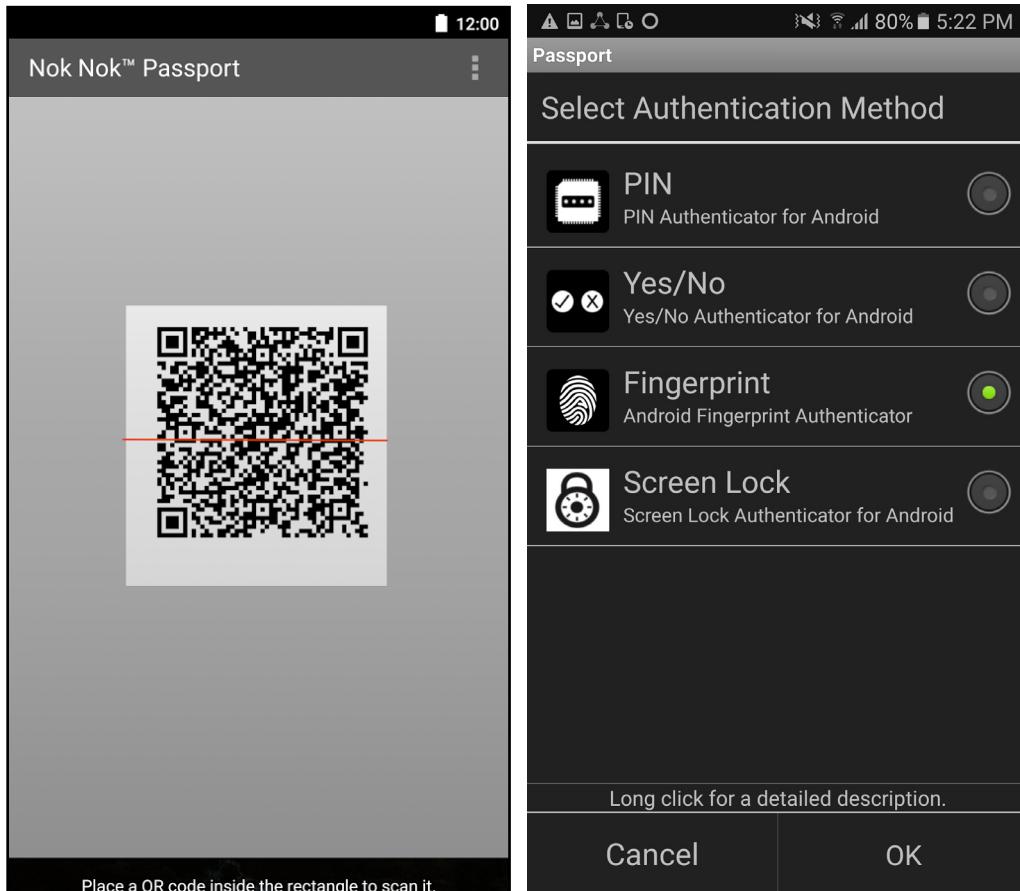
537 4. You will see a window with a QR code and a countdown (Figure 2-18). You have three minutes
538 to finish the registration process with your device.

- 539 a. Once the QR image appears, launch the Passport app on the phone. The Passport app
540 activates the device camera to enable capturing the QR code by centering the code in
541 the square frame in the middle of the screen (Figure 2-19).
- 542 b. Once the QR code is scanned, the app prompts the user to select the type of verification
543 (fingerprint, PIN, etc.) to use (Figure 2-19). The selections may vary based on the au-
544 thenticator modules installed on the device.

545 Figure 2-18 FIDO UAF Registration QR Code



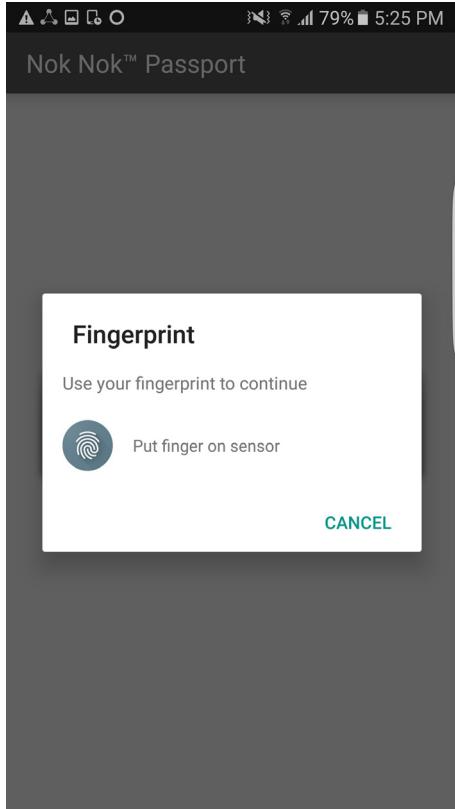
546

547 **Figure 2-19 FIDO UAF Registration Device Flow**

548

- 549 5. In this example, a fingerprint authenticator is registered. The user is prompted for a fingerprint
550 scan to complete registration (Figure 2-20). The fingerprint authenticator uses a fingerprint
551 previously registered in the Android screen-lock settings. If a PIN authenticator were registered,
552 the user would be prompted to set a PIN instead.

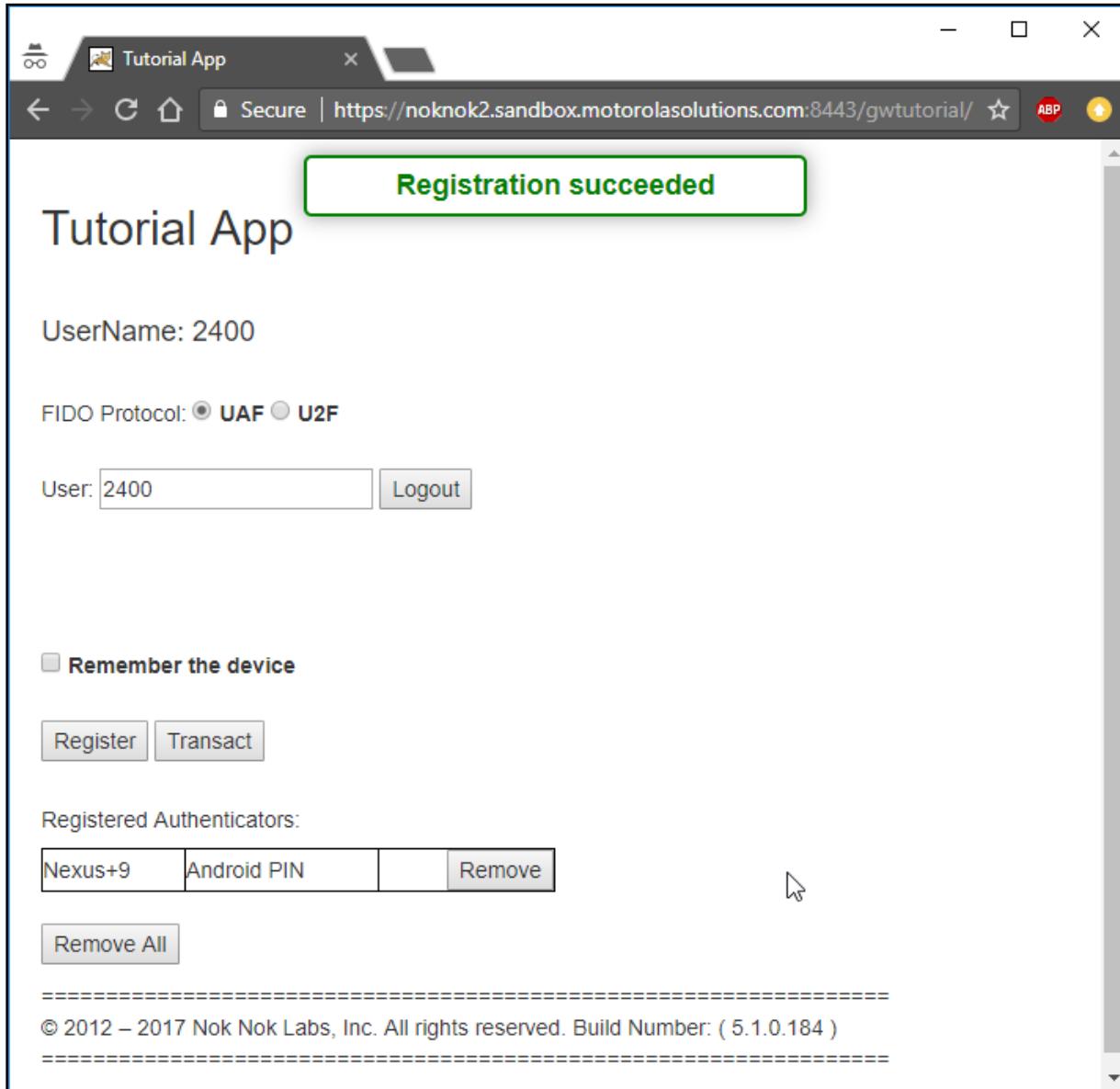
553 **Figure 2-20 FIDO UAF Fingerprint Authenticator**



554

- 555 6. If the fingerprint scan matches the user's registered fingerprint, then a new UAF key pair is
556 generated, the public key is sent to the server, and registration is completed (Figure 2-21).

557 Figure 2-21 FIDO UAF Registration Success



558

559

2.3 How App Developers Must Integrate AppAuth for SSO

560 App developers can easily integrate AppAuth to add SSO capabilities to their app. The first step to doing
561 this is reading through the AppAuth for Android documentation on GitHub [10]. After doing so, an app
562 developer can begin the integration of AppAuth. The degree of this integration can vary—for instance,

563 you may choose to utilize user attributes to personalize the user's app experience. Each separate step
564 will be displayed here.

565 *Note: In this example, we use Android Studio 3.0, Android Software Development Kit (SDK) 25, and*
566 *Gradle 2.14.1. In addition, before beginning this, you must register your app with your AS and obtain a*
567 *client ID, which will be needed in [Section 2.3.4](#).*

568 2.3.1 Adding the Library Dependency

569 1. Edit your app's *build.gradle* file, and add this line to its dependencies (note that the AppAuth
570 library will most likely be updated in the future, so you should use the most recent version for
571 your dependency, not necessarily the one in this document):

```
572 =====  
573     dependencies {  
574         ...  
575         compile 'net.openid:appauth:0.7.0'  
576     }  
577 =====
```

578 2.3.2 Adding Activities to the Manifest

579 1. First, you need to identify your AS's hostname, OAuth redirect path, and what scheme was set
580 when you registered your app. The scheme here is contrived, but it is common practice to use
581 reverse DNS style names; you should choose whatever aligns with your organization's common
582 practices. Another alternative to custom schemes is to use App Links.

583 2. Edit your *AndroidManifest.xml* file, and add these lines:

```
584 =====  
585 <manifest xmlns:android="http://schemas.android.com/apk/res/android"  
586     xmlns:tools="http://schemas.android.com/tools"  
587     package="com.example.app">  
588     ...  
589     <activity  
590         android:name="net.openid.appauth.RedirectUriReceiverActivity"  
591         tools:node="replace">  
592         <intent-filter>  
593             <action android:name="android.intent.action.VIEW" />
```

```

594      <category android:name="android.intent.category.DEFAULT" />
595      <category android:name="android.intent.category.BROWSABLE" />
596      <data
597          android:host="as.example.com"
598          android:path="/oauth2redirect"
599          android:scheme="myappsscheme" />
600      </intent-filter>
601  </activity>
602  <activity android:name=".activity.AuthResultHandlerActivity" />
603  <activity android:name=".activity.AuthCanceledHandlerActivity" />
604 </application>
605 </manifest>
606 =====

```

2.3.3 Create Activities to Handle Authorization Responses

1. Create a utility class for reusable code (**Utility**), and create activities to handle successful authorizations (**AuthResultHandlerActivity**) and canceled authorizations (**AuthCanceledHandlerActivity**):

```

611 =====
612 public class Utility {
613     public static AuthorizationService getAuthorizationService(Context context) {
615         AppAuthConfiguration appAuthConfig = new AppAuthConfiguration.Builder()
616             .setBrowserMatcher(new BrowserWhitelist(
617                 VersionedBrowserMatcher.CHROME_CUSTOM_TAB,
618                 VersionedBrowserMatcher.SAMSUNG_CUSTOM_TAB))
619             // the browser matcher above allows you to choose which in-app
620             browser
621             // tab providers will be supported by your app in its OAuth2 flow
622             .setConnectionBuilder(new ConnectionBuilder() {
623                 @NonNull
624                 public HttpURLConnection openConnection(@NonNull Uri uri)

```

```
625             throws IOException {
626
627                 URL url = new URL(uri.toString());
628
629                 HttpURLConnection connection =
630                         (HttpURLConnection) url.openConnection();
631
632                 if (connection instanceofHttpsURLConnection) {
633
634                     // optional: use your own trust manager to set a custom
635                     // SSLSocketFactory on the HttpsURLConnection
636
637                     return connection;
638
639                 }
640
641             }).build();
642
643
644             return new AuthorizationService(context, appAuthConfig);
645
646         }
647
648
649         public static AuthState restoreAuthState(Context context) {
650
651             // we use SharedPreferences to store a String version of the JSON
652             // Auth State, and here we retrieve it to convert it back to a POJO
653
654             SharedPreferences sharedPreferences =
655
656                 PreferenceManager.getDefaultSharedPreferences(context);
657
658             String jsonString = sharedPreferences.getString("AUTHSTATE", null);
659
660             if (!TextUtils.isEmpty(jsonString)) {
661
662                 try {
663
664                     return AuthState.jsonDeserialize(jsonString);
665
666                 } catch (JSONException jsonException) {
667
668                     // handle this appropriately
669
670                 }
671
672             }
673
674             return null;
675
676         }
```

```
655     }
656 =====
657     public class AuthResultHandlerActivity extends Activity {
658
659         private static final String TAG = AuthResultHandlerActivity.class.getName();
660
661         private AuthState mAuthState;
662         private AuthorizationService mAuthService;
663
664         @Override
665         protected void onCreate(Bundle savedInstanceState) {
666             super.onCreate(savedInstanceState);
667
668             AuthorizationResponse res =
669             AuthorizationResponse.fromIntent(getIntent());
670
671             AuthorizationException ex =
672             AuthorizationException.fromIntent(getIntent());
673
674             mAuthState = new AuthState(res, ex);
675             mAuthService = Utility.getAuthorizationService(this);
676
677             if (res != null) {
678                 Log.d(TAG, "Received AuthorizationResponse");
679                 performTokenRequest(res.createTokenExchangeRequest());
680             } else {
681                 Log.d(TAG, "Authorization failed: " + ex);
682             }
683
684         @Override
685         protected void onDestroy() {
686             super.onDestroy();
```

```
686         mAuthService.dispose();
687     }
688
689     private void performTokenRequest(TokenRequest request) {
690         TokenResponseCallback callback = new TokenResponseCallback() {
691             @Override
692             public void onTokenRequestCompleted(
693                 TokenResponse tokenResponse,
694                 AuthorizationException authException) {
695                 receivedTokenResponse(tokenResponse, authException);
696             }
697         };
698         mAuthService.performTokenRequest(request, callback);
699     }
700
701     private void receivedTokenResponse(TokenResponse tokenResponse,
702                                         AuthorizationException authException) {
703         Log.d(TAG, "Token request complete");
704         if (tokenResponse != null) {
705             mAuthState.update(tokenResponse, authException);
706
707             // persist auth state to SharedPreferences
708             PreferenceManager.getDefaultSharedPreferences(this)
709                 .edit()
710                 .putString("AUTHSTATE", mAuthState.jsonSerializeString())
711                 .commit();
712
713             String accessToken = mAuthState.getAccessToken();
714             if (accessToken != null) {
715                 // optional: pull claims out of JWT (name, etc.)
```

```
716             }
717         } else {
718             Log.d(TAG, " ", authException);
719         }
720     }
721 }
722 =====
723 public class AuthCanceledHandlerActivity extends Activity {
724
725     private static final String TAG =
726     AuthCanceledHandlerActivity.class.getName();
727
728     @Override
729     protected void onCreate(Bundle savedInstanceState) {
730         super.onCreate(savedInstanceState);
731
732         Log.d(TAG, "OpenID Connect authorization flow canceled");
733
734         // go back to MainActivity
735         finish();
736     }
737 }
738 =====
```

2.3.4 Executing the OAuth 2 Authorization Flow

1. In whatever activity you are using to initiate authentication, add in the necessary code to use the AppAuth SDK to execute the OAuth 2 authorization flow:

```
742 =====
743 ...
744
745 // some method, usually a "login" button, activates the OAuth2 flow
746
747 String OAUTH_AUTH_ENDPOINT =
748 "https://as.example.com:9031/as/authorization.oauth2";
```

```

749     String OAUTH_TOKEN_ENDPOINT = "https://as.example.com:9031/as/token.oauth2";
750     String OAUTH_REDIRECT_URI = "myappsscheme://app.example.com/oauth2redirect";
751     String OAUTH_CLIENT_ID = "myapp";
752     String OAUTH_PKCE_CHALLENGE_METHOD = "S256"; // options are "S256" and "plain"
753
754     // CREATE THE SERVICE CONFIGURATION
755     AuthorizationServiceConfiguration config = new
756     AuthorizationServiceConfiguration(
757         Uri.parse(OAUTH_AUTH_ENDPOINT), // auth endpoint
758         Uri.parse(OAUTH_TOKEN_ENDPOINT), // token endpoint
759         null // registration endpoint
760     );
761
762     // OPTIONAL: Add any additional parameters to the authorization request
763     HashMap<String, String> additionalParams = new HashMap<>();
764     additionalParams.put("acr_values", "urn:acr:form");
765
766     // BUILD THE AUTHORIZATION REQUEST
767     AuthorizationRequest.Builder builder = new AuthorizationRequest.Builder(
768         config,
769         OAUTH_CLIENT_ID,
770         ResponseTypeValues.CODE,
771         Uri.parse(OAUTH_REDIRECT_URI))
772         .setScopes("profile") // scope is optional, set whatever is needed by
773         your app
774         .setAdditionalParameters(additionalParams);
775
776     // SET UP PKCE CODE VERIFIER
777     String codeVerifier = CodeVerifierUtil.generateRandomCodeVerifier();
778     String codeVerifierChallenge =
779     CodeVerifierUtil.deriveCodeVerifierChallenge(codeVerifier);
780     builder.setCodeVerifier(codeVerifier, codeVerifierChallenge,
781
782         OAUTH_PKCE_CHALLENGE_METHOD);
783
784     AuthorizationRequest request = builder.build();
785
786     // PERFORM THE AUTHORIZATION REQUEST
787     // this pauses and leaves the current activity
788     Intent postAuthIntent = new Intent(this, AuthResultHandlerActivity.class);
789     Intent authCanceledIntent = new Intent(this,
790     AuthCanceledHandlerActivity.class);
791     mAuthService.performAuthorizationRequest(
792         request,
793         PendingIntent.getActivity(this, request.hashCode(), postAuthIntent, 0),
794         PendingIntent.getActivity(this, request.hashCode(), authCanceledIntent,
795         0));
796
797     ...
798
799     // when the activity resumes, check if the OAuth2 flow was successful

```

```

800     @Override
801     protected void onResume() {
802         super.onResume();
803
804         AuthState authState = Utility.restoreAuthState(this);
805         if (authState != null) {
806
807             // we are authorized!
808             // proceed to the next activity that requires an access token
809         }
810
811     ...
812 =====

```

2.3.5 Fetching and Using the Access Token

1. After you have proceeded from the prior activity, you can fetch your access token. If some time has passed since you obtained the access token, you may need to use your refresh token to get a new access token. AppAuth handles both cases the same way. Implement the following code wherever you need to use the access token:

```

818 =====
819 ...
820
821     // assuming we have an instance of a Context as mContext...
822
823     // ensure we have a fresh access token to perform any future actions
824     final AuthorizationService authService =
825     Utility.getAuthorizationService(mContext);
826     AuthState authState = Utility.restoreAuthState(mContext);
827     authState.performActionWithFreshTokens(authService, new
828     AuthState.AuthStateAction() {
829         @Override
830         public void execute(String accessToken, String idToken,
831                             AuthorizationException ex) {
832             JWT jwt = null;
833             if (ex != null) {
834                 // negotiation for fresh tokens failed, check ex for more details
835             } else {
836                 // we can now use accessToken to access remote services
837
838                 // this is typically done by including the token in an HTTP header,
839                 // or in a handshake transaction if another transport protocol is
840             used
841         }
842         authService.dispose();
843     }
844 
```

```
841         }
842     } );
843
844     ...
845 =====
```

846 **3 How to Install and Configure the OAuth 2 AS**

847 **3.1 Platform and System Requirements**

848 Ping Identity is used as the AS for this build. The AS issues access tokens to the client after successfully
849 authenticating the resource owner and obtaining authorization [\[11\]](#).

850 The requirements for Ping Identity can be categorized into three groups: software, hardware, and
851 network.

852 **3.1.1 Software Requirements**

853 The software requirements are as follows:

- 854 ▪ OS: Microsoft Windows Server, Oracle Enterprise Linux, Oracle Solaris, Red Hat Enterprise, SUSE
855 Linux Enterprise
- 856 ▪ Virtual systems: VMware, Xen, Windows Hyper-V
- 857 ▪ Java environment: Oracle Java Standard Edition (SE)
- 858 ▪ Data integration: Ping Directory, Microsoft Active Directory (AD), Oracle Directory Server,
859 Microsoft Structured Query Language (SQL) Server, Oracle Database, Oracle MySQL 5.7,
860 PostgreSQL

861 **3.1.2 Hardware Requirements**

862 The minimum hardware requirements are as follows:

- 863 ▪ Intel Pentium 4, 1.8-gigahertz (GHz) processor
- 864 ▪ 1 gigabyte (GB) of Random Access Memory (RAM)
- 865 ▪ 1 GB of available hard drive space

866 A detailed discussion on this topic and additional information can be found at
867 [https://documentation.pingidentity.com/pingfederate/pf82/index.shtml#gettingStartedGuide/concept/
868 systemRequirements.html](https://documentation.pingidentity.com/pingfederate/pf82/index.shtml#gettingStartedGuide/concept_systemRequirements.html).

869 **3.1.3 Network Requirements**

870 Ping Identity identifies several ports to be open for different purposes. These purposes can include
871 communication with the administrative console, runtime engine, cluster engine, and Kerberos engine.

872 A detailed discussion on each port can be found at

873 https://documentation.pingidentity.com/pingfederate/pf84/index.shtml#gettingStartedGuide/pf_t_inst_allPingFederateRedHatEnterpriseLinux.html.

875 In this implementation, we needed ports to be opened to communicate with the administrative console
876 and the runtime engine.

877 For this experimentation, we have used the configuration identified in the following subsections.

878 **3.1.3.1 Software Configuration**

879 The software configuration is as follows:

- 880 □ OS: CentOS Linux Release 7.3.1611 (Core)
- 881 □ Virtual systems: Vmware ESXI 6.5
- 882 □ Java environment: OpenJDK Version 1.8.0_131
- 883 □ Data integration: Active Directory (AD)

884 **3.1.3.2 Hardware Configuration**

885 The hardware configuration is as follows:

- 886 □ Processor: Intel(R) Xeon(R) central processing unit (CPU) E5-2420 0 at 1.90 GHz
- 887 □ Memory: 2 GB
- 888 □ Hard drive: 25 GB

889 **3.1.3.3 Network Configuration**

890 The network configuration is as follows:

- 891 □ 9031: This port allows access to the runtime engine; this port must be accessible to client
892 devices and federation partners.
- 893 □ 9999: This port allows the traffic to the administrative console; only PingFederate administrators
894 need access.

895 **3.2 How to Install the OAuth 2 AS**

896 Before the installation of Ping Identity AS, the prerequisites identified in the following subsections need
897 to be fulfilled.

898 **3.2.1 Java Installation**

899 Java 8 can be installed in several ways on CentOS 7 using *yum*. Yum is a package manager on the
900 CentOS 7 platform that automates software processes, such as installation, upgrade, and removal, in a
901 consistent way.

- 902 1. Download the Java Development Kit (JDK) in the appropriate format for your environment, from
903 Oracle's website; for CentOS, the Red Hat Package Manager (RPM) download can be used:
904 <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>.

- 905 2. As root, install the RPM by using the following command, substituting the actual version of the
906 downloaded file:

907 **rpm -ivh jdk-8u151-linux-x64.rpm**

- 908 3. Alternatively, the JDK can be downloaded in *.tar.gz* format and unzipped in the appropriate
909 location (i.e., */usr/share* on CentOS 7).

910 **3.2.2 Java Post Installation**

911 The *alternatives* command maintains symbolic links determining default commands. This command
912 can be used to select the default Java command. This is helpful even in cases where there are multiple
913 installations of Java on the system.

- 914 1. Use the following command to select the default Java command:

915 **alternatives --config java**

916 There are 3 programs which provide 'java'.

Selection	Command
1	/usr/java/jre1.8.0_111/bin/java
2	java-1.8.0-openjdk.x86_64 (/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.131-3.b12.el7_3.x86_64/jre/bin/java)
3	/usr/java/jdk1.8.0_131/jre/bin/java

917 Selection Command
918 -----
919 1 /usr/java/jre1.8.0_111/bin/java
920 *+ 2 java-1.8.0-openjdk.x86_64 (/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.131-3.b12.el7_3.x86_64/jre/bin/java)
921 3 /usr/java/jdk1.8.0_131/jre/bin/java
922 Enter to keep the current selection[+], or type selection number:

923 This presents the user with a configuration menu for choosing a Java instance. Once a selection
924 is made, the link becomes the default command system wide.

- 926 2. To make Java available to all users, the JAVA_HOME environment variable was set by using the
927 following command:
- 928 echo export JAVA_HOME="/usr/java/latest" > /etc/profile.d/javaenv.sh
- 929 3. For cryptographic functions, download the *Java Cryptography Extension (JCE) Unlimited Strength
930 Jurisdiction Policy Files 8* from
931 <http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html>.
- 932 4. Uncompress and extract the downloaded file. The installation procedure is described in the
933 Readme document. In the lab, *local_policy.jar* was extracted to the default location, <*java-
934 home*>/lib/security.Network Configuration.
- 935 5. Check if the firewall is running or not by using the command below. If it is up, it will return a
936 status that shows it is running:
- 937 firewall-cmd --state
- 938 a. If it is not running, activate the firewall by using the following command:
- 939 sudo systemctl start firewalld.service
- 940 6. Check if the required ports, 9031 and 9999, are open by using the following command:
- 941 firewall-cmd --list-ports
- 942 a. This command will return the following values:
- 943 6031/tcp 9999/udp 9031/tcp 6031/udp 9998/udp 9031/udp 9999/tcp 9998/tcp
944 8080/tcp
- 945 From the returned ports, we can determine which ports and protocols are open.
- 946 b. In case the required ports are not open, issue the command below. It should return
947 success.
- 948 **firewall-cmd --zone=public --permanent --add-port=9031/tcp**
- 949 success
- 950 7. Reload the firewall by using the following command to make the rule change take effect:
- 951 **firewall-cmd --reload**
- 952 Success
- 953 a. Now, when the open ports are listed, the required ports should show up:
- 954 **firewall-cmd --zone=public --list-ports**
- 955 6031/tcp 9999/udp 9031/tcp 6031/udp 9998/udp 9031/udp 9999/tcp 9998/tcp
956 8080/tcp 5000/tcp

957 3.2.3 PingFederate Installation

958 Ping installation documentation is available at

959 https://docs.pingidentity.com/bundle/pf_sm_installPingFederate_pf82/page/pf_t_installPingFederateRedHatEnterpriseLinux.html?#.

961 Some important points are listed below:

- 962 ▪ Obtain a Ping Identity license. It can be acquired from
<https://www.pingidentity.com/en/account/sign-on.html>.
- 963 ▪ For this experiment, installation was done using the zip file. Installation was done at */usr/share*.
- 964 ▪ The license was updated.
- 965 ▪ The PingFederate service can be configured as a service that automatically starts at system boot.
PingFederate provides instructions for doing this on different OSs. In the lab, the Linux
instructions at the link provided below were used. Note that, while the instructions were written
for an *init.d*-based system, these instructions will also work on a systemd-based system.
https://docs.pingidentity.com/bundle/pf_sm_installPingFederate_pf82/page/pf_t_installPingFederateServiceLinuxManually.html?#

972 The following configuration procedures are completed in the PingFederate administrative console,
973 which is available at *https://<ping-server-hostname>:9999/pingfederate/app*.

974 3.2.4 Certificate Installation

975 During installation, PingFederate generates a self-signed TLS certificate, which is not trusted by desktop
976 or mobile device browsers. A certificate should be obtained from a trusted internal or external CA, and
977 should be installed on the PingFederate server. The private key and signed certificate can be uploaded
978 and activated for use on the run-time server port and the admin port by navigating to **Server Settings** in
979 the console and clicking on **SSL Server Certificates**.

980 In addition, most server roles described in this guide will require the creation of a signing certificate. This
981 is required for a SAML or OIDC IdP, and for an OAuth AS if access tokens will be issued as JWTs. To
982 create or import a signing certificate, under **Server Configuration – Certificate Management**, click
983 **Signing & Decryption Keys & Certificates**. A self-signed certificate can be created, or a trusted certificate
984 can be obtained and uploaded there.

985 3.3 How to Configure the OAuth 2 AS

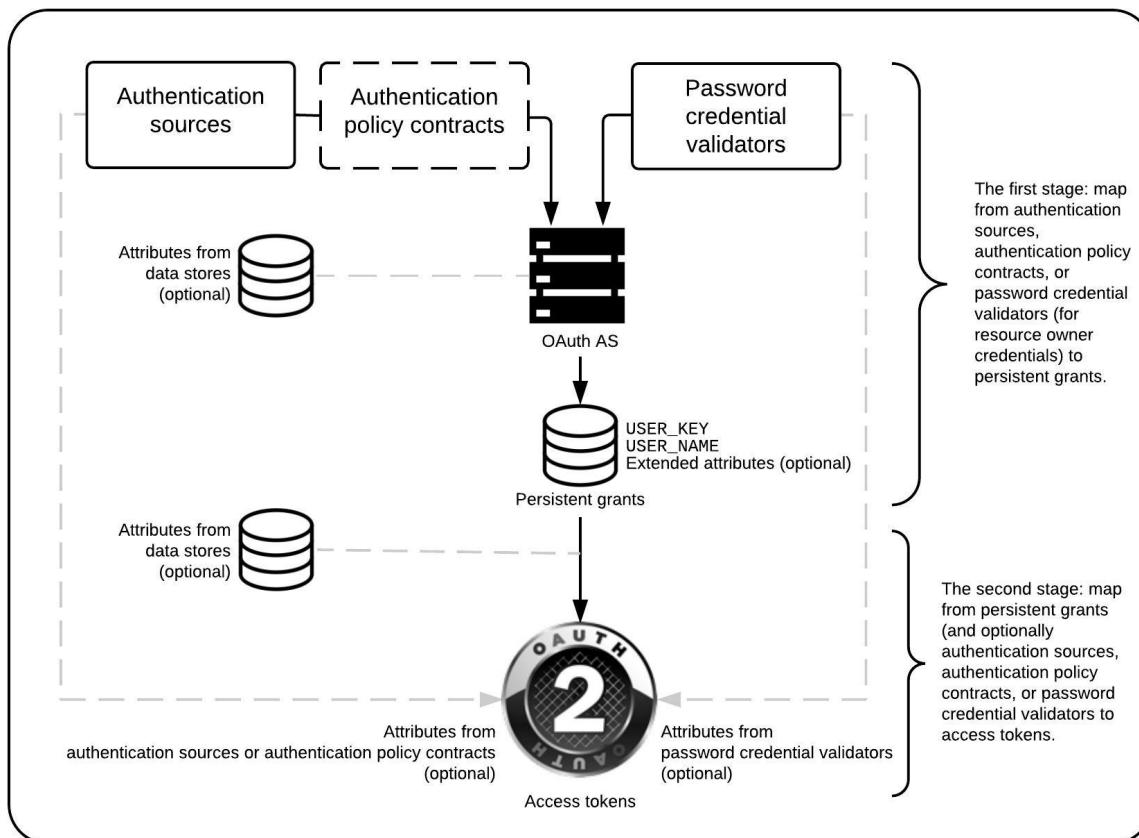
986 Configuration of a Ping OAuth 2 AS is described at

987 https://documentation.pingidentity.com/pingfederate/pf82/index.shtml#concept_usingOauthMenuSelections.html#concept_usingOauthMenuSelections.

989 This guide documents the configuration for an AS serving the role of the *idm.sandbox* server hosted in
 990 the Motorola Solutions cloud instance, as depicted in Figure 1-1. This AS is configured to support the
 991 three usage scenarios—local user authentication at the AS, redirection to a SAML IdP, and redirection to
 992 an OIDC IdP—and to initiate the correct login flow based on an IdP discovery mechanism.

993 An understanding of the PingFederate OAuth implementation helps provide context for the
 994 configurations documented in this guide. PingFederate supports several different authentication flows
 995 and mechanisms, but there is a common framework for how user attributes are mapped into OAuth
 996 tokens. This framework is depicted in Figure 3-1, which is taken from PingFederate’s documentation at
[997 https://documentation.pingidentity.com/pingfederate/pf83/index.shtml#concept_mappingOauthAttributes.html#concept_mappingOauthAttributes](https://documentation.pingidentity.com/pingfederate/pf83/index.shtml#concept_mappingOauthAttributes.html#concept_mappingOauthAttributes)
 998

999 **Figure 3-1 Access Token Attribute Mapping Framework**



1000

1001 The overall OAuth processing flow at the AS is as follows:

- 1002 1. The AS receives an OAuth authorization request from an unauthenticated user.

1003 2. The AS authenticates the user through the configured authentication adapters, IdP connections,
1004 and/or authentication policies.

1005 3. Information from adapters or policy contracts, optionally combined with user information
1006 retrieved from data stores such as Lightweight Directory Access Protocol (LDAP), are used to
1007 build a persistent grant context. The two mandatory attributes in the persistent grant context are
1008 listed below:

1009 ▪ **USER_KEY** – This is a globally unique user identifier. For ASs that interact with multiple
1010 IdPs, this name should be resistant to naming collisions across user organizations (e.g.,
1011 email address or distinguished name).

1012 ▪ **USER_NAME** – If the user is prompted to authorize the request, this name will be
1013 displayed on the page, so a user-friendly name, such as [givenName lastName], could be
1014 used here; the name does not need to be unique.

1015 4. If authorization prompts are enabled, the user is prompted to approve the authorization
1016 request; for this lab build, these prompts were disabled on the assumption that fast access to
1017 apps is a high priority for the PSFR community.

1018 5. If the request is authorized, a second mapping process takes place to populate the access token
1019 with information from the persistent grant and, optionally, from adapters, policy contracts, or
1020 data stores.

1021 Note that persistent grant attributes are stored and can be retrieved and reused when the client uses a
1022 refresh token to obtain a new access token, whereas attributes that are looked up in the second stage
1023 would be looked up again during the token refresh request. Storing attributes in the persistent grant can
1024 therefore reduce the need for repeated directory queries; however, it may be preferable to always
1025 query some attributes that are subject to change (like account status) again when a new access token is
1026 requested. In addition, it is important to note that storing persistent grant attributes requires a
1027 supported relational database or LDAP data store. Refer to the following documentation for a list of
1028 supported data stores:

1029 <https://documentation.pingidentity.com/pingfederate/pf82/index.shtml#gettingStartedGuide/task/installingPingFederate.html>.

1031 The following steps go through the configuration of the AS.

1032 1. Enable the PingFederate installation to work as an AS. This can be done in the following steps:

1033 a. Under **Main**, click the **Server Configuration** section tab, and then click **Server Settings**.

1034 b. In **Server Settings**, click the **Roles & Protocols** tab. The Roles & Protocols screen will ap-
1035 pear as shown in Figure 3-2.

1036 i. Click **ENABLE OAUTH 2.0 AUTHORIZATION SERVER (AS) ROLE**.

- 1037 ii. Click **ENABLE IDENTITY PROVIDER (IDP) ROLE AND SUPPORT THE FOLLOWING**,
1038 and then under it, click **SAML 2.0**. Although this server does not act as a SAML
1039 IdP, it is necessary to enable the IdP role and at least one protocol to configure
1040 the local user authentication use case.
- 1041 iii. Click **ENABLE SERVICE PROVIDER (SP) ROLE AND SUPPORT THE FOLLOWING**,
1042 and then under it, click **SAML 2.0** and **OPENID CONNECT**; this enables integra-
1043 tion with both types of IdPs.

1044

Figure 3-2 Server Roles for AS

The screenshot shows the 'Server Settings' page in the PingFederate interface. The left sidebar has a 'MAIN' category with links for IdP Configuration, SP Configuration, OAuth Settings, and Server Configuration (which is selected). The main content area is titled 'Server Settings' and contains a navigation bar with tabs: System Administration, System Info, Runtime Notifications, Runtime Reporting, and Account Management. Under 'System Administration', there are five sub-tabs: Roles & Protocols (selected), Federation Info, System Options, Metadata Signing, and Metadata Lifetime. The 'Summary' tab is also visible.

The main configuration area is titled 'Select the role(s) and protocol(s) that you intend to use with your federation partners.' It includes sections for enabling roles and specifying supported protocols.

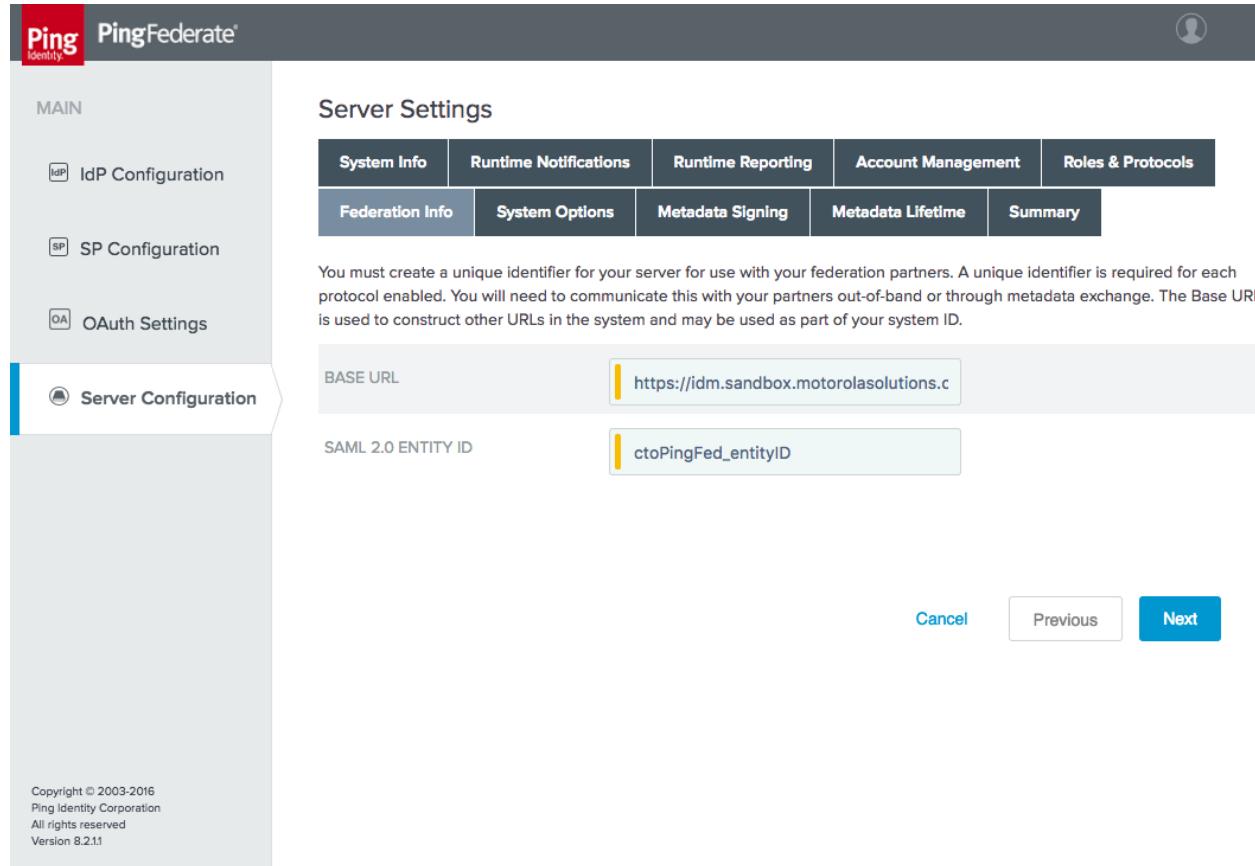
- ENABLE OAUTH 2.0 AUTHORIZATION SERVER (AS) ROLE:**
- OPENID CONNECT:**
- ENABLE IDENTITY PROVIDER (IDP) ROLE AND SUPPORT THE FOLLOWING:**
 - SAML 2.0
 - AUTO-CONNECT PROFILE
 - SAML 1.1
 - SAML 1.0
 - WS-FEDERATION
 - OUTBOUND PROVISIONING
 - WS-TRUST
- ENABLE SERVICE PROVIDER (SP) ROLE AND SUPPORT THE FOLLOWING:**
 - SAML 2.0
 - AUTO-CONNECT PROFILE
 - ATTRIBUTE REQUESTER MAPPING FOR X.509 ATTRIBUTE SHARING PROFILE (XASP)
 - SAML 1.1
 - SAML 1.0
 - WS-FEDERATION
 - WS-TRUST
 - INBOUND PROVISIONING
 - OPENID CONNECT
- ENABLE IDP DISCOVERY ROLE (SAML 2.0 ONLY):**

At the bottom right are buttons for Cancel, Previous, Next, and Save.

1045

- 1046 c. Also under **Server Settings**, on the **Federation Info** tab, enter the **BASE URL** and **SAML**
 1047 **2.0 ENTITY ID** (Figure 3-3). The **BASE URL** should use a public DNS name that is resolvable
 1048 by any federation partners. The **SAML 2.0 ENTITY ID** is simply an identifier string that
 1049 must be unique among federation partners; it is recommended to be a Uniform Re-
 1050 source Identifier (URI), per the SAML 2.0 Core specification [\[12\]](#).

1051 **Figure 3-3 Federation Info**



The screenshot shows the 'PingFederate' interface. On the left, a sidebar titled 'MAIN' contains links for 'IdP Configuration', 'SP Configuration', 'OAuth Settings', and 'Server Configuration'. The 'Server Configuration' link is highlighted with a blue bar. The main content area is titled 'Server Settings' and has a sub-navigation bar with tabs: 'System Info' (selected), 'Runtime Notifications', 'Runtime Reporting', 'Account Management', and 'Roles & Protocols'. Below this, another tab bar for 'Federation Info' (selected), 'System Options', 'Metadata Signing', 'Metadata Lifetime', and 'Summary' is shown. A note states: 'You must create a unique identifier for your server for use with your federation partners. A unique identifier is required for each protocol enabled. You will need to communicate this with your partners out-of-band or through metadata exchange. The Base URL is used to construct other URLs in the system and may be used as part of your system ID.' Two input fields are present: 'BASE URL' containing 'https://idm.sandbox.motorolasolutions.c' and 'SAML 2.0 ENTITY ID' containing 'ctoPingFed_entityID'. At the bottom right are buttons for 'Cancel', 'Previous', and 'Next'.

- 1052 2. The next step is to configure the OAuth AS. Click the **OAuth Settings** section tab under **Main**.
 1053 a. Click **Authorization Server Settings** under the **Authorization Server** header. This displays
 1054 the **Authorization Server Settings** (Figure 3-4).

1056 Figure 3-4 AS Settings

PingFederate®

The screenshot shows the PingFederate configuration interface. The left sidebar has a 'MAIN' section with 'IdP Configuration', 'SP Configuration', 'OAuth Settings' (which is highlighted with a blue bar), and 'Server Configuration'. The main content area is titled 'Authorization Server Settings' and describes providing general configuration and policy for the PingFederate Authorization Server. It includes sections for 'Authorization Code Timeout (Seconds)' (set to 60), 'Authorization Code Entropy (Bytes)' (set to 30), 'Refresh Token and Persistent Grant Settings' (with a persistent grant lifetime of 42 days), and various other configuration options like rolling refresh tokens and reuse of access grants. At the bottom, there's a table for 'Persistent Grant Extended Attributes' with an 'Add' button, and an 'OAuth Administrative Web Services Settings' section with a dropdown for 'PASSWORD CREDENTIAL VALIDATOR' set to '- SELECT -'. Navigation icons for 'Cancel' and 'Save' are at the bottom right.

MAIN

- IdP Configuration
- SP Configuration
- OAuth Settings**
- Server Configuration

Authorization Server Settings

Provide general configuration and policy for the PingFederate Authorization Server.

AUTHORIZATION CODE TIMEOUT (SECONDS)

AUTHORIZATION CODE ENTROPY (BYTES)

Refresh Token and Persistent Grant Settings

PERSISTENT GRANT LIFETIME (BLANK FOR INDEFINITE) Days

REFRESH TOKEN LENGTH (CHARACTERS)

ROLL REFRESH TOKEN VALUES (DEFAULT POLICY)

MINIMUM INTERVAL TO ROLL REFRESH TOKENS (HOURS)

REUSE EXISTING PERSISTENT ACCESS GRANTS FOR GRANT TYPES IMPLICIT AUTHORIZATION CODE RESOURCE OWNER PASSWORD CREDENTIALS

BYPASS AUTHORIZATION FOR PREVIOUSLY APPROVED PERSISTENT GRANTS

ALLOW UNIDENTIFIED CLIENTS TO MAKE RESOURCE OWNER PASSWORD CREDENTIALS GRANTS

ALLOW UNIDENTIFIED CLIENTS TO REQUEST EXTENSION GRANTS

Persistent Grant Extended Attributes

Attribute	Action
<input type="text"/>	<input type="button" value="Add"/>

OAuth Administrative Web Services Settings

PASSWORD CREDENTIAL VALIDATOR

1057

1058 The default settings are suitable for the lab build architecture; organizations may wish
1059 to customize these default settings in accordance with organizational security policy or
1060 usage requirements. Some notes on individual settings are provided below:

- 1061 ■ **AUTHORIZATION CODE TIMEOUT (SECONDS):** Once an authorization code has
1062 been returned to a client, it must be exchanged for an access token within this
1063 interval. This reduces the risk of an unauthorized client obtaining an access
1064 token through brute-force guessing or intercepting a valid client's code. *Proof*
1065 *Key for Code Exchange (PKCE)* [13], as implemented by the AppAuth library, is
1066 another useful mechanism to protect the authorization code.
 - 1067 ■ **AUTHORIZATION CODE ENTROPY (BYTES):** Length of the authorization code
1068 returned by the AS to the client, in bytes
 - 1069 ■ **REFRESH TOKEN LENGTH (CHARACTERS):** Length of the refresh token, in
1070 characters
 - 1071 ■ **ROLL REFRESH TOKEN VALUES (DEFAULT POLICY):** When selected, the OAuth
1072 AS generates a new refresh token value when a new access token is obtained.
 - 1073 ■ **MINIMUM INTERVAL TO ROLL REFRESH TOKENS (HOURS):** The minimum
1074 number of hours that must pass before a new refresh token value can be issued.
 - 1075 ■ **REUSE EXISTING PERSISTENT ACCESS GRANTS FOR GRANT TYPES:**
 - 1076 • **IMPLICIT:** Consent from the user is requested only for the first OAuth
1077 resource request associated with the grant.
 - 1078 • **AUTHORIZATION CODE:** Same as above if the **BYPASS AUTHORIZATION**
1079 **FOR PREVIOUSLY APPROVED PERSISTENT GRANTS** is selected; this can
1080 be used to prompt the user for authorization only once to avoid
1081 repeated prompts for the same client.
 - 1082 ■ **PASSWORD CREDENTIAL VALIDATOR:** Required for Hypertext Transfer Protocol
1083 (HTTP) Basic authentication if the OAuth Representational State Transfer (REST)
1084 Web Service is used for managing client apps; this functionality was not used for
1085 this build.
- 1086 3. Next, configure scopes, as required, for the app. Click the **OAuth Settings** section tab, and then
1087 click **Scope Management**. The specific scope values will be determined by the client app
1088 developer. Generally speaking, scopes refer to different authorizations that can be requested by
1089 the client and granted by the user. Access tokens are associated with the scopes for which they
1090 are authorized, which can limit the authorities granted to clients. Figure 3-5 shows several
1091 scopes that were added to the AS for this lab build that have specific meanings in the PSX apps
1092 suite.

1093 **Figure 3-5 Scopes**

The screenshot shows the PingFederate configuration interface. The left sidebar has tabs for MAIN, IdP Configuration, SP Configuration, OAuth Settings (which is selected), and Server Configuration. The main content area has a table with columns for Scope Value and Scope Description. The table contains the following data:

Scope Value	Scope Description
bio_only	Add to scope to select FIDO biometric only policy
https://motorolasolutions.com/v1/calcium	Access your Whiteboards
location	This application is requesting access to your location information
msi_uns.connect	msi_uns.connect
msi_uns.gateway	msi_uns.gateway
msi_uns.location	msi_uns.location
msi_uns.messaging	msi_uns.messaging
msi_uns.presence	msi_uns.presence
msi_uns.register	msi_uns.register
msi_uns.telemetry	msi_uns.telemetry
msi_unsapi_groupmgt.read	msi_unsapi_groupmgt.read
msi_unsapi_groupmgt.write	msi_unsapi_groupmgt.write
msi_unsapi_location.watch	msi_unsapi_location.watch

At the bottom left of the main area, there is a copyright notice: Copyright © 2003-2016 Ping Identity Corporation. All rights reserved. Version 8.2.1!

1094

- 1095 4. Define an Access Token Management profile. This profile determines whether access tokens are
 1096 issued as simple reference token strings or as JWTs. For this lab build, JWTs were used. JWTs are
 1097 signed and optionally encrypted, so resource servers can validate them locally and they can
 1098 contain user attributes and other information. Reference tokens are also a viable option, but
 1099 resource servers must contact the AS's introspection endpoint to determine whether they are
 1100 valid, and must obtain the granted scopes and any other information associated with them. The
 1101 Access Token Management Profile also defines any additional attributes that will be associated
 1102 with the token.
- 1103 a. Create an Access Token Manager by following these steps:
- 1104 i. Click the **OAuth Settings** section tab, click **Access Token Management**, and then
 1105 click **Create New Instance**.
- 1106 ii. On the **Type** tab, give the instance a meaningful name and ID, and select the to-
 1107 ken type (Figure 3-6).

1108 **Figure 3-6 Access Token Management Instance**

Access Token Management | Create Access Token Management Instance

Type Instance Configuration Access Token Attribute Contract Resource URIs Access Control Summary

Enter an Access Token Management Instance Name and Id, select the plugin Access Token Management Type, and a parent if applicable. The types available are limited to the plugins currently installed on your server.

INSTANCE NAME	fidoJwt
INSTANCE ID	fidoJwt
TYPE	JSON Web Tokens
PARENT INSTANCE	None

[Visit PingIdentity.com for additional types](#)

Cancel Next

1109

- 1110 5. On the next tab, **Instance Configuration**, select a symmetric key or certificate to use for JWT
 1111 signing (Figure 3-7). In this instance, a signing certificate was created as described in
 1112 [Section 3.2.4](#). Tokens can also optionally be encrypted using JSON Web Encryption (JWE) [\[14\]](#); in
 1113 this case, the client developer would provide a certificate in order to receive encrypted
 1114 messages. JWE was not used in the lab build.

1115 Figure 3-7 Access Token Manager Instance Configuration

PingFederate[®]

MAIN

- IdP Configuration**
- SP Configuration**
- OAuth Settings**
- Server Configuration**

Access Token Management | Create Access Token Management Instance

Type **Instance Configuration** **Access Token Attribute Contract** **Resource URIs** **Access Control** **Summary**

Complete the configuration necessary to issue and validate access tokens. This configuration was designed into, and is specific to, the selected Access Token Management plugin.

A JSON Web Token (JWT) Bearer Access Token Management Plug-in that enables PingFederate to issue (and optionally validate) cryptographically secure self-contained OAuth access tokens.

Symmetric Keys
(A group of keys for use with symmetric encryption and MAC algorithms.)

KEY ID	KEY	ENCODING	Action
(An identifier for the given key)	(Encoded symmetric key)	(How the binary key is encoded as a string)	

[Add a new row to 'Symmetric Keys'](#)

CERTIFICATES
(A group of certificates and their corresponding public/private key pairs for use with signatures)

KEY ID	CERTIFICATE	Action
jwt signer	CN=as1.cpsdd.mso, OU=NCCoE, O=NIST, L=Rockville, ST=Maryland, C=US	Edit Delete

[Add a new row to 'Certificates'](#)

Field Name **Field Value** **Description**

TOKEN LIFETIME	120	Defines how long, in minutes, an access token is valid.
JWS ALGORITHM	RSA using SHA-256	The HMAC or signing algorithm used to protect the integrity of the token. For HMAC, the active symmetric key must be selected below. For RSA or EC, the active signing certificate must be selected. Integrity protection can also be achieved using symmetric encryption, in which case this field can be left unselected.
ACTIVE SYMMETRIC KEY ID	— Select One —	The Key ID of the key to use when producing JWTs using an HMAC-based algorithm.
ACTIVE SIGNING CERTIFICATE KEY ID	jwt signer	The Key ID of the key pair and certificate to use when producing JWTs using an RSA-based or EC-based algorithm.
JWE ALGORITHM	— Select One —	The algorithm used to encrypt or otherwise determine the value of the content encryption key.
JWE CONTENT ENCRYPTION ALGORITHM	— Select One —	The content encryption algorithm used to perform authenticated encryption on the plaintext payload of the token.
ACTIVE SYMMETRIC ENCRYPTION KEY ID	— Select One —	The Key ID of the key to use when using a symmetric encryption algorithm.
ASYMMETRIC ENCRYPTION KEY		An asymmetric encryption public key, which can be in either JWK format or a certificate.
ASYMMETRIC ENCRYPTION JWKS URL		The HTTPS URL of a JSON Web Key Set endpoint that has public key(s) for encryption.

[Manage Signing Certificates](#) [Show Advanced Fields](#)

[Cancel](#) [Previous](#) **Next**

1116

1117 6. On the **Access Token Attribute Contract** tab, add the two values **realm** and **sub** to the attribute
 1118 contract (Figure 3-8).

1119 **Figure 3-8 Access Token Manager Attribute Contract**

The screenshot shows the 'Access Token Management | Create Access Token Management Instance' page. The left sidebar has tabs for 'IdP Configuration', 'SP Configuration', 'OAuth Settings' (which is selected), and 'Server Configuration'. The main area has tabs for 'Type', 'Instance Configuration', 'Access Token Attribute Contract' (selected), 'Resource URIs', 'Access Control', and 'Summary'. Under 'Access Token Attribute Contract', it says 'Provide the names of the attributes that will be carried in (or referenced by) the OAuth access token.' A table lists 'realm' and 'sub' with 'Edit | Delete' links. There is an 'Add' button and navigation buttons for 'Cancel', 'Previous', and 'Next'.

1120

1121 7. The **Resource URIs** and **Access Control** tabs were not used for this build. Click **Save** to complete
 1122 the Access Token Manager.

1123 8. Next, one or more OAuth clients need to be registered with the AS. In the Motorola Solutions
 1124 use case, the PSX Cockpit app is registered as a client. OAuth Client registration is described for
 1125 PingFederate at:
https://documentation.pingidentity.com/pingfederate/pf82/index.shtml#concept_configuringClient.html.

1126

1127

1128 To create a new client, click the **OAuth Settings** section tab, click **Clients**, and then click **Create**
 1129 **New**. Clients are displayed on the rightmost side of the screen in the **OAuth Settings** window.
 1130 Once **Create New** is clicked, the screen shown in Figure 3-9 and Figure 3-10 will appear. Due to
 1131 the vertical size of the pages of this document, the screenshot is divided into two parts for
 1132 legibility.

1133 Figure 3-9 OAuth Client Registration, Part 1

MAIN

IdP Configuration

SP Configuration

OAuth Settings

Server Configuration

Client

Manage the configuration and policy information about a client.

CLIENT ID: ssoclient_nist

CLIENT AUTHENTICATION: NONE CLIENT SECRET

SECRET
..... [Generate Secret](#)

CHANGE SECRET

CLIENT TLS CERTIFICATE

ISSUER: - SELECT -

SUBJECT DN

You can also extract the Subject DN from a certificate file.

No file selected [Choose file](#)

[Extract](#)

Copyright © 2003-2016
Ping Identity Corporation
All rights reserved
Version 8.2.1.1

1134

1135 Figure 3-10 OAuth Client Registration, Part 2

<div style="background-color: #f0f0f0; height: 150px;"></div>	<table border="0"> <tr> <td>NAME</td> <td><input type="text" value="ssoclient_nist"/></td> </tr> <tr> <td>DESCRIPTION</td> <td><input type="text"/></td> </tr> <tr> <td>REDIRECT URIS</td> <td> <table border="0"> <tr> <td>Redirection URIs</td> <td>Action</td> </tr> <tr> <td>http://localhost/</td> <td>Edit Delete</td> </tr> <tr> <td>napps://localhost/</td> <td>Edit Delete</td> </tr> <tr> <td><input type="text"/></td> <td>Add</td> </tr> </table> </td> </tr> <tr> <td>LOGO URL</td> <td><input type="text"/></td> </tr> <tr> <td>BYPASS AUTHORIZATION APPROVAL</td> <td> <input checked="" type="checkbox"/> Bypass <input type="checkbox"/> Restrict </td> </tr> <tr> <td>RESTRICT SCOPES</td> <td> <input type="checkbox"/> </td> </tr> <tr> <td>ALLOWED GRANT TYPES</td> <td> <input checked="" type="checkbox"/> Authorization Code <input type="checkbox"/> Resource Owner Password Credentials <input checked="" type="checkbox"/> Refresh Token <input checked="" type="checkbox"/> Implicit <input type="checkbox"/> Client Credentials <input type="checkbox"/> Access Token Validation (Client is a Resource Server) <input type="checkbox"/> Extension Grants </td> </tr> <tr> <td>DEFAULT ACCESS TOKEN MANAGER</td> <td><input type="text" value="fidoJwt"/></td> </tr> <tr> <td>PERSISTENT GRANTS EXPIRATION</td> <td> <input checked="" type="radio"/> Use Global Setting <input type="radio"/> Grants Do Not Expire <input type="radio"/> <input type="text"/> Days </td> </tr> <tr> <td>REFRESH TOKEN ROLLING POLICY</td> <td> <input checked="" type="radio"/> Use Global Setting <input type="radio"/> Don't Roll <input type="radio"/> Roll </td> </tr> <tr> <td>OPENID CONNECT</td> <td> ID Token Signing Algorithm <input type="text" value="HMAC using SHA-256"/> </td> </tr> <tr> <td colspan="2"> Policy <input type="text" value="fidoPolicy"/> </td> </tr> <tr> <td colspan="2"> <input type="checkbox"/> Grant Access to Session Revocation API </td> </tr> </table>	NAME	<input type="text" value="ssoclient_nist"/>	DESCRIPTION	<input type="text"/>	REDIRECT URIS	<table border="0"> <tr> <td>Redirection URIs</td> <td>Action</td> </tr> <tr> <td>http://localhost/</td> <td>Edit Delete</td> </tr> <tr> <td>napps://localhost/</td> <td>Edit Delete</td> </tr> <tr> <td><input type="text"/></td> <td>Add</td> </tr> </table>	Redirection URIs	Action	http://localhost/	Edit Delete	napps://localhost/	Edit Delete	<input type="text"/>	Add	LOGO URL	<input type="text"/>	BYPASS AUTHORIZATION APPROVAL	<input checked="" type="checkbox"/> Bypass <input type="checkbox"/> Restrict	RESTRICT SCOPES	<input type="checkbox"/>	ALLOWED GRANT TYPES	<input checked="" type="checkbox"/> Authorization Code <input type="checkbox"/> Resource Owner Password Credentials <input checked="" type="checkbox"/> Refresh Token <input checked="" type="checkbox"/> Implicit <input type="checkbox"/> Client Credentials <input type="checkbox"/> Access Token Validation (Client is a Resource Server) <input type="checkbox"/> Extension Grants	DEFAULT ACCESS TOKEN MANAGER	<input type="text" value="fidoJwt"/>	PERSISTENT GRANTS EXPIRATION	<input checked="" type="radio"/> Use Global Setting <input type="radio"/> Grants Do Not Expire <input type="radio"/> <input type="text"/> Days	REFRESH TOKEN ROLLING POLICY	<input checked="" type="radio"/> Use Global Setting <input type="radio"/> Don't Roll <input type="radio"/> Roll	OPENID CONNECT	ID Token Signing Algorithm <input type="text" value="HMAC using SHA-256"/>	Policy <input type="text" value="fidoPolicy"/>		<input type="checkbox"/> Grant Access to Session Revocation API	
NAME	<input type="text" value="ssoclient_nist"/>																																		
DESCRIPTION	<input type="text"/>																																		
REDIRECT URIS	<table border="0"> <tr> <td>Redirection URIs</td> <td>Action</td> </tr> <tr> <td>http://localhost/</td> <td>Edit Delete</td> </tr> <tr> <td>napps://localhost/</td> <td>Edit Delete</td> </tr> <tr> <td><input type="text"/></td> <td>Add</td> </tr> </table>	Redirection URIs	Action	http://localhost/	Edit Delete	napps://localhost/	Edit Delete	<input type="text"/>	Add																										
Redirection URIs	Action																																		
http://localhost/	Edit Delete																																		
napps://localhost/	Edit Delete																																		
<input type="text"/>	Add																																		
LOGO URL	<input type="text"/>																																		
BYPASS AUTHORIZATION APPROVAL	<input checked="" type="checkbox"/> Bypass <input type="checkbox"/> Restrict																																		
RESTRICT SCOPES	<input type="checkbox"/>																																		
ALLOWED GRANT TYPES	<input checked="" type="checkbox"/> Authorization Code <input type="checkbox"/> Resource Owner Password Credentials <input checked="" type="checkbox"/> Refresh Token <input checked="" type="checkbox"/> Implicit <input type="checkbox"/> Client Credentials <input type="checkbox"/> Access Token Validation (Client is a Resource Server) <input type="checkbox"/> Extension Grants																																		
DEFAULT ACCESS TOKEN MANAGER	<input type="text" value="fidoJwt"/>																																		
PERSISTENT GRANTS EXPIRATION	<input checked="" type="radio"/> Use Global Setting <input type="radio"/> Grants Do Not Expire <input type="radio"/> <input type="text"/> Days																																		
REFRESH TOKEN ROLLING POLICY	<input checked="" type="radio"/> Use Global Setting <input type="radio"/> Don't Roll <input type="radio"/> Roll																																		
OPENID CONNECT	ID Token Signing Algorithm <input type="text" value="HMAC using SHA-256"/>																																		
Policy <input type="text" value="fidoPolicy"/>																																			
<input type="checkbox"/> Grant Access to Session Revocation API																																			

1136

- 1137 The following are notes on the parameters on this screen:
- 1138 ▪ **CLIENT ID:** This is a required parameter. This is the unique identifier accompanied with
1139 each request that is presented to the AS's token and authorization endpoints. For this
1140 lab build, Motorola Solutions assigned a client ID of "ssoclient_nist" for the instances of
1141 their apps on the test devices.
- 1142 ▪ **CLIENT AUTHENTICATION:** May be set to **NONE**, **CLIENT SECRET** (for HTTP basic
1143 authentication), or **CLIENT TLS CERTIFICATE**. For native mobile app clients, there is no
1144 way to protect a client secret or private key and provide it to all instances of the app
1145 with any guarantee of confidentiality, as a user might be able to reverse-engineer the
1146 app to obtain any secrets delivered with it, or to debug the app to capture any secrets
1147 delivered at run-time. Therefore, a value of **NONE** is acceptable for native mobile apps,
1148 when mitigated with the use of PKCE. For web clients, servers are capable of protecting
1149 secrets; therefore, some form of client authentication should be required.
- 1150 ▪ **REDIRECT URIS:** Redirection URIs are the URIs to which the OAuth AS may redirect the
1151 resource owner's user agent after authorization is obtained. A redirect URI is used with
1152 the **Authorization Code** and **Implicit** grant types. This value is typically provided by the
1153 app developer to the AS administrator.
- 1154 ▪ **ALLOWED GRANT TYPES:** These are the allowed grant types for the client. For this lab
1155 build, the **Authorization Code** grant type was used exclusively.
- 1156 ▪ **DEFAULT ACCESS TOKEN MANAGER:** This is the Access Token Manager profile to be
1157 used for this client.
- 1158 ▪ **PERSISTENT GRANTS EXPIRATION:** This setting offers the option to override the global
1159 AS persistent grants settings for this client.
- 1160 ▪ **REFRESH TOKEN ROLLING POLICY:** This setting offers the option to override the global
1161 AS token rolling policy settings for this client.

1162 Once these values are set, click **Save** to store the client.

1163 This completes the required configuration for the AS's interactions with OAuth clients. The following
1164 section outlines the steps to set up the AS to authenticate users.

1165 3.4 How to Configure the OAuth 2 AS for Authentication

1166 In this section, the AS is configured to authenticate users locally or through federation with a SAML or
1167 OIDC IdP. These settings depend on the selection of roles and protocols, as shown in [Figure 3-2](#),
1168 therefore, ensure that has been completed before proceeding.

3.4.1 How to Configure Direct Authentication

The AS was configured to authenticate users with FIDO UAF authentication. This depends on the NNAS, Nok Nok Labs Gateway, and Nok Nok Labs UAF Plugin for PingFederate. See [Section 5](#) for the installation and configuration instructions for those components. This section assumes that those components have already been installed and configured.

3.4.1.1 Configure Adapter Instance

1. First, an instance of the FIDO UAF adapter must be configured. Click the **IdP Configuration** section tab, and then click **Adapters** under **Application Integration**.
2. Click **Create New Instance** to create an IdP adapter instance. This will bring up the new tabbed screen shown in [Figure 3-11](#).
 - a. On the **Type** tab, the **INSTANCE NAME** and **INSTANCE ID** are internal identifiers and can be set to any meaningful values. The **TYPE** selection, “FIDO Adapter,” will not appear until the Nok Nok Labs UAF plugin has been successfully installed on the PingFederate server as described in [Section 5](#).

1183 Figure 3-11 Create Adapter Instance

MAIN

IdP Configuration

SP Configuration

OAuth Settings

Server Configuration

Manage IdP Adapter Instances | Create Adapter Instance

Type	IdP Adapter	Extended Contract	Adapter Attributes	Adapter Contract Mapping	Summary
Enter an Adapter Instance Name and Id, select the Adapter Type, and a parent if applicable. The Adapter Type is limited to the adapters currently installed on your server.					
INSTANCE NAME	FIDO UAF				
INSTANCE ID	fidoUaf				
TYPE	FIDO Adapter Visit Pingidentity.com for additional types				
PARENT INSTANCE	None				

Copyright © 2003-2016
Ping Identity Corporation
All rights reserved
Version 8.2.2.0

Cancel Next

1184

- 1185 b. On the **IdP Adapter** tab, specify the URLs for the Nok Nok Labs API and Gateway end-
1186 points (Figure 3-12).
- 1187 i. The **NNL SERVER POLICY NAME** field can be used to select a custom policy, if
1188 one has been defined on the Nok Nok Labs server; for this build, the default pol-
1189 icy was used.

1190 Figure 3-12 FIDO Adapter Settings

Field Name	Field Value	Description
NNL SERVER AUTHENTICATION API ENDPOINT	<input type="text" value="https://mfas-nccoe.noknoktest.com:844"/>	Enter NNL Server Authentication Endpoint
NNL GATEWAY API ENDPOINT	<input type="text" value="https://mfas-nccoe.noknoktest.com:844"/>	Enter NNL Gateway Endpoint
NNL SERVER POLICY NAME	<input type="text" value="default"/>	Enter Policy Name Configured on NNL Server
TENANT IDENTIFIER	<input type="text" value="default"/>	Enter Tenant Identifier
LOGIN PAGE RENDERING OPTION	<input checked="" type="radio"/> Embedded Frame <input type="radio"/> Render Login Web Page	Specify your rendering option

1191

- 1192 c. The **Extended Contract** tab was also left as the default for the adapter, which provides
 1193 the **riskscore**, **transactionid**, **transactiontext**, and **username** values (Figure 3-13). If de-
 1194 sired, additional attributes could be added to the contract and looked up in a user direc-
 1195 tory, based on the username returned from the adapter.

1196 Figure 3-13 FIDO Adapter Contract

MAIN

PingFederate

Manage IdP Adapter Instances | Create Adapter Instance

Type	IdP Adapter	Extended Contract	Adapter Attributes	Adapter Contract Mapping	Summary
This adapter type supports the creation of an Extended Adapter Contract after initial deployment of the adapter instance. This Adapter Contract may be used to fulfill the Attribute Contract, look up additional attributes from a local data store, or create a persistent name identifier which uniquely identifies the user passed to your SP partners.					
Core Contract					
riskscore					
transactionid					
transactiontext					
username					
Extend the Contract		Action			
		Add			

Copyright © 2003-2016
Ping Identity Corporation
All rights reserved
Version 8.2.2.0

1197

- 1198 d. On the **Adapter Attributes** tab, select the **Pseudonym** checkbox for **username**. Pseudonyms were not used in the lab build, but a selection is required on this tab.
- 1199
- 1200 e. There is no need to configure an adapter contract, unless attributes have been added on the **Extended Contract** tab. Clicking **Done** and then **Save** completes the configuration of the adapter. Clicking the adapter name in the list of adapters brings up the Adapter Instance **Summary** tab, which lists all of the configured settings (Figure 3-14).
- 1201
- 1202
- 1203

1204 **Figure 3-14 FIDO Adapter Instance Summary**

PingFederate

MAIN

IdP Configuration

SP Configuration

OAuth Settings

Server Configuration

Manage IdP Adapter Instances | Create Adapter Instance

Type IdP Adapter Extended Contract Adapter Attributes Adapter Contract Mapping Summary

IdP adapter instance summary information.

Create Adapter Instance

Type

Instance Name	fidoonly
Instance Id	fidoonly
Type	FIDO Adapter
Class Name	com.noknok.adapter.ping.FidoAdapter
Parent Instance Name	None

IdP Adapter

NNI Server Authentication API Endpoint	https://noknok.sandbox.motorolasolutions.com:8443/nni/v2/auth
NNI Gateway API Endpoint	https://noknok.sandbox.motorolasolutions.com:8443/ngateway/nni
NNI Server Policy Name	default
Tenant Identifier	default

Extended Contract

Attribute	riskscore
Attribute	transactiontext
Attribute	transactionid
Attribute	username

Adapter Attributes

Mask all OGNL expression log values	false
Pseudonym	username

Adapter Contract Mapping

Attribute Sources & User Lookup

Data Sources	(None)
--------------	--------

Adapter Contract Fulfillment

riskscore	riskscore (Adapter)
transactiontext	transactiontext (Adapter)
transactionid	transactionid (Adapter)
username	username (Adapter)

Issuance Criteria

Criterion	(None)
-----------	--------

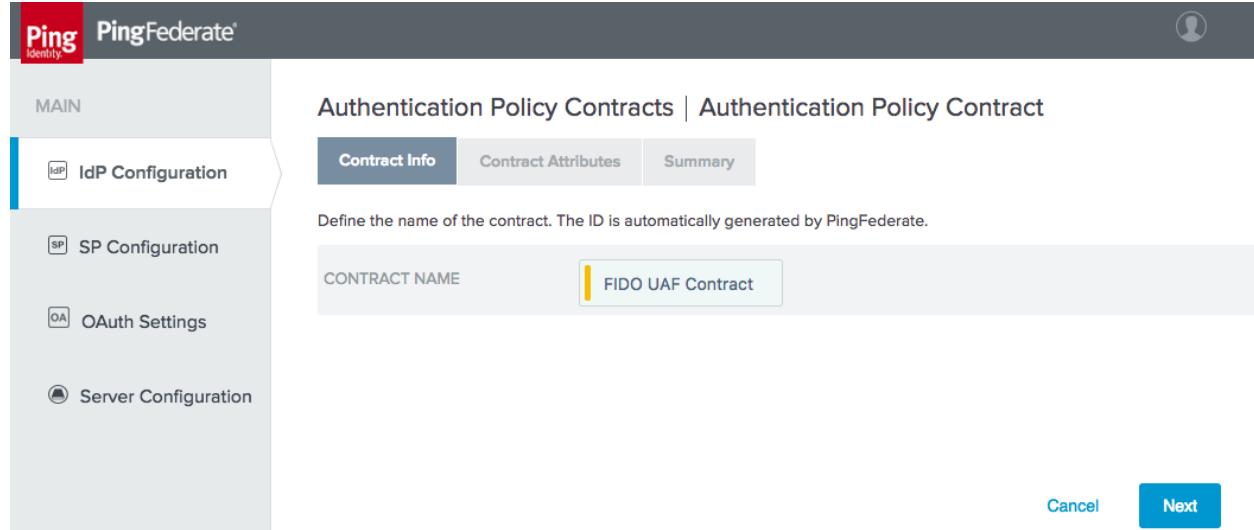
Cancel Previous

1205

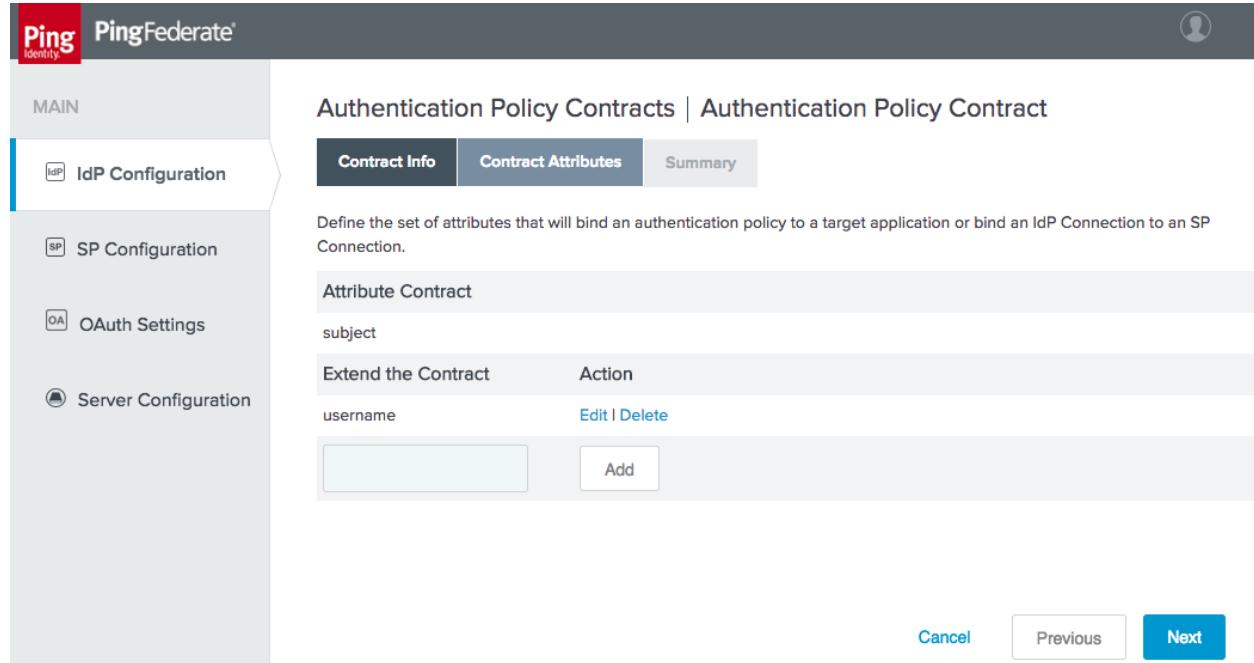
1206 Some additional configurations are needed to tie this authentication adapter to the issuance of an
 1207 OAuth token. It is possible to directly map the adapter to the access token context, but because the
 1208 adapter will be incorporated into an authentication policy in this case, an Authentication Policy Contract
 1209 Mapping is used instead.

1210 *3.4.1.2 Create Policy Contract*

- 1211 1. To create a Policy Contract, navigate to the **IdP Configuration** section tab, and select **Policy Contracts** under **Authentication Policies**. A policy contract defines the set of attributes that will be provided by an authentication policy.
- 1214 2. Click **Create New Contract**.
 - a. On the **Contract Info** tab, give the contract a meaningful name (Figure 3-15).

1216 **Figure 3-15 Policy Contract Information**

1217

1218 b. On the **Contract Attributes** tab, add a value called **username** (Figure 3-16).1219 **Figure 3-16 Policy Contract Attributes**

1220

1221 c. Click **Done**, and then click **Save** to save the new contract.

1222 *3.4.1.3 Create Policy Contract Mapping*

- 1223 1. Create a mapping from the policy contract to the OAuth persistent grant. Click the **OAuth**
 1224 **Settings** section tab, and then click **Authentication Policy Contract Mapping** under **Token &**
 1225 **Attribute Mapping**.
- 1226 a. Select the newly-created policy contract, and then click **Add Mapping** (Figure 3-17).

1227 **Figure 3-17 Create Authentication Policy Contract Mapping**

- 1228 2. An attribute source could be added at this point to look up additional user attributes, but this is
 1229 not necessary. Click **Save**.
- 1230 3. Skip the **Attribute Sources & User Lookup** tab.
- 1231 4. On the **Contract Fulfillment** tab, map both **USER_KEY** and **USER_NAME** to the **subject** value
 1232 returned from the policy contract (Figure 3-18).

1234 **Figure 3-18 Authentication Policy Contract Fulfillment**

MAIN

- IdP IdP Configuration
- SP SP Configuration
- OA OAuth Settings
- Server Server Configuration

Copyright © 2003-2016
Ping Identity Corporation
All rights reserved
Version 8.2.1.1

Authentication Policy Contract Mappings | Policy Contract Mapping

Contract	Source	Value	Actions
USER_KEY	Authentication Policy Contract	subject	None available
USER_NAME	Authentication Policy Contract	subject	None available

Cancel Previous Next

1235

1236 5. No issuance criteria were specified. Click **Next**, and then click **Save** to complete the mapping.1237 *3.4.1.4 Create Access Token Mapping*1238 Finally, an access token mapping needs to be created. In this simple case, the adapter only provides a
1239 single attribute (username) and it is stored in the persistent grant, so a default attribute mapping can be
1240 used.1241 1. On the **OAuth Settings** section tab, under **Token & Attribute Mapping**, click **Access Token**
1242 **Mapping**.1243 a. Select **Default** for the **CONTEXT** (Figure 3-19).1244 b. Select the **ACCESS TOKEN MANAGER** created previously (Figure 3-19).

1245 Figure 3-19 Create Access Token Attribute Mapping

MAIN

- IdP IdP Configuration
- SP SP Configuration
- OA OAuth Settings
- Server Server Configuration

Access Token Attribute Mapping

Manage the attribute mapping(s) to fulfill the access token attribute contract. This configuration maps from the user attributes stored with the persistent grant into the access token attribute contract. A default mapping should be configured for each access token manager. The default can be overridden based on the context of the authentication event of the original grant (IdP Adapter, an IdP Connection, a Credentials Validator, or an Authentication Policy).

Context	Token Manager	Action
Authentication Policy Contract: op1 sp contract	JWT Token	Delete
Default	Minimal Token	Delete
IdP Adapter: FIDO UAF	Minimal Token	Delete
IdP Connection: OP1 Connection	JWT Token	Delete

CONTEXT: Default

ACCESS TOKEN MANAGER: fidoJwt

Add Mapping

Cancel Save

1246

- 1247 c. Click **Add Mapping**.
- 1248 d. Click **Next** to Skip the **Attribute Sources & User Lookup** tab.
- 1249 e. On the **Contract Fulfillment** tab, configure sources and values for the **realm** and **sub** contracts (Figure 3-20). In this case, **realm** is set to the text string **motorolasolutions.com**. Click **Next**.
- 1250
- 1251

1252 Figure 3-20 Access Token Mapping Contract Fulfillment

MAIN

- IdP IdP Configuration
- SP SP Configuration
- OA OAuth Settings
- Server Server Configuration

Access Token Attribute Mapping | Access Token Mapping

Select a Source and Value to map into each item in the Contract list.

Contract	Source	Value	Actions
realm	Text	motorolasolutions.com	None available
sub	Persistent Grant	USER_KEY	None available

Attribute Sources & User Lookup Contract Fulfillment Issuance Criteria Summary

Cancel Previous Next

1253

- 1254 f. Click **Next** through the **Issuance Criteria** tab, and then click **Save**.
- 1255 2. To complete the setup for direct authentication, the FIDO UAF adapter needs to be included
1256 in an authentication policy as described in Section 3.4.4.2.

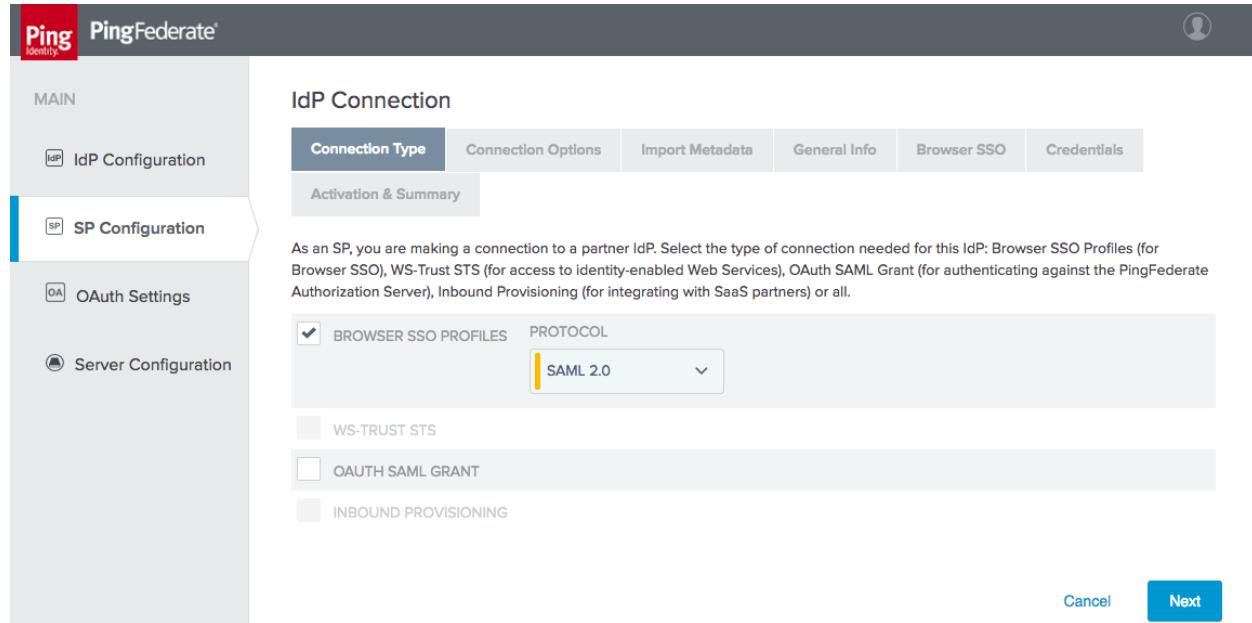
1257 3.4.2 How to Configure SAML Authentication

1258 This section explains how to configure the AS to accept SAML authentication assertions from a SAML 2.0
1259 IdP. This configuration is for RP-initiated SAML web browser SSO, where the authentication flow begins
1260 at the AS and the user is redirected to the IdP. Here, it is assumed that all of the steps outlined in
1261 [Section 3.4](#) have been completed, particularly enabling the SP role and protocols.

1262 3.4.2.1 Create IdP Connection

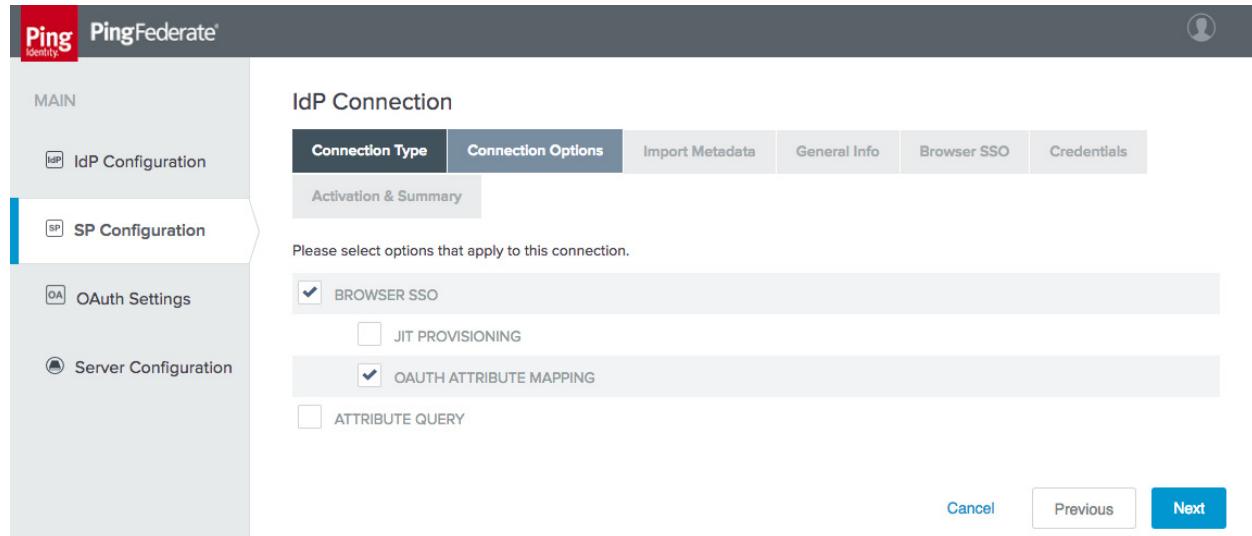
1263 Establishing the relationship between the AS and IdP requires coordination between the administrators
1264 of the two servers, which will typically belong to two separate organizations. The administrators of the
1265 SAML IdP and RP will need to exchange their **BASE URL** and **SAML 2.0 ENTITY ID** values (available on the
1266 **Federation Info** tab under **Server Settings**) to complete the configuration. The IdP administrator must
1267 also provide the signing certificate of the IdP. If assertions will be encrypted, the AS administrator will
1268 need to provide the IdP administrator with the certificate to be used for the public key. Alternatively,
1269 administrators can export their SAML metadata and provide it to the other party to automate parts of
1270 the setup.

- 1271 1. On the **SP Configuration** section tab, click **Create New** under **IdP Connections**.
- 1272 a. On the **Connection Type** tab, select **BROWSER SSO PROFILES**, and choose **SAML 2.0** for
1273 the **PROTOCOL** (Figure 3-21). If these options are not present, ensure that the roles are
1274 selected correctly in **Server Settings**.

1275 **Figure 3-21 Create IdP Connection**

1276

- 1277 b. On the **Connection Options** tab, select **BROWSER SSO**, and then under it, **OAUTH ATTRIBUTORY MAPPING** (Figure 3-22).

1279 **Figure 3-22 IdP Connection Options**

1280

- 1281 c. Metadata import was not configured for the lab build; therefore, skip the **Import**
1282 **Metadata** tab.

- 1283 d. On the **General Info** tab, enter the **PARTNER'S ENTITY ID (CONNECTION ID)** and **BASE URL**
 1284 of the IdP, and provide a **CONNECTION NAME** (Figure 3-23).

1285 **Figure 3-23 IdP Connection General Info**

The screenshot shows the 'IdP Connection' configuration interface. The left sidebar has sections for MAIN (IdP Configuration, SP Configuration, OAuth Settings, Server Configuration), and a copyright notice for Ping Identity Corporation. The top navigation bar includes tabs for Connection Type, Connection Options, Metadata URL, General Info (which is active), Browser SSO, and Credentials. The 'Activation & Summary' sub-tab is selected under General Info. The main content area contains the following fields:

- PARTNER'S ENTITY ID (CONNECTION ID): idp1.spsd.mss0
- CONNECTION NAME: idp1.spsd.mss0
- VIRTUAL SERVER IDS: (empty input field) Add
- BASE URL: https://idp1.spsd.mss0:9031
- COMPANY: (empty input field)
- CONTACT NAME: (empty input field)
- CONTACT NUMBER: (empty input field)
- CONTACT EMAIL: (empty input field)
- ERROR MESSAGE: (empty input field)
- LOGGING MODE: STANDARD (selected radio button)

At the bottom right are buttons for Cancel, Previous, and Next.

- 1286
 1287 e. On the **Browser SSO** tab, click **Configure Browser SSO**. The Browser SSO setup has multiple sub-pages.
 1288
 1289 i. On the **SAML Profiles** tab, select **SP-Initiated SSO**. The **User-Session Creation**
 1290 settings are summarized on the **Summary** tab; they extract the user ID and
 1291 email address from the SAML assertion (Figure 3-24).

1292 Figure 3-24 IdP Connection – User-Session Creation

The screenshot shows the 'User-Session Creation' configuration page in the PingFederate interface. The left sidebar has sections for MAIN (IdP Configuration, SP Configuration, OAuth Settings, Server Configuration), and a copyright notice for Ping Identity Corporation. The main content area has tabs for Identity Mapping, Attribute Contract, Target Session Mapping, and Summary. The Attribute Contract tab is active, showing mappings for SAML_SUBJECT, mail, and uid. The Target Session Mapping tab shows adapter instance and authentication policy contract names. The Adapter Instance tab shows a selected adapter. The Adapter Data Store tab indicates using only attributes available in the SSO Assertion. The Adapter Contract Fulfillment tab shows uid, mail, and subject mappings. The Issuance Criteria tab shows no criterion. The Authentication Policy Contract tab shows a selected contract. The Attribute Retrieval tab indicates using only attributes available in the SSO Assertion. The Contract Fulfillment tab shows uid, mail, and subject mappings. The Issuance Criteria tab shows no criterion. Navigation buttons at the bottom right include Cancel and Previous.

Attribute Contract	
Attribute	SAML_SUBJECT
Attribute	mail
Attribute	uid

Target Session Mapping	
Adapter instance name	instanceAdapterName
Authentication policy contract name	myContractName

Adapter Instance	
Selected adapter	instanceAdapterName

Adapter Data Store	
Attribute location	Use only the attributes available in the SSO Assertion

Adapter Contract Fulfillment	
uid	uid (Assertion)
mail	mail (Assertion)
subject	SAML_SUBJECT (Assertion)

Issuance Criteria	
Criterion	(None)

Authentication Policy Contract	
Selected contract	myContractName

Attribute Retrieval	
Attribute location	Use only the attributes available in the SSO Assertion

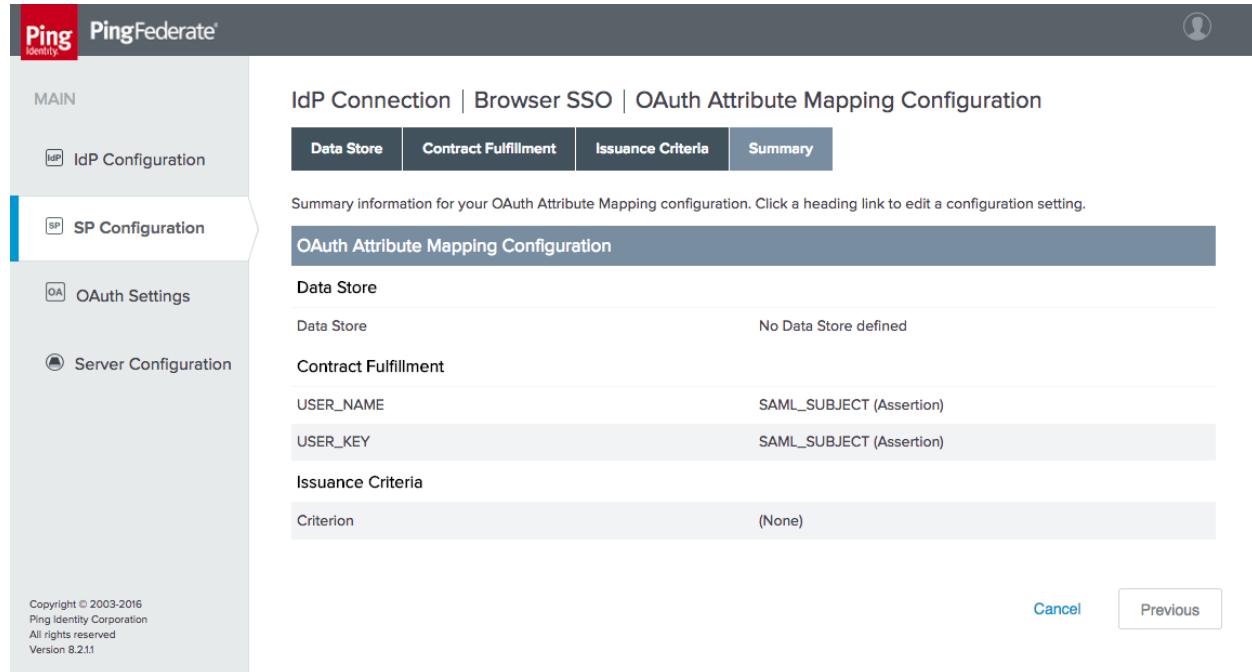
Contract Fulfillment	
uid	uid (Assertion)
mail	mail (Assertion)
subject	SAML_SUBJECT (Assertion)

Issuance Criteria	
Criterion	(None)

1293

- 1294 ii. On the **OAuth Attribute Mapping Configuration** tab, select **MAP DIRECTLY INTO**
 1295 **PERSISTENT GRANT**. Configure the OAuth attribute mapping as shown in Figure
 1296 3-25. This maps both required values in the persistent grant context to the
 1297 SAML subject. Click **Next**, then **Next** again to skip the **Issuance Criteria** tab. Click
 1298 **Save**.

1299 Figure 3-25 IdP Connection OAuth Attribute Mapping



The screenshot shows the PingFederate configuration interface. The left sidebar has a 'MAIN' section with 'IdP Configuration', 'SP Configuration' (which is selected), 'OAuth Settings', and 'Server Configuration'. The main content area is titled 'IdP Connection | Browser SSO | OAuth Attribute Mapping Configuration'. It shows tabs for 'Data Store', 'Contract Fulfillment', 'Issuance Criteria', and 'Summary'. The 'Contract Fulfillment' tab is active, displaying configuration for 'USER_NAME' and 'USER_KEY'. Both are mapped to 'SAML_SUBJECT (Assertion)'. The 'Issuance Criteria' section shows '(None)' under 'Criterion'. At the bottom right are 'Cancel' and 'Previous' buttons.

- 1300 iii. Click **Next** to proceed to the **Protocol Settings** tab. The **Protocol Settings** config-
 1301 ure specifics of the SAML protocol, such as the allowed bindings. Configure
 1302 these as shown in Figure 3-26. When finished, click **Save**, which will return you
 1303 to the **Browser SSO** tab of the **IdP Connection** settings.
 1304

1305 **Figure 3-26 IdP Connection – Protocol Settings**

Protocol Settings

SSO Service URLs	
Endpoint	URL: /idp/SSO.saml2 (POST)
Endpoint	URL: /idp/SSO.saml2 (Redirect)

Allowable SAML Bindings	
Artifact	false
POST	true
Redirect	true
SOAP	false

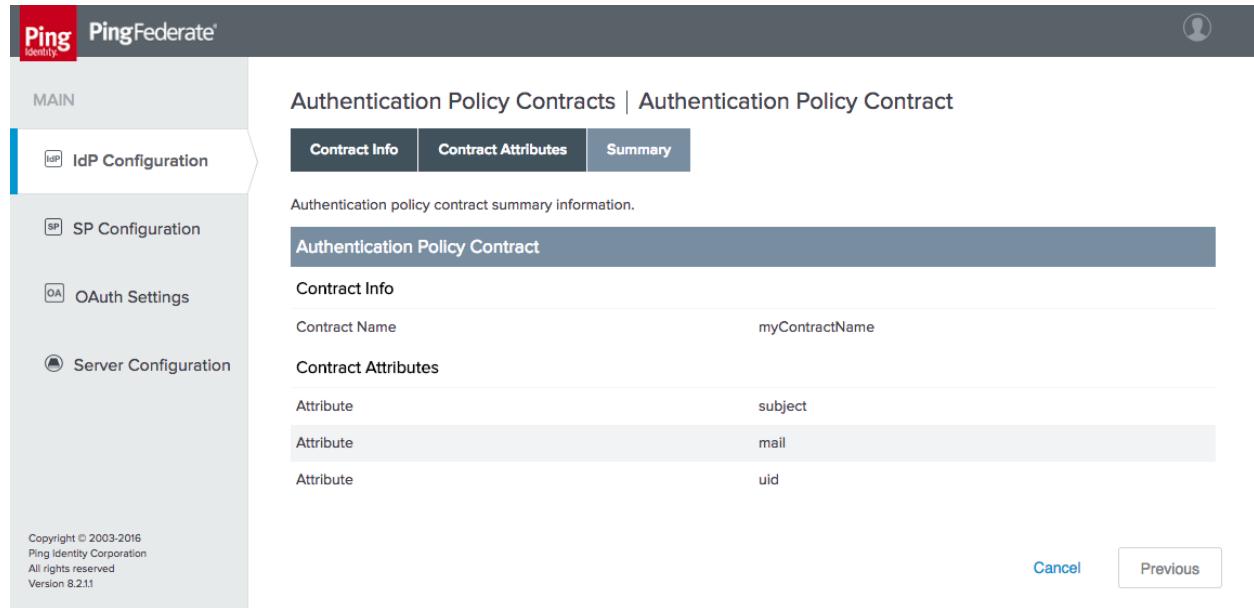
Overrides	
Signature Policy	
Sign AuthN requests over POST and Redirect	false
Require digitally signed SAML Assertion	false
Encryption Policy	
Status	Inactive

1306

- 1307 f. Click **Next**. On the **Credentials** tab, the IdP's signing certificate can be uploaded. This is
1308 not necessary if the certificate is signed by a trusted CA.

1309 *3.4.2.2 Create Policy Contract*

- 1310 1. Create a policy contract as described in [Section 3.4.1.2](#), with the attributes **subject**, **mail**, and **uid**
 1311 (Figure 3-27).

1312 **Figure 3-27 Policy Contract for SAML RP**


The screenshot shows the PingFederate configuration interface. The left sidebar has a 'MAIN' section with 'IdP Configuration' selected, which is further divided into 'SP Configuration', 'OAuth Settings', and 'Server Configuration'. The main content area is titled 'Authentication Policy Contracts | Authentication Policy Contract'. It displays a summary of the 'Authentication policy contract summary information'. Below this, a table lists the 'Contract Info' and 'Contract Attributes' for the 'Authentication Policy Contract'. The 'Contract Info' section shows 'Contract Name' as 'myContractName'. The 'Contract Attributes' section maps three attributes to the 'subject' claim: 'Attribute' 'mail' and 'Attribute' 'uid'. At the bottom right, there are 'Cancel' and 'Previous' buttons.

Contract Info	
Contract Name	myContractName

Contract Attributes	
Attribute	subject
Attribute	mail
Attribute	uid

1313

1314 *3.4.2.3 Create Policy Contract Mapping*

- 1315 1. Create an OAuth policy contract mapping for the newly created policy as described in
 1316 [Section 3.4.1.3](#), mapping **USER_NAME** and **USER_KEY** to **subject** (Figure 3-28).

1317 **Figure 3-28 Contract Mapping for SAML RP**

1318

- 1319 2. To complete the setup for SAML authentication, the FIDO UAF adapter needs to be included in
 1320 an authentication policy as described in [Section 3.4.4.2](#).

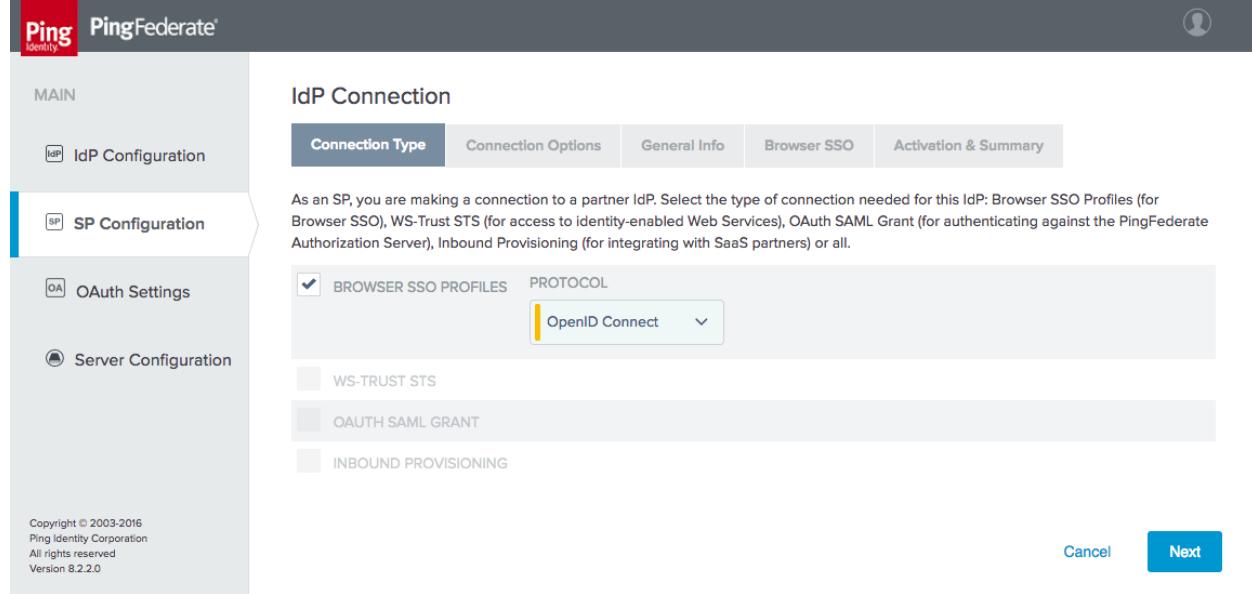
1321 **3.4.3 How to Configure OIDC Authentication**

1322 As with the configuration of a SAML IdP connection, integrating the AS with an OIDC IdP requires
 1323 coordination between the administrators of the two systems. The administrator of the IdP must create
 1324 an OIDC client registration before the connection can be configured on the AS side. The AS administrator
 1325 must provide the redirect URI and, if encryption of the ID Token is desired, a public key. Unlike with
 1326 SAML, there is no metadata file to exchange; however, if the IdP supports the OIDC discovery endpoint,
 1327 the client can automatically obtain many of the required configuration settings from the discovery URL.

1328 This section assumes that the AS role and OIDC SP support have been enabled via **Server Settings**, as
 1329 described in [Section 3.4](#). This section also uses the same authentication policy contract as the SAML
 1330 authentication implementation. Create the policy contract as described in [Section 3.4.2.2](#), if it does not
 1331 already exist.

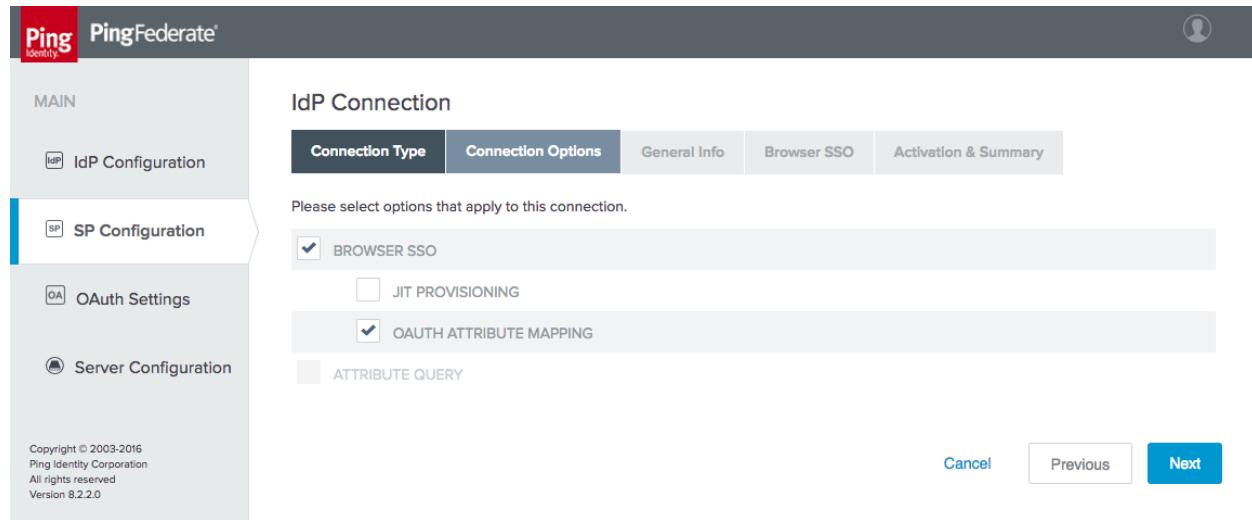
1332 ***3.4.3.1 Create IdP Connection***

- 1333 1. On the **SP Configuration** section tab, click **Create New** under **IdP Connections**.
 1334 a. On the **Connection Type** tab, select **BROWSER SSO PROFILES**, and then under it, select
 1335 **OpenID Connect** for the **PROTOCOL** (Figure 3-29).

1336 **Figure 3-29 IdP Connection Type**

1337

- 1338 b. On the **Connection Options** tab, select **BROWSER SSO**, and then under it, select **OAUTH ATTRIBUTE MAPPING** (Figure 3-30).

1340 **Figure 3-30 IdP Connection Options**

1341

- 1342 c. On the **General Info** tab, enter the **ISSUER** value for the IdP (Figure 3-31). This is the
1343 **BASE URL** setting available on the **Federation Info** tab, under the **Server Configuration**
1344 section tab on the IdP. Then click **Load Metadata**, which causes the AS to query the IdP's

1345 discovery endpoint. The message “Metadata successfully loaded” should appear. Pro-
 1346 vide a **CONNECTION NAME**, and enter the **CLIENT ID** and **CLIENT SECRET** provided by
 1347 the IdP administrator.

1348 **Figure 3-31 IdP Connection General Info**

The screenshot shows the 'IdP Connection' configuration page in the PingFederate interface. The left sidebar has sections for MAIN (IdP Configuration, SP Configuration, OAuth Settings, Server Configuration), and a copyright notice for Ping Identity Corporation. The main content area is titled 'IdP Connection' and includes tabs for Connection Type, Connection Options, General Info (selected), Browser SSO, and Activation & Summary. The 'General Info' tab contains fields for ISSUER (https://op1.lpsd.mss0:9031), CONNECTION NAME (op1.lpsd.mss0), CLIENT ID (MotorolaAS), and CLIENT SECRET (redacted). A note says 'Metadata successfully loaded.' Below these are fields for BASE URL, COMPANY, CONTACT NAME, CONTACT NUMBER, and CONTACT EMAIL, all of which are empty. An 'ERROR MESSAGE:' field contains 'errorDetail.spSsoFailure'. At the bottom, a 'LOGGING MODE' section has radio buttons for NONE, STANDARD (selected), ENHANCED, and FULL. Navigation buttons at the bottom right are 'Cancel', 'Previous', 'Next', and a blue 'Save' button.

- 1349
- 1350 d. On the **Browser SSO** tab, click **Configure Browser SSO**, then click **Configure User-Ses-**
 1351 **sion Creation**. The **User-Session Creation** page will appear.
- 1352 i. On the **Target Session Mapping** tab, click **Map New Authentication Policy**.

- 1353 ii. On the **Authentication Policy Contract** tab, select the **AUTHENTICATION POLICY**
 1354 **CONTRACT** created in [Section 3.4.2.2](#) (in the example shown in Figure 3-32, it is
 1355 called **myContractName**). If the policy contract has not been created, click **Man-**
 1356 **age Authentication Policy Contracts**, and create it now.

1357 Figure 3-32 IdP Connection Authentication Policy Contract

The screenshot shows the PingFederate interface for managing IdP connections. The left sidebar has tabs for MAIN, IdP Configuration, SP Configuration (which is selected), OAuth Settings, and Server Configuration. The main content area has tabs for IdP Connection, Browser SSO, User-Session Creation, and Authentication Policy Mapping. The Authentication Policy Contract tab is active. A sub-section titled 'Select the authentication policy contract you would like to activate for incoming provider claims from this partner.' contains a dropdown menu set to 'myContractName'. Below this are sections for 'Contract Attributes' with fields for 'mail', 'subject', and 'uid'. At the bottom are buttons for 'Manage Authentication Policy Contracts', 'Cancel', 'Save Draft', and 'Next'.

- 1358 iii. On the **Attribute Retrieval** tab, leave the default setting (use only the attributes available in the provider claims).
 1359 iv. On the **Contract Fulfillment** tab, map the **mail**, **subject**, and **uid** attributes to the **email**, **sub**, and **sub** provider claims (Figure 3-33).

1363 Figure 3-33 IdP Connection Policy Contract Mapping

MAIN

IdP Configuration

SP Configuration

OAuth Settings

Server Configuration

Copyright © 2003-2016
Ping Identity Corporation
All rights reserved
Version 8.2.11

IdP Connection | Browser SSO | User-Session Creation | Authentication Policy Mapping

Authentication Policy Contract Attribute Retrieval Contract Fulfillment Issuance Criteria Summary

You can fulfill your Authentication Policy Contract with values from the provider claims, dynamic text, expressions, or from a data-store lookup.

Authentication Policy Contract	Source	Value	Actions
mail	Provider Claims	email	None available
subject	Provider Claims	sub	None available
uid	Provider Claims	sub	None available

Cancel Previous Next

1364

- 1365 v. No **Issuance Criteria** were configured; therefore, skip the **Issuance Criteria** tab.
- 1366 vi. Click **Next**, then **Done**, and then click **Done** again to exit the **User-Session Creation** tab.
- 1367
- 1368 vii. On the **OAuth Attribute Mapping Configuration** tab, select **Map Directly into Persistent Grant**, and then click **Configure OAuth Attribute Mapping**.
- 1369
- 1370 viii. Click **Next** to skip the Data Store tab. On the **Contract Fulfillment** tab, map both **USER_NAME** and **USER_KEY** to the **sub** provider claim (Figure 3-34).
- 1371

1372 Figure 3-34 IdP Connection OAuth Attribute Mapping

The screenshot shows the PingFederate interface for managing OAuth attribute mappings. The left sidebar has sections for MAIN (IdP Configuration, SP Configuration, OAuth Settings, Server Configuration), and a footer with copyright information. The main area is titled 'IdP Connection | Browser SSO | OAuth Attribute Mapping Configuration'. It includes tabs for Data Store, Contract Fulfillment, Issuance Criteria, and Summary. A sub-section titled 'Manage attribute mappings into the persistent grant contract and the USER_NAME value displayed to end users.' is shown. It lists two mappings:

Contract	Source	Value	Actions
USER_KEY	Provider Claims	sub	None available
USER_NAME	Provider Claims	sub	None available

At the bottom are buttons for Cancel, Save Draft, Previous, Next, and Done.

1373

- 1374 ix. Click **Done** to exit the **OAuth Attribute Mapping Configuration** setup. The **Protocol Settings** should be automatically populated through the information gathered from the discovery endpoint (Figure 3-35). If necessary, the scopes to be requested can be customized on the **Protocol Settings** tab; in the lab, these settings were left at the default.
- 1375
- 1376
- 1377
- 1378

1379 **Figure 3-35 IdP Connection Protocol Settings**

Protocol Settings

OpenID Provider Info	Overrides	Summary
Scopes	oob-reg address test phone reg composite openid profile name email	
Authorization Endpoint	https://op1.lpsd.msso:9031/as/authorization.oauth2	
Authentication Scheme	Post	
Token Endpoint	https://op1.lpsd.msso:9031/as/token.oauth2	
Userinfo Endpoint	https://op1.lpsd.msso:9031/idp/userinfo.openid	
JWKS URL	https://op1.lpsd.msso:9031/pf/JWKS	

Overrides

Copyright © 2003-2016
Ping Identity Corporation
All rights reserved
Version 8.2.2.0

Cancel Save Draft Previous Done

1380

- 1381 x. Click **Done** to exit the **Browser SSO** configuration setup.
- 1382 e. On the **Activation & Summary** tab, a **Redirect URI** will be generated (Figure 3-36). Provide this information to the IdP administrator, as it needs to be configured in the OpenID Client settings on the IdP side.
 - 1383 i. The **Connection Status** can also be configured to **ACTIVE** or **INACTIVE** on this tab.

1387 **Figure 3-36 IdP Connection Activation and Summary**

The screenshot shows the PingFederate interface with the following details:

- Left Sidebar (MAIN):**
 - IdP Configuration** (selected)
 - SP Configuration** (highlighted in blue)
 - OAuth Settings**
 - Server Configuration**
- Top Bar:** PingFederate
- Header:** IdP Connection
- Tab Bar:** Connection Type, Connection Options, General Info, Browser SSO, Activation & Summary (selected)
- Content Area:**
 - Summary information for your IdP connection. Click a heading in a section to edit a particular configuration setting.
 - Connection Status:** ACTIVE (radio button selected)
 - Redirect URI:** https://idm.sandbox.motorolasolutions.com/sp/eyJpc3MiOiJodHRwczpcL1wvb3AxLmxwc2QubXNzbzo5MDMxIn0/cb.openid
- Section Headers:** Summary, IdP Connection
- Table Rows (Connection Type):**

Connection Role	IdP
Browser SSO Profiles	true
Protocol	OpenID Connect
WS-Trust STS	false
OAuth SAML Grant	false
Inbound Provisioning	false

1388

1389 f. Click **Save** to complete the **IdP Connection** setup.1390 *3.4.3.2 Create the Policy Contract Mapping*1391 The same policy contract mapping created earlier for the SAML integration can also be used for OIDC
1392 integration, as the attribute names are identical. If this policy contract mapping has not already been
1393 created, refer to [Section 3.4.2.3](#) to create it.1394 *3.4.4 How to Configure the Authentication Policy*1395 *3.4.4.1 Install the Domain Selector Plugin*1396 When a single AS is integrated with multiple IdPs, it needs a means of determining which IdP can
1397 authenticate each user. In the lab build, a domain selector is used to determine whether the AS should
1398 authenticate the user locally, redirect to the SAML IdP, or redirect to the OIDC IdP. The domain selector
1399 prompts the user to enter the user's email address or domain. The specified domain is used to select
1400 which branch of the authentication policy should be applied. Upon successful authentication, the
1401 domain selector sets a cookie in the browser to persist the domain selection to avoid prompting the
1402 user each time that the user authenticates.

1403 PingFederate includes sample code for a Domain Selector plugin. Before the Domain Selector can be
 1404 used in an authentication policy, it must be built. The source code for the selector is located under the
 1405 PingFederate directory, in the directory `sdk/plugin-src/authentication-selector-example`.

1406 1. Complete the following steps to build the selector:

1407 a. Edit the `build.local.properties` file in the PingFederate SDK directory to set the tar-
 1408 get plugin as follows:

```
1409 target-plugin.name=authentication-selector-example
```

1410 b. Run the following commands to build and install the plugin:

```
1411 $ ant clean-plugin
```

```
1412 $ ant jar-plugin
```

```
1413 $ ant deploy-plugin
```

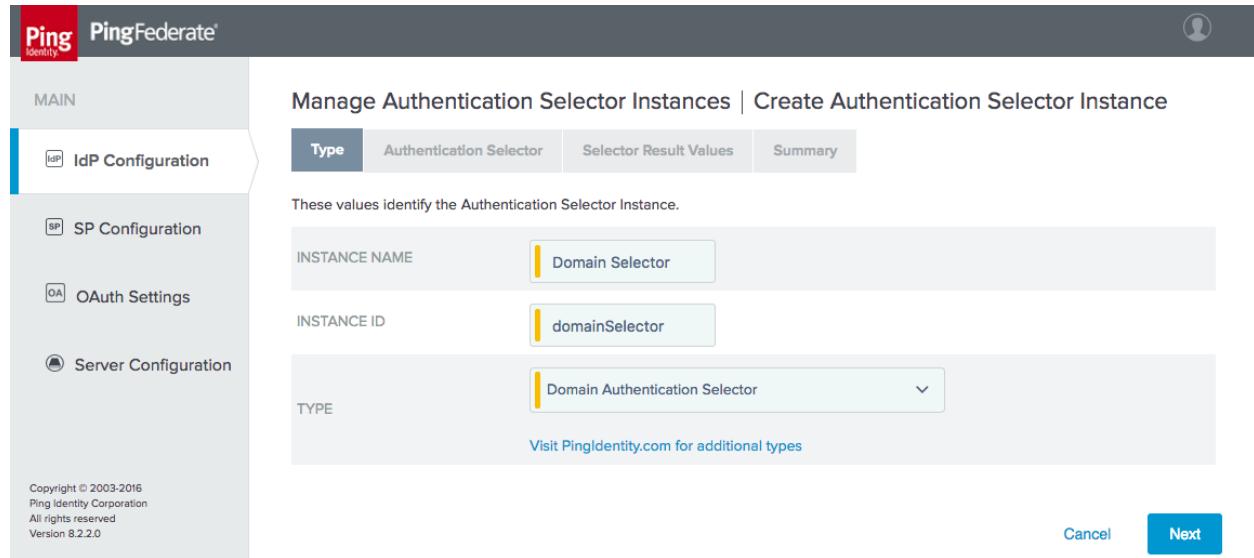
```
1414 $ sudo service pingfederate restart
```

1415 2. Once installed, the Domain Selector can be configured with the required values. On the **IdP**
 1416 **Configuration** section tab, click **Selectors** under **Authentication Policies**.

1417 3. Click **Create New Instance**.

1418 a. On the **Type** tab, provide a meaningful name and ID for the selector instance (Figure
 1419 3-37). For the **TYPE**, select **Domain Authentication Selector**.

1420 **Figure 3-37 Authentication Selector Instance**



1421

- 1422 b. The next tab, **Authentication Selector**, prompts for the HyperText Markup Language
 1423 (HTML) template for the page that will prompt the user to enter the domain or email
 1424 address (Figure 3-38). The default value will use the template delivered with the
 1425 adapter; if desired, a custom template can be used instead to modify the appearance of
 1426 the page. Provide a cookie name, which will be used to persist the domain selection. Fi-
 1427 nally, the age of the cookie can be modified. By default, users will be prompted again to
 1428 enter their domain after 30 days.

1429 **Figure 3-38 Authentication Selector Details**

Field Name	Field Value	Description
EMAIL ADDRESS OR DOMAIN NAME TEMPLATE	sample.authn.selector.email.template.htm	HTML template (in <pf_home>/server/default/conf/template) to render when a user is expected to provide an email address or a domain name. If the a email address is provided, a domain will be extracted from the input email address. An attempt will be made to match the extracted domain with a Selector Result Value hence resulting in the mapped authentication source.
COOKIE NAME	userDomainSelectorValue	Name of the cookie which saves the domain name. Once the email address or domain name is provided, upon successful authentication (or login), a cookie will be saved with this name. If left blank, a default cookie name, prefixed with pf-authn-selector- will be generated.
COOKIE AGE	30	Number of days that the domain name is stored as a cookie in the browser. The cookie age is reset upon each successful login. The default value is 30.

- 1430
 1431 c. On the **Selector Result Values** tab, specify the expected domain values (Figure 3-39).
 1432 When the domain selector is used in an access policy, different policy branches will be
 1433 created for each of these values. In this case, if the domain is *motorolasolutions.com*,
 1434 the user will be authenticated locally; if it is *lpsd.mss0* or *spsd.mss0*, the user will be re-
 1435 directed to the corresponding IdP to authenticate.

1436 **Figure 3-39 Selector Result Values**

Result Values	Action
lpsd.msso	Edit Delete
motorolasolutions.com	Edit Delete
spsd.msso	Edit Delete

1437

1438 d. Click **Done**, and then click **Save** to complete the selector configuration.

3.4.4.2 Define the Authentication Policy

1440 1. On the IdP Configuration page, click **Policies** under **Authentication Policies**.

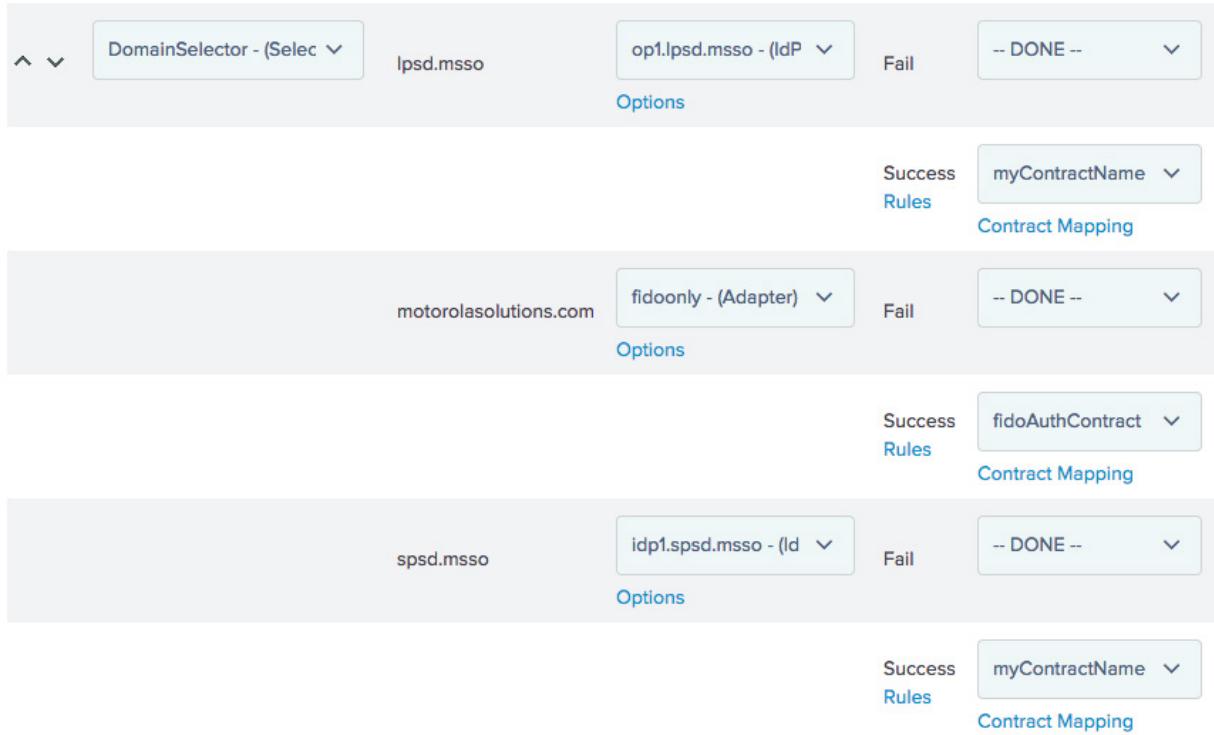
1441 a. Select the three checkboxes at the top of the **Manage Authentication Policies** page,
1442 which are shown in Figure 3-40.

1443 **Figure 3-40 Policy Settings**

1444

1445 b. Select the **Domain Selector** as the first element in the policy (Figure 3-41). This will cre-
1446 ate policy branches for the three values defined for the policy selector.

1447 i. Select the corresponding authentication mechanism for each domain. The ex-
1448 ample shown in Figure 3-41 uses the IdP connections for the **lpsd.msso** and
1449 **spsd.msso**, as well as the “fidoonly” adapter for local authentication of users in
1450 the **motorolasolutions.com** domain.

1451 **Figure 3-41 Authentication Policy**

1452

- 1453 ii. There is no need to specify **Options** or **Success Rules**. For the two IdP connec-
 1454 tions, apply the **myContractName** policy contract upon success, with the con-
 1455 tract mapping configured as shown in Figure 3-42.

1456 Figure 3-42 Policy Contract Mapping for IdP Connections

Manage Authentication Policies | Authentication Policy Contract Mapping

Attribute Sources & User Lookup Contract Fulfillment Issuance Criteria Summary

Summary of Authentication Policy Contract Mapping

Authentication Policy Contract Mapping

Attribute Sources & User Lookup

Data Sources	(None)
--------------	--------

Contract Fulfillment

uid	sub (IdP Connection)
mail	email (IdP Connection)
subject	sub (IdP Connection)

Issuance Criteria

Criterion	(None)
-----------	--------

Copyright © 2003-2016
Ping Identity Corporation
All rights reserved
Version 8.2.1

Cancel Previous

1457

- 1458 c. For the “fidoonly” adapter, apply the **fidoAuthContract** with the contract mapping shown in Figure 3-43.
- 1459

1460 **Figure 3-43 Policy Contract Mapping for Local Authentication**

The screenshot shows the PingFederate web interface. The top navigation bar includes the PingFederate logo and a user icon. The left sidebar has a 'MAIN' section with 'IdP Configuration', 'SP Configuration' (which is selected and highlighted in blue), 'OAuth Settings', and 'Server Configuration'. The main content area is titled 'Manage Authentication Policies | Authentication Policy Contract Mapping'. It features tabs for 'Attribute Sources & User Lookup', 'Contract Fulfillment', 'Issuance Criteria', and 'Summary'. The 'Contract Fulfillment' tab is active. Below it, there's a summary table:

Authentication Policy Contract Mapping	
Attribute Sources & User Lookup	(None)
Contract Fulfillment	subject: username (Adapter) username: username (Adapter)
Issuance Criteria	(None)

At the bottom right are 'Cancel' and 'Previous' buttons. A copyright notice at the bottom left states: 'Copyright © 2003-2016 Ping Identity Corporation. All rights reserved. Version 8.2.1.'

1461

1462 This completes the configuration of the AS.

1463

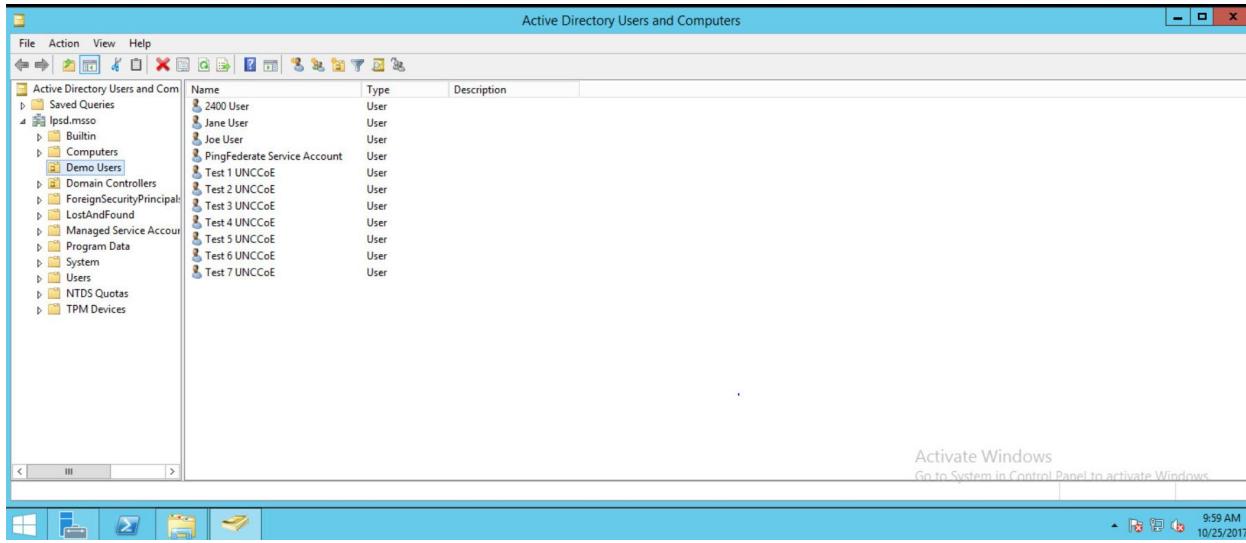
4 How to Install and Configure the Identity Providers

1464 PingFederate 8.3.2.0 was used for the SAML and OIDC IdP installs. The system requirements and
 1465 installation process for PingFederate are identical to the OAuth AS installation documentation in
 1466 [Section 3.1](#) and [Section 3.2](#). The IdP configuration sections pick up the installation process after the
 1467 software has been installed, at the selection of roles and protocols.

1468

4.1 How to Configure the User Store

1469 Each IdP uses its own AD forest as a user store. AD was chosen due to its widespread use across many
 1470 organizations. For the purposes of this project, any LDAP directory could have served the same purpose,
 1471 but in a typical organization, AD would be used for other functions, such as workstation login and
 1472 authorization to apps, shared drives, printers, and other services. The **Active Directory Users and**
 1473 **Computers** console (Figure 4-1) was used to create user accounts and set attributes.

1474 **Figure 4-1 Active Directory Users and Computers**

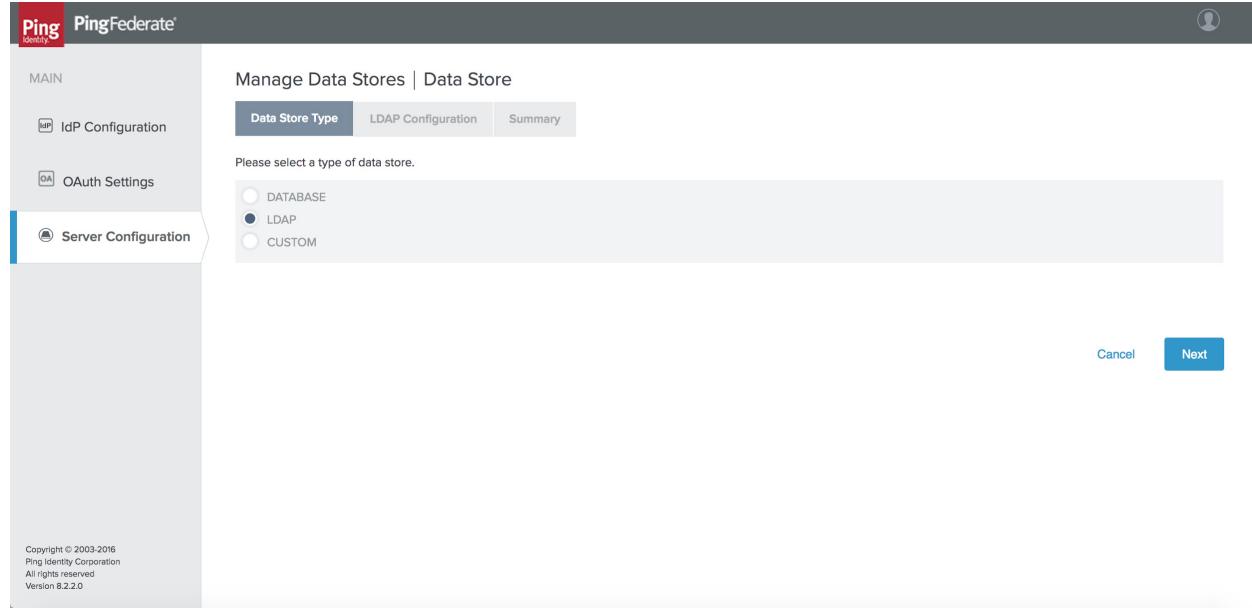
1475

- 1476 In addition to the user accounts that log into the lab apps, a service account must be created to enable
 1477 the IdP to access and query the AD. This user's LDAP Distinguished Name (DN) and password (in the
 1478 example shown in Figure 4-1) are used in the PingFederate directory integration described below.
 1479 The procedure for connecting a PingFederate IdP to an LDAP directory is the same for a SAML or OIDC
 1480 IdP. Documentation is provided at
 1481 https://documentation.pingidentity.com/pingfederate/pf82/index.shtml#concept_configuringLdapConnection.html#concept_configuringLdapConnection.
 1483 1. To start the process, click the **Server Configuration** section tab on the left side of the
 1484 PingFederate administrative console. The screen shown in Figure 4-2 will appear.

1485 **Figure 4-2 Server Configuration**

1486

- 1487 2. Click **Data Stores** under **SYSTEM SETTINGS**.
- 1488 3. On the next screen, click **Add New Data Store**.
- 1489 a. The screen shown in Figure 4-3 will appear. On the **Data Store Type** tab, select **LDAP** for
1490 the data store type.
- 1491 i. Click **Next**.

1492 **Figure 4-3 Data Store Type**

1493

- 1494 b. On the **LDAP Configuration** tab, enter the connection parameters for your AD or LDAP environment (Figure 4-4). Some notes on the fields on this tab are provided below. Click **Save** to exit the LDAP configuration screen once the required settings have been entered.
- 1498 ■ **HOSTNAME(S)**: Enter the Fully Qualified Domain Name (FQDN) or the complete Internet Protocol (IP) address of an AD domain controller. A port number can be specified if AD is running on non-standard ports.
- 1499 ■ **LDAP TYPE**: This is the LDAP server in use—AD in this case.
- 1500 ■ **BIND ANONYMOUSLY**: For AD environments, allowing anonymous BIND (Berkeley Internet Name Domain) is not recommended.
- 1501 ■ **USER DN**: This is the Distinguished Name of the PingFederate user account created in AD; in this build architecture, this account is used only for querying AD, so it does not require any special privileges.
- 1502 ■ **PASSWORD**: This is the password for the PingFederate AD user.
- 1503 ■ **USE LDAPS**: This can be enabled if AD is configured to serve LDAP over TLS.
- 1504 ■ **MASK VALUES IN LOG**: This prevents attributes returned from this data source from being exposed in server logs.

1511 **Figure 4-4 LDAP Data Store Configuration**

MAIN

PingFederate^{Identity}

Manage Data Stores | Data Store

Data Store Type LDAP Configuration Summary

Please provide the details for configuring this LDAP connection.

HOSTNAME(S)

LDAP TYPE

BIND ANONYMOUSLY

USER DN

PASSWORD

USE LDAPS

MASK VALUES IN LOG

Advanced

Copyright © 2003-2016
Ping Identity Corporation
All rights reserved
Version 8.2.2.0

Cancel Previous Next

1512

1513 **4.2 How to Install and Configure the SAML Identity Provider**1514 1. On the **Server Configuration** screen, click **Server Settings**.1515 a. On the **Roles & Protocols** tab, enable roles and protocols to configure the server as a
1516 SAML IdP (Figure 4-5).

1517 Figure 4-5 Server Roles for SAML IdP

The screenshot shows the 'Server Settings' page in the PingFederate interface. The left sidebar has 'MAIN' and 'Server Configuration' sections, with 'Server Configuration' being active. The main content area has tabs for 'System Administration', 'System Info', 'Runtime Notifications', 'Runtime Reporting', and 'Account Management', with 'Roles & Protocols' selected. Under 'Roles & Protocols', there are two main sections: 'ENABLE IDENTITY PROVIDER (IDP) ROLE AND SUPPORT THE FOLLOWING:' and 'ENABLE SERVICE PROVIDER (SP) ROLE AND SUPPORT THE FOLLOWING:'. Both sections include checkboxes for SAML 2.0 and other protocols like AUTO-CONNECT PROFILE, SAML 1.1, SAML 1.0, WS-FEDERATION, OUTBOUND PROVISIONING, WS-TRUST, X.509 ATTRIBUTE SHARING PROFILE (XASP), SAML 1.1, SAML 1.0, WS-FEDERATION, WS-TRUST, INBOUND PROVISIONING, OPENID CONNECT, and ENABLE IDP DISCOVERY ROLE (SAML 2.0 ONLY). At the bottom right are 'Cancel', 'Previous', 'Next', and a blue 'Save' button.

1518

- 1519 b. On the **Federation Info** tab, specify the **BASE URL** and **SAML 2.0 ENTITY ID** of the IdP
 1520 (Figure 4-6). The **BASE URL** should be a URL resolvable by your mobile clients. The **EN-
 1521 TITY ID** should be a meaningful name that is unique among federation partners; in this
 1522 case, the FQDN of the server is used.

1523 **Figure 4-6 SAML IdP Federation Info**

The screenshot shows the PingFederate interface with the title "PingFederate". The left sidebar has a "MAIN" section with "IdP Configuration", "SP Configuration", and "Server Configuration" (which is selected). The main content area is titled "Server Settings" and contains tabs for "System Administration", "System Info", "Runtime Notifications", "Runtime Reporting", and "Account Management". Under "System Administration", there are sub-tabs for "Roles & Protocols", "Federation Info" (which is selected), "System Options", "Metadata Signing", "Metadata Lifetime", and "Summary". A note states: "You must create a unique identifier for your server for use with your federation partners. A unique identifier is required for each protocol enabled. You will need to communicate this with your partners out-of-band or through metadata exchange. The Base URL is used to construct other URLs in the system and may be used as part of your system ID." Below this are fields for "BASE URL" (https://idp1.spsd.mss:9031) and "SAML 2.0 ENTITY ID" (idp1.spsd.mss). At the bottom are "Cancel", "Previous", "Next", and a blue "Save" button. Copyright information at the bottom left reads: "Copyright © 2003-2017 Ping Identity Corporation. All rights reserved. Version 8.3.2.0".

1524

4.2.1 Configuring Authentication to the IdP

1525 This example configures an authentication policy that requires the user to authenticate with username
1526 and password and then with a FIDO U2F token.

4.2.1.1 Configure the Password Validator

- 1529 1. On the **Server Configuration** section tab, click **Password Credential Validators** under
1530 **Authentication**.
- 1531 2. Click **Create New Instance**.
 - 1532 a. On the **Type** tab, for the **TYPE**, choose **LDAP Username Password Credential Validator**
1533 (**Figure 4-7**). This example will authenticate AD usernames and passwords by using the
1534 AD data store defined in [Section 4.1](#).

1535 Figure 4-7 Create Password Credential Validator

Manage Credential Validator Instances | Create Credential Validator Instance

Type Instance Configuration Extended Contract Summary

Identify this Credential Validator Instance. The Validator types available are limited to the plug-in implementations currently installed on your server.

INSTANCE NAME	Password Validator
INSTANCE ID	PasswordValidator
TYPE	LDAP Username Password Credential Validator
Visit PingIdentity.com for additional types	
PARENT INSTANCE	None

Copyright © 2003-2017
Ping Identity Corporation
All rights reserved
Version 8.3.2.0

Cancel Next

1536

- 1537 b. On the **Instance Configuration** tab, specify the parameters for searching the LDAP directory
 1538 for user accounts (Figure 4-8). Select the data store created in [Section 4.1](#), and enter
 1539 the appropriate search base and filter. This example will search for a *sAMAccountName*
 1540 matching the username entered on the login form.

1541 Figure 4-8 Credential Validator Configuration

Copyright © 2003-2017
Ping Identity Corporation
All rights reserved
Version 8.3.2.0

1542

- 1543 c. The **Extended Contract** tab enables the retrieval of additional attributes from the LDAP server, which can be used in assertions to RPs (Figure 4-9). The example shown in 1544 Figure 4-9 adds several AD attributes to the contract.
- 1545

1546 Figure 4-9 Password Credential Validator Extended Contract

MAIN

PingFederate[®]

Manage Credential Validator Instances | Create Credential Validator Instance

Type Instance Configuration Extended Contract Summary

You can extend the attribute contract of this Password Credential Validator instance.

Core Contract

DN

givenName
mail
username

Extend the Contract Action

memberOf Edit | Delete
objectGUID Edit | Delete
sn Edit | Delete
userPrincipalName Edit | Delete

Add

Copyright © 2003-2017
Ping Identity Corporation
All rights reserved
Version 8.3.2.0

Cancel Previous Next Done

1547

- 1548 d. Finally, the **Summary** tab shows all of the values for the configured validator
 1549 (Figure 4-10).

1550 **Figure 4-10 Password Validator Summary**

The screenshot shows the PingFederate interface with the following details:

- MAIN** menu: Manage IdP Adapter Instances | Create Adapter Instance | Manage Password Credential Validators | Create Credential Validator Instance
- IdP Configuration** tab selected.
- Type** tab selected.
- Create Credential Validator Instance** button is visible.
- Type** section:

Instance Name	Password Validator
Instance Id	PasswordValidator
Type	LDAP Username Password Credential Validator
Class Name	org.sourceforge.sam120.domain.LDAPUsernamePasswordCredentialValidator
Parent Instance Name	None
- Instance Configuration** section:

LDAP Datastore	dct.spsd.mss0
Search Base	OU=Demo Users,DC=spsd,DC=mss0
Search Filter	sAMAccountName=\${username}
Scope of Search	Subtree
Case-Sensitive Matching	true
Display Name Attribute	displayName
Mail Attribute	mail
SMS Attribute	
PingID Username Attribute	
- Extended Contract** section:

Attribute	mail
Attribute	givenName
Attribute	DN
Attribute	username
Attribute	memberOf
Attribute	objectGUID
Attribute	sn
Attribute	userPrincipalName
- Buttons at the bottom: Cancel, Previous, Done.

1551

1552 e. Click **Done**, and then click **Save** to complete the setup of the password validator.1553 *4.2.1.2 Configure the HTML Form Adapter*1554 1. On the **IdP Configuration** section tab, click **Adapters**.1555 2. Click **Create New Instance**.1556 a. On the **Type** tab, create the name and ID of the adapter, and select the **HTML Form IdP Adapter** for the **TYPE** (Figure 4-11).
1557

1558 **Figure 4-11 HTML Form Adapter Instance**

The screenshot shows the 'Manage IdP Adapter Instances | Create Adapter Instance' page. The left sidebar has 'MAIN' selected, with 'IdP Configuration' highlighted. The main area has tabs for 'Type', 'IdP Adapter', 'Extended Contract', 'Adapter Attributes', 'Adapter Contract Mapping', and 'Summary'. The 'Type' tab is active. A note says: 'Enter an Adapter Instance Name and Id, select the Adapter Type, and a parent if applicable. The Adapter Type is limited to the adapters currently installed on your server.' The 'INSTANCE NAME' field contains 'HTML Form Adapter'. The 'INSTANCE ID' field contains 'HTMLFormAdapter'. The 'TYPE' dropdown is set to 'HTML Form IdP Adapter'. A link 'Visit Pingidentity.com for additional types' is next to it. The 'PARENT INSTANCE' dropdown is set to 'None'. At the bottom right are 'Cancel' and 'Next' buttons.

1559

- 1560 b. On the **IdP Adapter** tab, add the **Password Validator** instance created in the previous
 1561 section (Figure 4-12). This tab provides several options for customizing the login page
 1562 and supporting password resets and password recovery that would be relevant to a Pro-
 1563 duction deployment. In the lab, password resets were not supported, and these fields
 1564 were left at their default values.

1565 Figure 4-12 Form Adapter Settings

PingFederate

MAIN

- IdP Configuration
- SP Configuration
- Server Configuration

Manage IdP Adapter Instances | Create Adapter Instance

Type IdP Adapter Extended Contract Adapter Attributes Adapter Contract Mapping Summary

Complete the configuration necessary to look up user security contexts in your environment. This configuration was designed into the adapter for use at your site.

CREDENTIAL VALIDATORS
(A list of Password Credential Validators to be used for authentication.)

PASSWORD CREDENTIAL VALIDATOR INSTANCE	Action
Password Validator	Edit Delete

Add a new row to 'Credential Validators'

Field Name	Field Value	Description
CHALLENGE RETRIES	<input type="text" value="3"/>	Max value of User Challenge Retries.
SESSION STATE	<input checked="" type="radio"/> Globally <input type="radio"/> Per Adapter <input type="radio"/> None	Determines how state is maintained within one adapter or between different adapter instances.
SESSION TIMEOUT	<input type="text" value="60"/>	Session Idle Timeout (in minutes). If left blank the timeout will be the Session Max Timeout. Ignored if 'None' is selected for Session State.
SESSION MAX TIMEOUT	<input type="text" value="480"/>	Session Max Timeout (in minutes). Leave blank for indefinite sessions. Ignored if 'None' is selected for Session State.
ALLOW PASSWORD CHANGES	<input type="checkbox"/>	Allows users to change their password using this adapter.
PASSWORD MANAGEMENT SYSTEM	<input type="text"/>	A fully-qualified URL to your password management system where users can change their password. If left blank, password changes are handled by this adapter.
ENABLE 'REMEMBER MY USERNAME'	<input type="checkbox"/>	Allows users to store their username as a cookie when authenticating with this adapter. Once stored, the username is pre-populated in the login form's username field on subsequent transactions.
CHANGE PASSWORD EMAIL NOTIFICATION	<input type="checkbox"/>	Send users an email notification upon a password change. This feature relies on the underlying PCV returning 'mail' and 'givenName' attributes containing the user's first name and e-mail address. Additionally, mail settings should be configured within Server Settings.
SHOW PASSWORD EXPIRING WARNING	<input type="checkbox"/>	Show a warning message to the user on login about an approaching password expiration.
PASSWORD RESET TYPE	<input type="radio"/> Email One-Time Link <input type="radio"/> Email One-Time Password <input type="radio"/> PingID <input type="radio"/> Text Message <input checked="" type="radio"/> None	Select the method to use for self-service password reset. Depending on the selected method, additional settings are required to complete the configuration.

Manage Password Credential Validators Manage SMS Provider Settings Show Advanced Fields

Cancel Previous Next

1566

- 1567 c. On the **Extended Contract** tab, the same attributes returned from AD by the Password
 1568 Validator are added to the adapter contract, to make them available for further use by
 1569 the IdP (Figure 4-13).

1570 **Figure 4-13 Form Adapter Extended Contract**

This adapter type supports the creation of an Extended Adapter Contract after initial deployment of the adapter instance. This Adapter Contract may be used to fulfill the Attribute Contract, look up additional attributes from a local data store, or create a persistent name identifier which uniquely identifies the user passed to your SP partners.

Core Contract	
username	
Extend the Contract	Action
givenName	Edit Delete
mail	Edit Delete
memberOf	Edit Delete
objectGUID	Edit Delete
sn	Edit Delete
userPrincipalName	Edit Delete

Add Cancel Previous Next Done

- 1571 d. On the **Adapter Attributes** tab, select the **Pseudonym** checkbox for the **username** attribute.
 1572 e. There is no need to configure anything on the **Adapter Contract Mapping** tab, as all attributes are provided by the adapter. Click **Done**, and then click **Save** to complete the Form Adapter configuration.

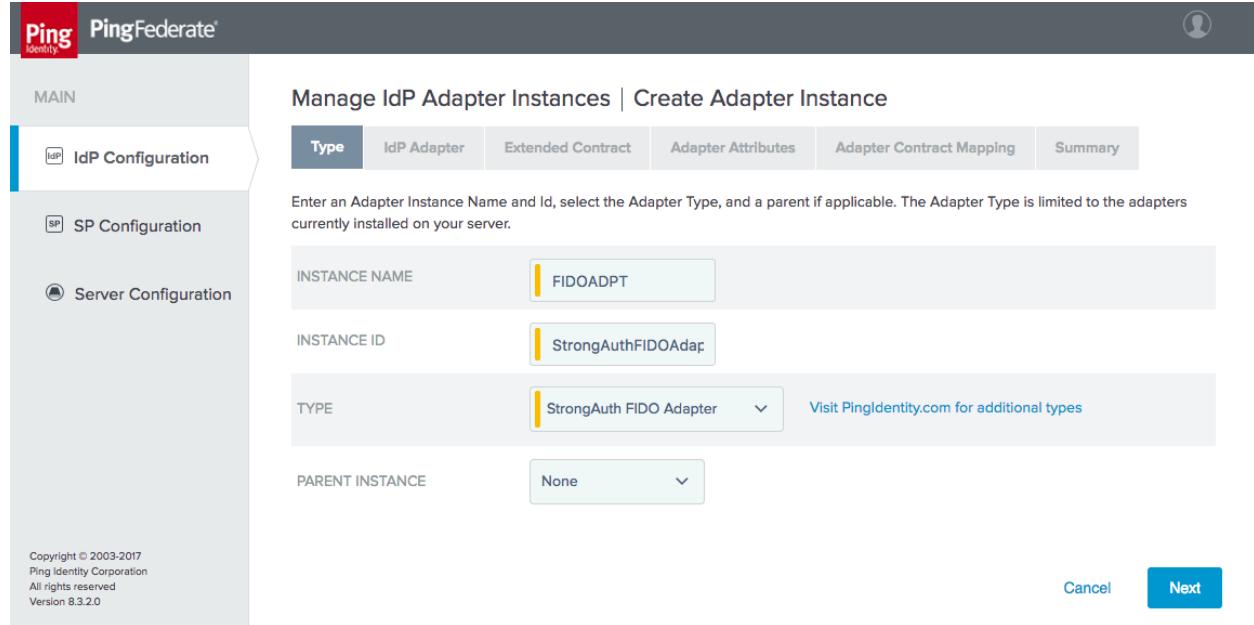
1577 *4.2.1.3 Configure the FIDO U2F Adapter*

1578 Before this step can be completed, the FIDO U2F server, StrongAuth StrongKey CryptoEngine (SKCE),
 1579 must be installed and configured, and the StrongAuth U2F adapter for PingFederate must be installed on
 1580 the IdP. See [Section 6](#) for details on completing these tasks.

- 1581 1. On the **IdP Configuration** section tab, click **Adapters**.
 1582 2. Click **Create New Instance**.

- 1583 a. Enter meaningful values for **INSTANCE NAME** and **INSTANCE ID**. For the **TYPE**, select
 1584 “StrongAuth FIDO Adapter.” Click **Next**.

1585 **Figure 4-14 Create U2F Adapter Instance**



The screenshot shows the 'Manage IdP Adapter Instances' page with the 'Create Adapter Instance' tab selected. The 'Type' dropdown is set to 'StrongAuth FIDO Adapter'. The 'INSTANCE NAME' field contains 'FIDOADPT' and the 'INSTANCE ID' field contains 'StrongAuthFIDOAdap'. The 'PARENT INSTANCE' dropdown is set to 'None'. A note at the bottom right suggests visiting [Pingidentity.com](#) for additional types. The left sidebar shows navigation options: MAIN, IdP Configuration (selected), SP Configuration, and Server Configuration. The bottom left corner displays copyright information: 'Copyright © 2003-2017 Ping Identity Corporation. All rights reserved. Version 8.3.2.0'. The bottom right corner has 'Cancel' and 'Next' buttons.

- 1586
 1587 b. On the **IdP Adapter** tab, keep the default value of the **HTML FORM TEMPLATE NAME** to
 1588 use the template that is provided with the StrongAuth U2F plugin, or specify a custom
 1589 template if desired to change the design of the user interface (Figure 4-15). The **FIDO**
 1590 **SERVER URL**, **DOMAIN ID**, **SKCE SERVICE USER**, and **SKCE SERVICE USER PASSWORD** are
 1591 determined in the setup of the SKCE; refer to [Section 6](#) for details.

1592 Figure 4-15 U2F Adapter Settings

Manage IdP Adapter Instances | Create Adapter Instance

Type	IdP Adapter	Extended Contract	Adapter Attributes	Adapter Contract Mapping	Summary
Complete the configuration necessary to look up user security contexts in your environment. This configuration was designed into the adapter for use at your site.					
Set the FIDO configuration from your StrongAuth CryptoEngine:					
Field Name	Field Value	Description			
HTML FORM TEMPLATE NAME	fido-main-template.html	HTML template (in <pf_home>/server/default/conf/template) to render for form submission.			
FIDO SERVER URL	https://strongauth2.lpsd.mss0:8181	The URL of the FIDO server. Must start with https and include the port number (8181 by default).			
DOMAIN ID	2	The Domain ID of the SKCE.			
SKCE SERVICE USER	svcfidouser	The service user that will communicate with the SKCE.			
SKCE SERVICE USER PASSWORD	dontPutRealPasswordsInScreenshots	The password for the service user.			

Cancel Previous Next Done

1593

- 1594 c. There is no need to extend the contract for the U2F adapter; therefore, skip the **Extended Contract** tab.
- 1595
- 1596 d. On the **Adapter Attributes** tab, select the **Pseudonym** checkbox for the **username** attribute.
- 1597
- 1598 e. There is also no need for an **Adapter Contract Mapping**; therefore, skip the **Adapter Contract Mapping** tab.
- 1599
- 1600 f. Click **Done**, and then click **Save**.

1601 *4.2.1.4 Configure the Authentication Policies*

- 1602 1. On the **IdP Configuration** page, click **Policies**.
- 1603 a. Under **Manage Authentication Policies**, click the **ENABLE IDP AUTHENTICATION POLICIES** checkbox, and create a policy that starts with the **HTML Form Adapter** action (Figure 4-16).
- 1604
- 1605

- 1606 i. On the **Success** branch, add the FIDO U2F adapter (**FIDOADPT**) for the **Action**.
 1607 ii. Click **Save**.

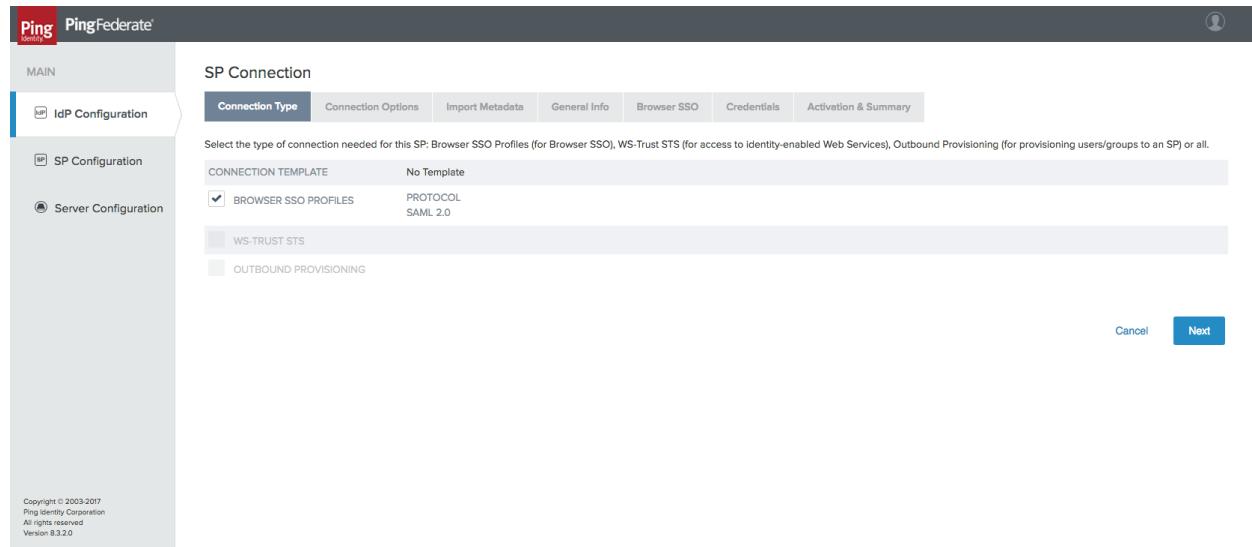
1608 **Figure 4-16 IdP Authentication Policy**

Copyright © 2003-2017
Ping Identity Corporation
All rights reserved
Version 8.3.2.0

- 1609
 1610 **4.2.2 Configure the SP Connection**
 1611 Each RP that will receive authentication assertions from the IdP must be configured as an SP connection.
 1612 As explained in [Section 3.4.2.1](#), this activity requires coordination between the administrators of the IdP
 1613 and the RP to provide the necessary details to configure the connection. Exchanging metadata files can
 1614 help automate some of the configuration process.
 1615 This section documents the configuration for the SP connection between the SAML IdP in the NCCoE Lab
 1616 and the OAuth AS in the Motorola Solutions cloud instance.

- 1617 1. To create a new SP connection, click the **IdP Configuration** section tab, and then click **Create**
 1618 **New** under **SP Connections**.
- 1619 a. On the **Connection Type** tab, select **BROWSER SSO PROFILES**, and select the **SAML 2.0**
 1620 protocol (Figure 4-17). In this case, SAML 2.0 is pre-selected because no other protocols
 1621 are enabled on this IdP.

1622 **Figure 4-17 SP Connection Type**



- 1623 b. On the **Connection Options** tab, only **BROWSER SSO** needs to be selected.
- 1624 c. If metadata for the SP is available, it can be imported on the **Import Metadata** tab. This
 1625 metadata can be specified in the form of a file upload or URL.
- 1626 d. On the **General Info** tab, enter the **PARTNER'S ENTITY ID (CONNECTION ID)**
 1627 (Figure 4-18); this must match the **ENTITY ID** configured on the **Federation Info** tab in
 1628 the **Server Configuration** of the SP. The SP's **BASE URL** should also be added on this
 1629 **General Info** tab.
- 1630

1631 **Figure 4-18 SP Connection General Info**

The screenshot shows the PingFederate interface with the following details:

- Header:** PingFederate
- Left Sidebar (MAIN):**
 - IdP Configuration** (selected)
 - SP Configuration**
 - Server Configuration**
- Top Navigation Bar:** Connection Type, Connection Options, Metadata URL, General Info, Browser SSO, Credentials.
- Active Tab:** Activation & Summary
- Content Area:**

This information identifies your partner's unique connection identifier (Connection ID). Connection Name represents the plain-language identifier for this connection. Optionally, you can specify multiple virtual server IDs for your own server to use when communicating with this partner. If set, these virtual server IDs will be used in place of the unique protocol identifier configured for your server in Server Settings. The Base URL may be used to simplify configuration of partner endpoints.

PARTNER'S ENTITY ID (CONNECTION ID)	ctoPingFed_entityID
CONNECTION NAME	ctoPingFed_entityID
VIRTUAL SERVER IDS	<input type="text"/> Add
BASE URL	https://idm.sandbox.motorolasolutions.co
COMPANY	<input type="text"/>
CONTACT NAME	<input type="text"/>
CONTACT NUMBER	<input type="text"/>
CONTACT EMAIL	<input type="text"/>
APPLICATION NAME	<input type="text"/>
APPLICATION ICON URL	<input type="text"/>
LOGGING MODE	<input type="radio"/> NONE <input checked="" type="radio"/> STANDARD <input type="radio"/> ENHANCED <input type="radio"/> FULL
- Bottom Buttons:** Cancel, Previous, Next, Save

1632

1633 e. On the **Browser SSO** tab, click **Configure Browser SSO**. This opens another multi-tabbed configuration screen.

1634

1635 i. On the **SAML Profiles** tab, different SSO and Single Log-Out (SLO) profiles can be enabled (Figure 4-19). Only **SP-INITIATED SSO** is demonstrated in this lab build.

1637 **Figure 4-19 SP Browser SSO Profiles**

A SAML Profile defines what kind of messages may be exchanged between an Identity Provider and a Service Provider, and how the messages are transported (bindings). As an IdP, you configure this information for your SP connection.

Single Sign-On (SSO) Profiles	Single Logout (SLO) Profiles
<input checked="" type="checkbox"/> IDP-INITIATED SSO	<input type="checkbox"/> IDP-INITIATED SLO
<input checked="" type="checkbox"/> SP-INITIATED SSO	<input type="checkbox"/> SP-INITIATED SLO

1638

- 1639 ii. On the **Assertion Lifetime** tab, time intervals during which SPs should consider
 1640 assertions valid can be configured in minutes before and after assertion crea-
 1641 tion. In the lab, these were both set to the default of five minutes.
- 1642 iii. On the **Assertion Creation** tab, click **Configure Assertion Creation**. This opens a
 1643 new multi-tabbed configuration screen.
- 1644 1) On the **Identity Mapping** tab, select the **STANDARD** mapping (Figure 4-20).
 1645 The other options are more suitable for situations where identifiers are
 1646 sensitive or where there are privacy concerns over the tracking of users.

1647 Figure 4-20 Assertion Identity Mapping

SP Connection | Browser SSO | Assertion Creation

Identity mapping is the process in which users authenticated by the IdP are associated with user accounts local to the SP. Select the type of name identifier that you will send to the SP. Your selection may affect the way that the SP will look up and associate the user to a specific local account.

- STANDARD: Send the SP a known attribute value as the name identifier. The SP will often use account mapping to identify the user locally.
- PSEUDONYM: Send the SP a unique, opaque name identifier that preserves user privacy. The identifier cannot be traced back to the user's identity at this IdP and may be used by the SP to make a persistent association between the user and a specific local account. The SP will often use account linking to identify the user locally.
- INCLUDE ATTRIBUTES IN ADDITION TO THE PSEUDONYM.
- TRANSIENT: Send the SP an opaque, temporary value as the name identifier.
- INCLUDE ATTRIBUTES IN ADDITION TO THE TRANSIENT IDENTIFIER.

Copyright © 2003-2017
Ping Identity Corporation
All rights reserved
Version 8.3.2.0

Cancel Save Draft Next

1648

- 1649 2) On the **Attribute Contract** tab, extend the contract to include the **mail** and **uid** attributes with the basic name format (Figure 4-21). Other attributes can be added here as needed.
- 1650
- 1651

1652 Figure 4-21 Assertion Attribute Contract

SP Connection | Browser SSO | Assertion Creation

An Attribute Contract is a set of user attributes that this server will send in the assertion.

Attribute Contract	Subject Name Format	Action
SAML_SUBJECT	urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified	
Extend the Contract	Attribute Name Format	Action
mail	urn:oasis:names:tc:SAML:2.0:attrname-format:basic	Edit Delete
uid	urn:oasis:names:tc:SAML:2.0:attrname-format:basic	Edit Delete
	urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified	Add

Copyright © 2003-2017
Ping Identity Corporation
All rights reserved
Version 8.3.2.0

Cancel Previous Next Done Save

1653

- 1654
- 1655 3) On the **Authentication Source Mapping** tab, attributes provided by au-
- 1656 thentication adapters and policy contracts can be mapped to the assertion
- 1657 attribute contract, identifying which data will be used to populate the as-
- 1658 sertions. The FIDO U2F adapter and the HTML Form Adapter should appear
- 1659 under **Adapter Instance Name**. Select the HTML Form Adapter, as it can
- 1660 provide the needed attributes from LDAP via the Password Validator and
- 1661 the AD data store connection. This brings up another multi-tabbed configu-
- 1662 ration screen.
- 1663 a) The **Adapter Instance** tab shows the attributes that are returned by
- 1664 the selected adapter. Click **Next**.
- 1665 b) The **Mapping Method** tab provides options to query additional data
- 1666 stores to build the assertions, but in this case, all of the required at-
- 1667 tributes are provided by the HTML Form Adapter. Select **USE ONLY**
- 1668 **THE ADAPTER CONTRACT VALUES IN THE SAML ASSERTION**.
- 1669 c) On the **Attribute Contract Fulfillment** tab, map the **SAML SUBJECT**,
- 1670 **mail**, and **uid** attributes to the **username**, **mail**, and **userPrincipal-**
- 1671 **Name** adapter values (Figure 4-22).

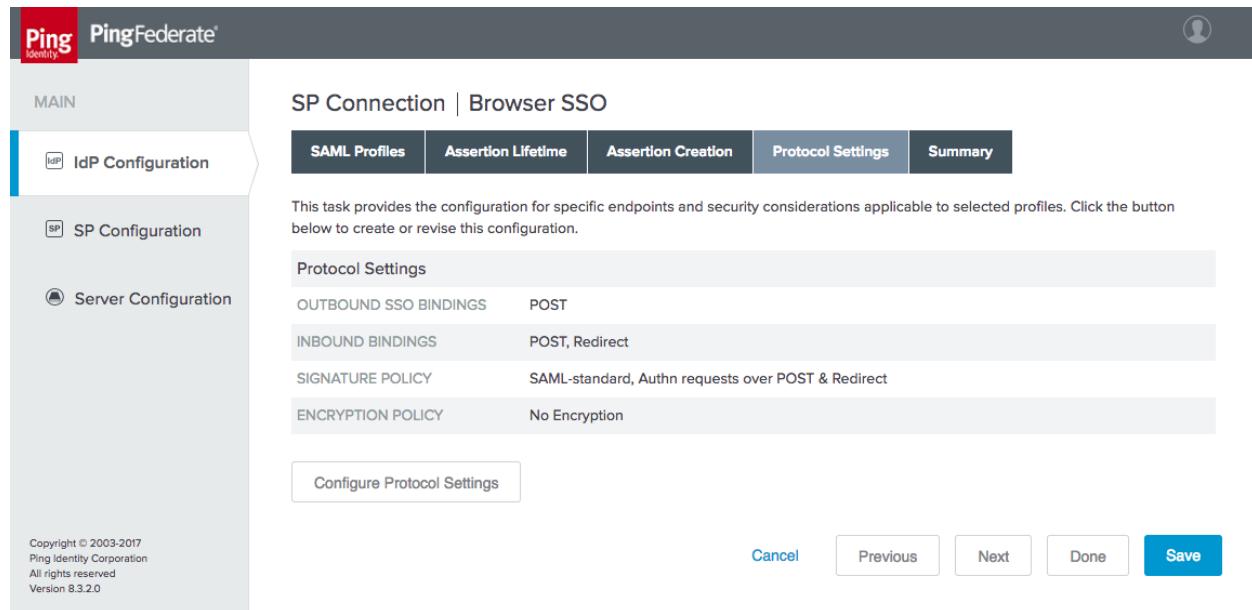
1671 **Figure 4-22 Assertion Attribute Contract Fulfillment**

Attribute Contract	Source	Value	Actions
SAML_SUBJECT	Adapter	username	None available
mail	Adapter	mail	None available
uid	Adapter	userPrincipalName	None available

1672

- 1673 d) No **Issuance Criteria** are required; therefore, skip the **Issuance Criteria**
 1674 tab.
- 1675 e) Click **Done** to exit the IdP Adapter Mapping.
- 1676 4) Click **Done** to exit the Assertion Creation.
- 1677 iv. On the **Protocol Settings** tab, options such as additional SAML bindings, signa-
 1678 ture policy details, and assertion encryption policies can be specified
 1679 (Figure 4-23). For the lab build, these values were left at their default settings.

1680 **Figure 4-23 Browser SSO Protocol Settings**



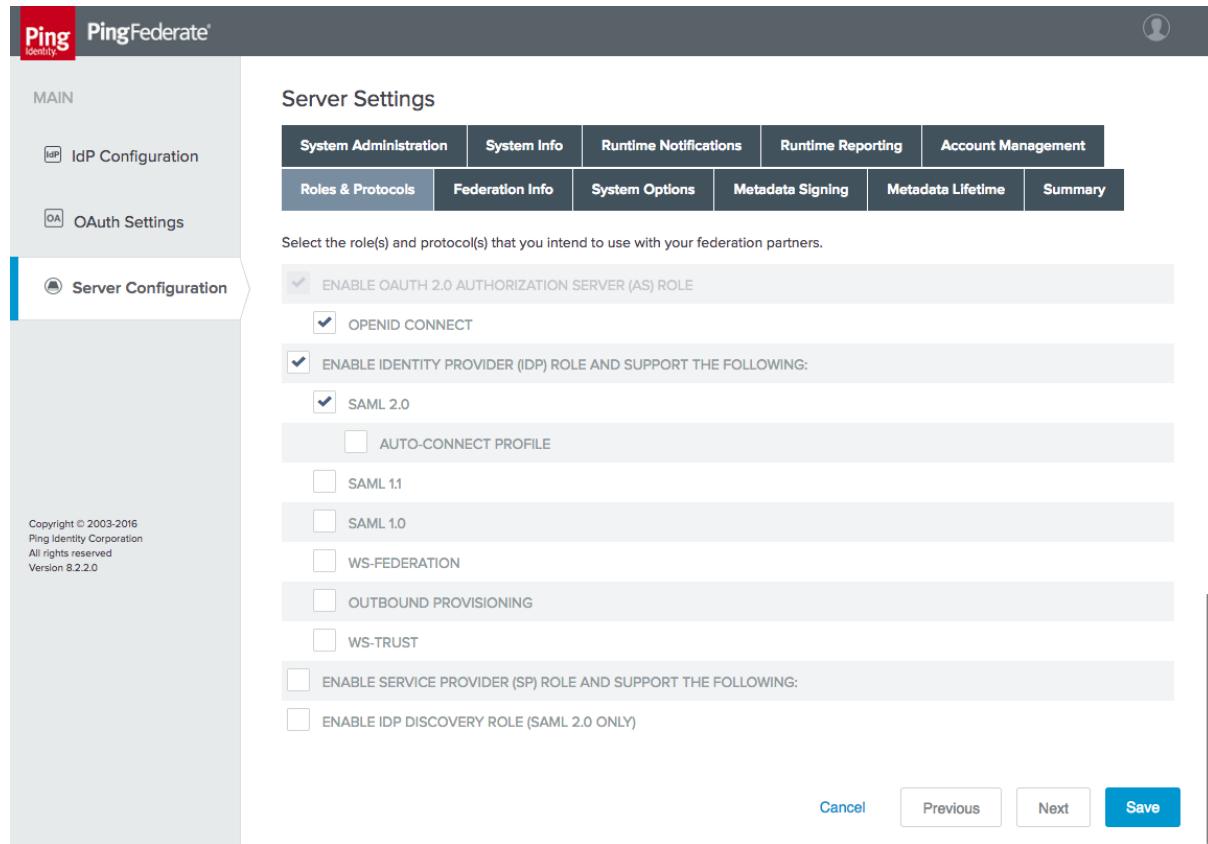
- 1681
- 1682 v. Click **Done** to exit Browser SSO.
- 1683 f. On the **Credentials** tab, the certificate to use for signing assertions can be specified. A self-signed certificate can be generated by PingFederate, or a trusted certificate can be obtained and uploaded. Click **Configure Credentials** to create or manage signing credentials.
- 1684
- 1685
- 1686
- 1687 g. On the **Activation & Summary** tab, the connection status can be set to **ACTIVE**. All con-
 1688 figured settings for the SP connection are also displayed for verification.
- 1689 h. Click **Save** to complete the SP connection configuration.
- 1690 This completes the configuration of the SAML IdP.

1691 4.3 How to Install and Configure the OIDC Identity Provider

1692 1. On the **Server Configuration** section tab, click **Server Settings**.

1693 a. On the **Roles & Protocols** tab, enable the roles and protocols as shown in Figure 4-24.
 1694 Although the OIDC IdP does not actually use the SAML protocol, some required configu-
 1695 ration settings are unavailable if the IdP role is not enabled.

1696 **Figure 4-24** OIDC IdP Roles



The screenshot shows the 'PingFederate' interface with the 'MAIN' sidebar. The 'Server Configuration' tab is selected. In the main content area, the 'Server Settings' page is displayed. The 'Roles & Protocols' tab is active. Under the 'ENABLE IDENTITY PROVIDER (IDP) ROLE AND SUPPORT THE FOLLOWING:' section, the 'SAML 2.0' checkbox is checked, while 'SAML 1.0', 'WS-FEDERATION', 'OUTBOUND PROVISIONING', and 'WS-TRUST' are unchecked. Other tabs like 'System Administration', 'System Info', 'Runtime Notifications', 'Runtime Reporting', and 'Account Management' are visible at the top.

1697

1698 b. On the **Federation Info** tab, specify the **BASE URL** and **SAML 2.0 ENTITY ID**. The **BASE**
 1699 **URL** must be a URL that is exposed to clients.

1700 2. On the **OAuth Settings** section tab, click **Authorization Server Settings** to configure general
 1701 OAuth and OIDC parameters. The OIDC IdP's settings on this page are identical to those for the
 1702 OAuth AS; refer to [Section 3.3](#) for notes on these settings.

1703 3. On the **OAuth Settings** section tab, click **Scope Management**.

1704 a. Add the scopes defined in the OpenID Connect Core specification [\[15\]](#):

- 1705 ▪ openid
- 1706 ▪ profile
- 1707 ▪ email
- 1708 ▪ address
- 1709 ▪ phone

1710 [**4.3.1 Configuring Authentication to the OIDC IdP**](#)

1711 In the lab architecture, the OIDC IdP supports FIDO UAF authentication through integration with the
1712 NNAS and the Nok Nok Labs Gateway, using the Nok Nok FIDO UAF adapter for PingFederate.

1713 Configuring UAF authentication to the OIDC IdP cannot be completed until the Nok Nok Labs servers are
1714 available and the UAF plugin has been installed on the IdP server as specified in [Section 5](#).

1715 [**4.3.1.1 Configure the FIDO UAF Plugin**](#)

1716 The steps to configure the FIDO UAF plugin for the OIDC IdP are identical to those documented in
1717 [Section 3.4.1.1](#) for direct authentication using UAF at the AS. The only difference in the lab build was the
1718 URLs for the NNAS and the Nok Nok Labs Gateway, as the AS and the OIDC IdP used two different
1719 instances of the Nok Nok Labs server.

1720 [**4.3.1.2 Configure an Access Token Management Instance**](#)

1721 1. On the **OAuth Settings** section tab, click **Access Token Management**.

1722 2. Click **Create New Instance**.

1723 a. On the **Type** tab, provide an **INSTANCE NAME** and **INSTANCE ID** (Figure 4-25).

1724 i. Select **Internally Managed Reference Tokens** for the **TYPE**.

1725 **Figure 4-25 Create Access Token Manager**

Access Token Management | Create Access Token Management Instance

Type Instance Configuration Access Token Attribute Contract Resource URLs Access Control Summary

Enter an Access Token Management Instance Name and Id, select the plugin Access Token Management Type, and a parent if applicable. The types available are limited to the plugins currently installed on your server.

INSTANCE NAME	FIDO UAF
INSTANCE ID	fidoUaf
TYPE	Internally Managed Reference Tokens Visit Pingidentity.com for additional types
PARENT INSTANCE	None

Cancel Next

1726

1727 Although we have selected reference tokens, the ID Token is always issued in
 1728 the form of a JWT. The token that is being configured here is not the ID Token,
 1729 but rather the access token that will be issued to authorize the RP to call the
 1730 userinfo endpoint at the IdP to request additional claims about the user.
 1731 Because this access token only needs to be validated by the OIDC IdP itself,
 1732 reference tokens are sufficient. In the Authorization Code flow, the RP obtains
 1733 both the ID Token and the access token in exchange for the authorization code
 1734 at the IdP's token endpoint.

- 1735 b. Click the **Instance Configuration** tab to configure some security properties of the access
 1736 token, such as its length and lifetime (Figure 4-26). For the lab build, the default values
 1737 were accepted.

1738 Figure 4-26 Access Token Manager Configuration

Access Token Management | Create Access Token Management Instance

Type	Instance Configuration	Access Token Attribute Contract	Resource URLs	Access Control	Summary
Complete the configuration necessary to issue and validate access tokens. This configuration was designed into, and is specific to, the selected Access Token Management plugin.					
Field Name	Field Value	Description			
TOKEN LENGTH	28	Defines how many alphanumeric characters make up an access token.			
TOKEN LIFETIME	120	Defines how long, in minutes, an access token is valid.			
LIFETIME EXTENSION POLICY	No Extension	Dictates which tokens are eligible for lifetime extension. Similar to a session inactivity timeout, the lifetime period of an access token can be reset each time the token is validated at the AS (subject to the values defined for the Lifetime Extension Threshold Percentage and the Maximum Token Lifetime).			
MAXIMUM TOKEN LIFETIME		(Optional) Defines an absolute maximum token lifetime, in minutes, for use with the Lifetime Extension Policy. An access token's lifetime cannot be extended beyond this setting.			
LIFETIME EXTENSION THRESHOLD PERCENTAGE	30	Defines the percentage of a token's lifetime remaining before the lifetime is actually extended, which can improve cluster performance.			

Show Advanced Fields

Cancel Previous Next Done Save

1739

- 1740 c. On the **Access Token Attribute Contract** tab, extend the contract with any attributes
 1741 that will be included in the ID Token (Figure 4-27). In the example shown in Figure 4-27,
 1742 several attributes that will be queried from AD have been added.

1743 **Figure 4-27 Access Token Attribute Contract**

The screenshot shows the 'Access Token Management' section of the PingFederate interface. The left sidebar has tabs for 'MAIN' (selected), 'IdP Configuration', 'OAuth Settings' (which is active), and 'Server Configuration'. The right panel title is 'Access Token Management | Create Access Token Management Instance'. Below it, a table lists attributes under the heading 'Extend the Contract' with 'Action' columns:

Attribute	Action
department	Edit Delete
email	Edit Delete
family_name	Edit Delete
given_name	Edit Delete
l	Edit Delete
name	Edit Delete
phone_number	Edit Delete
postal_code	Edit Delete
preferred_username	Edit Delete
state	Edit Delete
street_address	Edit Delete
sub	Edit Delete
title	Edit Delete
updated_at	Edit Delete

At the bottom right are buttons for 'Cancel', 'Previous', 'Next', 'Done', and a blue 'Save' button.

1744

- 1745 d. There is no need to configure the **Resource URIs** or **Access Control** tabs; these tabs can
1746 be skipped.
- 1747 e. Click **Done**, and then click **Save**.

1748 *4.3.1.3 Configure an IdP Adapter Mapping*

1749 The IdP Adapter Mapping determines how the persistent grant attributes are populated using
1750 information from authentication adapters.

- 1751 1. Click the **OAuth Settings** section tab, and then click **IdP Adapter Mapping**.
- 1752 2. Select the UAF adapter instance created in [Section 4.3.1.1](#), and then click **Add Mapping**.

- 1753 a. On the **Contract Fulfillment** tab, map both **USER_KEY** and **USER_NAME** to the
 1754 **username** value returned from the adapter (Figure 4-28).

1755 **Figure 4-28 Access Token Contract Fulfillment**

Contract	Source	Value	Actions
USER_KEY	Adapter	username	None available
USER_NAME	Adapter	username	None available

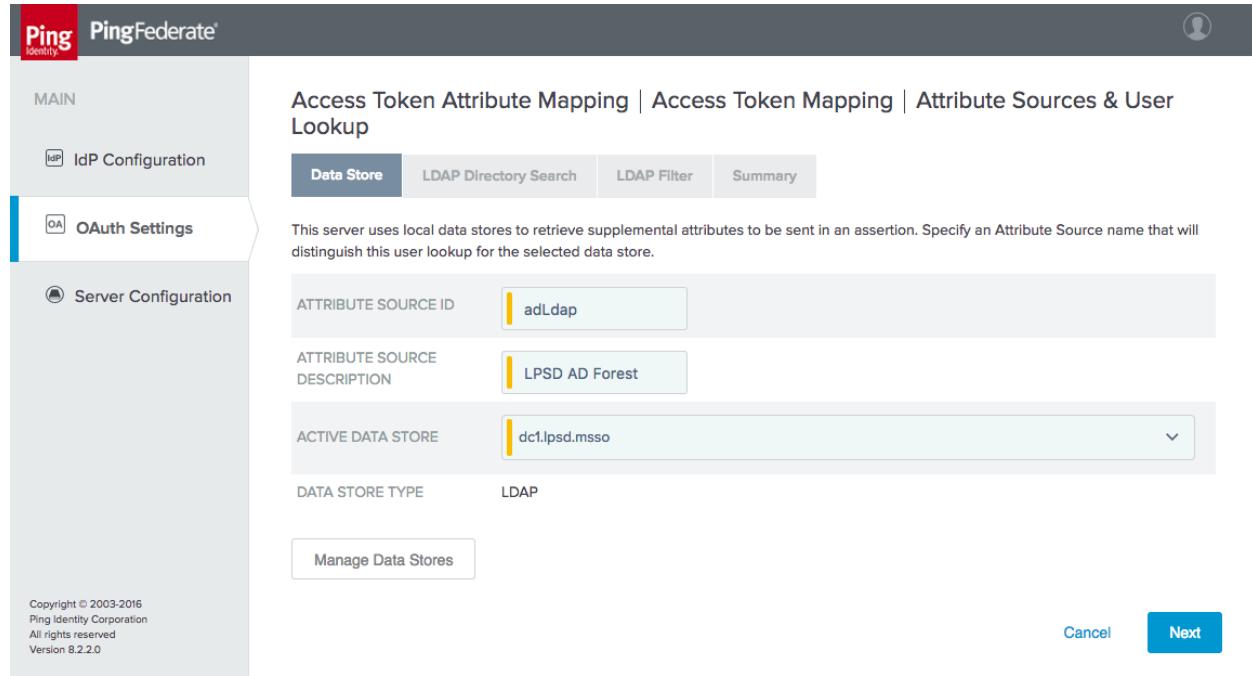
1756

1757 *4.3.1.4 Configure an Access Token Mapping*

1758 The Access Token Mapping determines how the access token attribute contract is populated. In this
 1759 example, the values returned from the adapter are supplemented with attributes retrieved from AD,
 1760 and issuance criteria are used to require the user to be actually found in AD for a token to be issued.
 1761 Depending on the credential and access life-cycle processes used in a given organization, there may be a
 1762 lag in deactivating the authenticator or the AD account when a user's access is terminated.
 1763 Organizations' authentication policies should account for these conditions and should allow or deny
 1764 access appropriately.

- 1765 1. On the **OAuth Settings** section tab, click **Access Token Mapping**.
- 1766 2. Under **CONTEXT** and **ACCESS TOKEN MANAGER**, select the IdP Adapter and Access Token
 Manager created in the preceding steps, and click **Add Mapping**.
 - 1768 a. On the **Attribute Sources & User Lookup** tab, click **Add Attribute Source**. This brings up
 another multi-tabbed configuration.
 - 1770 i. On the **Data Store** tab, give the attribute source an ID and description
 (Figure 4-29). For **ACTIVE DATA STORE**, select the user store created in
 [Section 4.1](#).

1773 Figure 4-29 Data Store for User Lookup



1774

1782 Figure 4-30 Attribute Directory Search

Access Token Attribute Mapping | Access Token Mapping | Attribute Sources & User Lookup

Please configure your directory search. This information, along with the attributes supplied in the contract, will be used to fulfill the contract.

ROOT OBJECT CLASS	ATTRIBUTE	Action
	Subject DN	
	department	Remove
	displayName	Remove
	givenName	Remove
I	mail	Remove
	objectClass	Remove
	postalCode	Remove
	sn	Remove
	st	Remove
	streetAddress	Remove
	telephoneNumber	Remove
	title	Remove
	whenChanged	Remove
<input type="button" value="- SELECT -"/> <input type="button" value="Add Attribute"/>		

[View Attribute Contract](#)

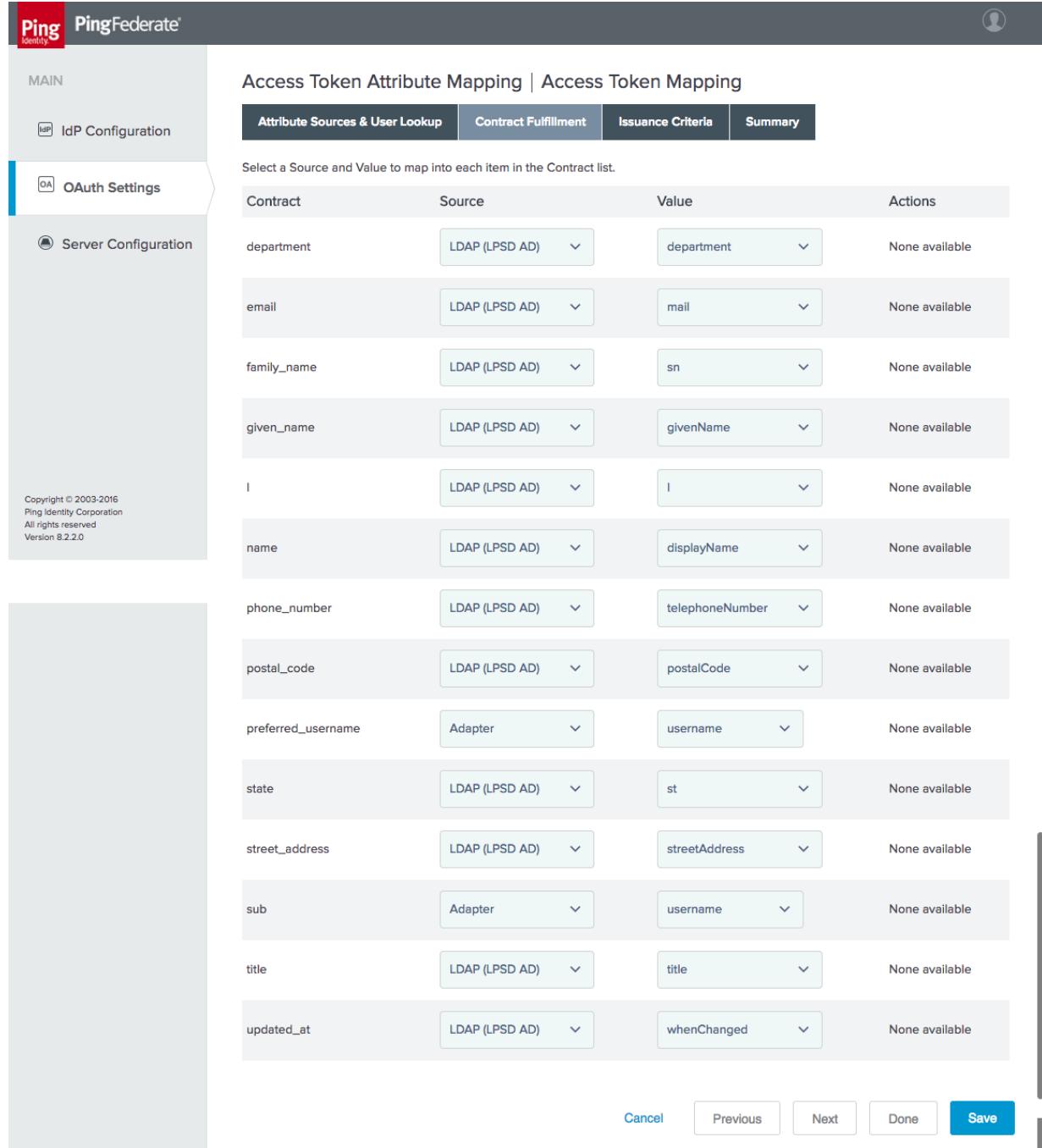
Cancel Previous Next Done **Save**

1783

- 1784 iii. On the **LDAP Filter** tab, create the filter to select the relevant user account. In
1785 this example, the username from the adapter is matched against the AD SAM
1786 account name:
1787 `sAMAccountName=${adapter.username}`
- 1788 iv. Click **Done** to exit the attribute source configuration.

- 1789 b. On the **Contract Fulfillment** tab, specify the source and value to use for each attribute in
 1790 the access token attribute contract (Figure 4-31).

1791 **Figure 4-31 Access Token Contract Fulfillment**



The screenshot shows the 'Access Token Attribute Mapping' page in the PingFederate interface. The left sidebar has sections for MAIN, IdP Configuration, OAuth Settings (selected), and Server Configuration. A copyright notice at the bottom left reads: Copyright © 2003-2016 Ping Identity Corporation. All rights reserved. Version 8.2.2.0.

The main content area is titled 'Access Token Attribute Mapping | Access Token Mapping'. It displays a table with columns: Contract, Source, Value, and Actions. The table lists 15 attributes and their mappings:

Contract	Source	Value	Actions
department	LDAP (LPSD AD)	department	None available
email	LDAP (LPSD AD)	mail	None available
family_name	LDAP (LPSD AD)	sn	None available
given_name	LDAP (LPSD AD)	givenName	None available
I	LDAP (LPSD AD)	I	None available
name	LDAP (LPSD AD)	displayName	None available
phone_number	LDAP (LPSD AD)	telephoneNumber	None available
postal_code	LDAP (LPSD AD)	postalCode	None available
preferred_username	Adapter	username	None available
state	LDAP (LPSD AD)	st	None available
street_address	LDAP (LPSD AD)	streetAddress	None available
sub	Adapter	username	None available
title	LDAP (LPSD AD)	title	None available
updated_at	LDAP (LPSD AD)	whenChanged	None available

At the bottom right are buttons for Cancel, Previous, Next, Done, and Save.

- 1793 c. On the **Issuance Criteria** tab, define a rule that will prevent token issuance if the user
 1794 account doesn't exist in AD (Figure 4-32). In this case, the **objectClass** attribute, which
 1795 all AD objects have, is checked for the **Value** called **user**. If no user account is found in
 1796 AD, this attribute will have no **Value**, the **Condition** will be false, and the specified **Error
 1797 Result** will appear in the PingFederate server log.

1798 **Figure 4-32 Access Token Issuance Criteria**

Source	Attribute Name	Condition	Value	Error Result	Action
LDAP (IpsdAd)	objectClass	multi-value contains (case insensitive)	user	User object does not exist in AD	Edit Delete

- 1799 d. Click **Done**, and then click **Save** to finish the Access Token Attribute Mapping configuration.

1800 *4.3.1.5 Configure an OIDC Policy*

- 1801 1. On the **OAuth Settings** tab, click **OpenID Connect Policy Management**.
- 1802 2. Click **Add Policy**.
- 1803 a. On the **Manage Policy** tab, create a **POLICY ID** and **NAME**, and select the **INCLUDE USER
 INFO IN ID TOKEN** checkbox (Figure 4-33). This selection means that the user's attrib-
 utes will be included as claims in the ID Token JWT. The advantage of this approach is
 that the RP can directly obtain user attributes from the ID Token without making addi-
 tional requests to the IdP. The alternative is to include only a subject claim in the ID To-
 ken, and to have the RP call the IdP's userinfo endpoint to obtain additional user attrib-
 utes.

1812 Figure 4-33 OIDC Policy Creation

The screenshot shows the 'Policy Management | Policy' screen in the PingFederate web interface. The left sidebar has 'MAIN' selected, with 'IdP Configuration' and 'OAuth Settings' visible. The main content area is titled 'Policy Management | Policy'. It displays fields for creating a new policy:

- POLICY ID:** fidoUaf
- NAME:** FIDO UAF
- ACCESS TOKEN MANAGER:** FIDO UAF
- ID TOKEN LIFETIME:** 5 minutes
- INCLUDE SESSION IDENTIFIER IN ID TOKEN:**
- INCLUDE USER INFO IN ID TOKEN:**

At the bottom right are buttons for **Cancel**, **Next**, **Done**, and a blue **Save** button.

1813

- 1814 b. On the **Attribute Contract** tab, the set of attributes in the contract can be edited
 1815 (Figure 4-34). The contract is automatically populated with the standard claims defined
 1816 in the OIDC Core specification. In the example shown in Figure 4-34, some claims have
 1817 been removed and others have been added to accommodate the attribute available
 1818 from AD.

1819 Figure 4-34 OIDC Policy Attribute Contract

The screenshot shows the PingFederate Policy Management interface. The left sidebar has sections for MAIN, IdP Configuration, OAuth Settings (which is selected), and Server Configuration. The bottom-left of the sidebar displays copyright information: Copyright © 2003-2016 Ping Identity Corporation. All rights reserved. Version 8.2.2.0. The main content area is titled "Policy Management | Policy" and shows the "Attribute Contract" tab selected. A note states: "The required Attribute Contract here consists of a user identifier ("sub"). You may extend the contract to include attributes that will be returned to OAuth clients in response to requests received at the PingFederate UserInfo endpoint. The preset extended-contract list contains OpenID Connect standard attributes: add, edit, or delete items in this list as needed for this policy." Below this is a table titled "Attribute Contract" with two columns: "Extend the Contract" and "Action". The table lists various attributes with "Edit | Delete" links: address.locality, address.postal_code, address.street_address, department, email, family_name, given_name, name, phone_number, preferred_username, state, title, and updated_at. At the bottom of the table are "Cancel", "Previous", "Next", "Done", and a blue "Save" button.

1820

- 1821 c. Skip the **Attribute Sources & User Lookup** tab; there is no need to retrieve additional
1822 attributes.
- 1823 d. On the **Contract Fulfillment** tab, populate the OIDC attributes with the corresponding
1824 values from the Access Token context (Figure 4-35).

1825 **Figure 4-35 OIDC Policy Contract Fulfillment**

The screenshot shows the PingFederate Policy Management interface. The left sidebar has sections for MAIN, IdP Configuration, OAuth Settings (which is selected), and Server Configuration. A copyright notice at the bottom left reads: Copyright © 2003-2016 Ping Identity Corporation. All rights reserved. Version 8.2.2.0.

The main content area is titled "Policy Management | Policy" and shows the "Contract Fulfillment" tab selected. It displays a table of attributes and their fulfillment sources:

Attribute Contract	Source	Value	Actions
address.locality	Access Token	I	None available
address.postal_code	Access Token	postal_code	None available
address.street_address	Access Token	street_address	None available
department	Access Token	department	None available
email	Access Token	email	None available
family_name	Access Token	family_name	None available
given_name	Access Token	given_name	None available
name	Access Token	name	None available
phone_number	Access Token	phone_number	None available
preferred_username	Access Token	preferred_username	None available
state	Access Token	state	None available
sub	Access Token	sub	None available
title	Access Token	title	None available
updated_at	Access Token	updated_at	None available

At the bottom right are buttons for Cancel, Previous, Next, Done, and Save.

1826

- 1827 e. There is no need for additional issuance criteria; therefore, skip the **Issuance Criteria**
1828 tab.
- 1829 f. Click **Save** to complete the OIDC Policy configuration.

4.3.2 Configuring the OIDC Client Connection

Registering a client at an OIDC IdP is analogous to creating an SP connection at a SAML IdP. Some coordination is required between the administrators of the two systems. The client ID and client secret must be provided to the RP, and the RP must provide the redirect URI to the IdP.

1. To add a client, click the **OAuth Settings** section tab, and then click **Create New** under **Clients**.

a. Create a **CLIENT ID** and **CLIENT SECRET** (Figure 4-36). If mutual TLS authentication is being used instead, the RP must provide its certificate, which can be uploaded to the client creation page. Only the **Authorization Code** grant type is needed for this integration. In the example shown in Figure 4-36, user prompts to authorize the sharing of the user's attributes with the RP have been disabled in favor of streamlining access to apps.

1840 Figure 4-36 OIDC Client Configuration

Client

Manage the configuration and policy information about a client.

CLIENT ID: MotorolaAS

CLIENT AUTHENTICATION: NONE CLIENT SECRET

SECRET: [Generate Secret](#)

[CHANGE SECRET](#)

CLIENT TLS CERTIFICATE

ISSUER: [- SELECT -](#)

SUBJECT DN:

You can also extract the Subject DN from a certificate file.
No file selected [Choose file](#)

[Extract](#)

NAME: Motorola's AS

DESCRIPTION:

REDIRECT URIS: <https://idm.sandbox.motorolasolutions.com/sp/ey/jpc3MjOJodhRwcpltwv63AxLmwvc2QubXNzbzo5MDMxh0cb.openid> [Edit](#) [Delete](#)
<https://mfes-nccoe.noknoktest.com:8443/nngateway/ml/ob/reg> [Edit](#) [Delete](#)

LOGO URL: <https://op1.lpsd.mso:9031/assets/image>

BYPASS AUTHORIZATION APPROVAL: Bypass

RESTRICT SCOPES: Restrict

ALLOWED GRANT TYPES: Authorization Code
 Resource Owner Password Credentials
 Refresh Token
 Implicit
 Client Credentials
 Access Token Validation (Client is a Resource Server)
 Extension Grants

DEFAULT ACCESS TOKEN MANAGER: FIDO UAF

PERSISTENT GRANTS EXPIRATION: Use Global Setting
 Grants Do Not Expire
 Days

REFRESH TOKEN ROLLING POLICY: Use Global Setting Don't Roll Roll

OPENID CONNECT: ID Token Signing Algorithm: Default

Policy: Default

Grant Access to Session Revocation API

[Cancel](#) [Save](#)

1841

1842

This completes configuration of the OIDC IdP.

5 How to Install and Configure the FIDO UAF Authentication Server

For the lab build environment, the Nok Nok Labs S3 Authentication Suite provides FIDO UAF integration. The S3 Authentication Suite can support a variety of different deployments and architectures, as described in the Solution Guide [17]. This section briefly describes the overall deployment architecture used for this build.

The Nok Nok Labs SDKs can be directly integrated into mobile apps, providing UAF client functionality directly within the app. This deployment would be more suitable to use cases that do not involve federation, where the requirement is to authenticate users directly at the app back-end. Nok Nok Labs also provides “Out-of-Band” (OOB) integration. OOB can support workflows where a mobile device is used for true OOB authentication of logins or transactions initiated on another device, such as a laptop or workstation. OOB also can be used for authentication flows in a mobile web browser, including OAuth authorization flows or IdP authentication, as implemented in this build by using the AppAuth pattern.

When OOB is used in a cross-device scenario, the user must first register the mobile device by scanning a QR code displayed in the browser. Subsequent authentication requests can be sent by push notification to the registered device. When the OOB flow is initiated in a mobile browser, however, the authentication request can be sent directly to the app running the Nok Nok Labs SDK by using mobile platform technologies to open links directly in mobile apps (*App Links* for Android, or *Universal Links* for iOS). The FIDO client that processes the OOB authentication request can be either a custom app incorporating the Nok Nok Labs SDK, or the Nok Nok Labs Passport app, which provides a ready-made implementation.

The components of the Nok Nok Labs deployment for this build architecture are as follows:

- Nok Nok Labs Passport – provides UAF client functionality as well as Authenticator-Specific Modules (ASMs) and authenticators on the mobile device
- Nok Nok Labs PingFederate UAF Adapter – a PingFederate plugin providing integration between a PingFederate AS or IdP and the NNAS, enabling UAF authentication or transaction verification to be integrated into PingFederate authentication policies
- NNAS – provides core UAF server functionality, including the generation and verification of challenges, as well as APIs for interactions with UAF clients and the PingFederate Adapter
- Nok Nok Labs Gateway – provides a simplified interface to request FIDO operations from the Authentication Server, as well as integration with the existing app session management infrastructure
- Nok Nok Labs Gateway Tutorial App – a demonstration web app implementation that provides simple U2F and UAF authentication and registration workflows

1877 In a typical production implementation, the gateway functions for authenticator management
1878 (registration and de-registration) would typically require strong authentication, implemented through
1879 the Gateway's session management integration. Nok Nok Labs' documentation for the PingFederate
1880 plugin provides examples for defining a "reg" OAuth scope to request authenticator registration. An
1881 OAuth Scope Authentication Selector could be used in a PingFederate authentication policy to trigger
1882 the required strong authentication process.

1883 **5.1 Platform and System Requirements**

1884 The following subsections list the hardware, software, and network requirements for the various Nok
1885 Nok Labs components.

1886 **5.1.1 Hardware Requirements**

1887 Nok Nok Labs specifies the following minimum hardware requirements for the NNAS and Nok Nok Labs
1888 Gateway components. The requirements for acceptable performance will depend on the anticipated
1889 user population and server load. See the *Enabling Scalability & Availability* section of the *Solution Guide*
1890 for architecture guidance on deploying the NNAS in a clustered configuration.

- 1891 ▪ Processor: 1 CPU
- 1892 ▪ Memory: 4 GB RAM
- 1893 ▪ Hard disk drive size: 10 GB

1894 **5.1.2 Software Requirements**

1895 Complete software requirements for the NNAS are provided in the *Nok Nok Labs Authentication Server*
1896 *Administration Guide* [18]. The major requirements are summarized below:

- 1897 ▪ OS: Red Hat Enterprise Linux 7 or CentOS 7
 - 1898 ▪ Relational database system: MySQL 5.7.10 or later versions, Oracle Database 12c, or PostgreSQL
1899 9.2 or 9.4
 - 1900 ▪ Application server: Apache Tomcat 8.0.x or 8.5.x
 - 1901 ▪ Java: Oracle JDK Version 8
 - 1902 ▪ Build tool: Apache Ant 1.7 or later versions
 - 1903 ▪ For clustered deployments: Redis 2.8 or later versions
 - 1904 ▪ Google Cloud Messenger (GCM) or Apple Push Notification System (APNS), if using push
1905 messages
- 1906 The Nok Nok Labs PingFederate Adapter is compatible with PingFederate 8.1.3 or later versions.
- 1907 The Nok Nok Labs Gateway is also deployed in Tomcat.

5.2 How to Install and Configure the FIDO UAF Authentication Server

The installation process for the Authentication Server is documented in the *Administration Guide*. A high-level summary is provided below, with notes relevant to the lab build:

- Install the OS and dependent software, including Java and Tomcat. The database can be installed on the same host as Tomcat, or remotely. Provision a TLS certificate for the server, and configure Tomcat to use TLS.
- The configuration for push notifications to support OOB authentication is not required for this build; push notifications would be used when the mobile device is used to authenticate logins or transactions initiated on a separate device.
- Follow the instructions to generate an encryption key, and encrypt database credentials in the installation script. Encrypting the push notification credentials is not required, unless that functionality will be used.
- For this lab build, the standalone installation was used. The standalone option uses the PostgreSQL database on the same host as the Authentication Server, and also installs the Tutorial app.
- After running the installation script, delete the encryption key (`NNL_ENCRYPTION_KEY_BASE64`) from `nnl-install-conf.sh`.
- For this lab build, the default policies and authenticators were used. In a production deployment, policies could be defined to control the authenticator types that could be registered and used to authenticate.
- Provisioning a Facet ID is not necessary for the OOB integration with Nok Nok Labs Passport, as used in the lab. If the Nok Nok Labs SDK were integrated with a custom mobile app, then the Facet ID would need to be configured, and the `facets.uaf` file would need to be published at a URL where it is accessible to clients.
- App link/universal link integration (optional) – In the lab, the default setting using an app link under <https://app.noknok.com> was used. This is acceptable for testing, but in a production deployment, an app link pointing to the IdP's actual domain name would typically be used. It should be noted that the FQDN for the app link must be different from the authentication endpoint (i.e., the IdP's URL) at least by sub-domain.
- Configure tenant-specific and global parameters. For the lab build, a single tenant was used. Many parameters can be left at the default settings. Some notes on specific parameters are provided below:
 - `uaf.application.id` – This should be a URL that is accessible to clients. In a production deployment, the AS may not be accessible, so this may need to be hosted on a different server.

- 1943 • `uaf.facet.id` – There is no need to modify the Facet ID setting to enable the use of the
1944 Passport app for OOB authentication; however, if other custom apps were directly
1945 integrating the Nok Nok Labs SDK, they would need to be added here.
- 1946 ▪ For a production deployment, client certificate authentication to the Authentication Server
1947 should be enabled. This is done by configuring the Tomcat HTTP connector to require client
1948 certificates. This requires provisioning a client certificate for the gateway (and any other servers
1949 that need to call the Nok Nok Labs APIs). See the notes in Section 5.3 of the *Administration*
1950 *Guide* about configuring the Gateway to use client certificate authentication. A general
1951 reference on configuring TLS in Tomcat 8 can be found at <https://tomcat.apache.org/tomcat-8.0-doc/ssl-howto.html>.
- 1952

1953 **5.3 How to Install and Configure the FIDO UAF Gateway Server**

1954 The Nok Nok Labs Gateway app is delivered as a Web Archive (WAR) file that can be deployed to a
1955 Tomcat server. For the lab build, it was deployed on the same server as the NNAS.

1956 Configure the required settings in the `nnlgateway.properties` file, including the settings listed below:

- 1957 ▪ `mfas_location` – NNAS URL
1958 ▪ `server.auth.enabled` – should be set to true; also requires configuring the trust-store settings
1959 ▪ `client.auth.enabled` – see notes in Section 5.2 above; should be enabled for strong client
1960 authentication in production deployments; also requires configuring the keystore settings

1961 In addition, the Gateway Tutorial app was installed by deploying the `gwtutorial.war` file and
1962 configuring the required URLs in `gwtutorial.properties`.

1963 **5.4 How to Install and Configure the FIDO UAF Adapter for the OAuth 2 AS**

1964 Nok Nok Labs provided a tar file containing a set of software tools for integration and testing with
1965 PingFederate. Version 5.1.0.501 of the Ping Integration library was used for the lab build. The
1966 installation process is summarized below; refer to the *Nok Nok PingFederate Adapter Integration Guide*
1967 [\[19\]](#) for full details:

- 1968 1. Extract the `adapter` folder from the `nnl-ping-integration-5.1.0.501.tar` file onto the PingFederate
1969 server where the adapter will be installed.
- 1970 2. Stop PingFederate if it is running, and run the installation script. The path to the PingFederate
1971 installation is passed as an argument; run the script by using an account with write access to the
1972 PingFederate installation:
1973

```
$ ./adapter-deploy.sh /usr/share/pingfederate-8.2.2/pingfederate
```
- 1974 3. Configure the `adapter.properties` file (located in the PingFederate directory under
1975 `server/default/conf`) as required for the server and client TLS authentication settings specified

earlier in the Authentication Server configuration. If push notifications are enabled, configure the relevant settings.

4. The *Configure Session Manager* and *Deploy Nok Nok Gateway OOB* sections of the *Integration Guide* provide settings to use PingFederate to protect the Registration endpoint on the Nok Nok Labs Gateway. This could be used in conjunction with the custom “reg” scope and a PingFederate authentication policy to require strong authentication prior to UAF authenticator registration. This configuration was not tested in the lab.

The *Configure PingFederate Console* section of the *Integration Guide* walks through the complete configuration of a PingFederate OIDC provider. See [Section 4.3](#) of this guide for the procedure to configure the OpenID Provider.

6 How to Install and Configure the FIDO U2F Authentication Server

The SKCE from StrongAuth performs the FIDO U2F server functionality in the build architecture. StrongAuth’s main product is the StrongAuth Key Appliance, but the company also distributes much of its software under the *Lesser General Public License (LGPL)*, published by the Free Software Foundation. SKCE 2.0 Build 163 was downloaded from its repository on *Sourceforge* and was used for this build. For more information, documentation, and download links, visit the vendor’s site at <https://www.strongauth.com/products/foss>.

6.1 Platform and System Requirements

The following subsections document the software, hardware, and network requirements for SKCE 2.0.

6.1.1 Software Requirements

StrongAuth’s website lists the OSs on which SKCE has been tested:

- CentOS 6.X or 7.X, 64-bit
- Windows 7 Professional, 64-bit

Since SKCE is a Java app, in theory it should be able to run on any OS that supports a compatible version of Java and the other required software. The app was built with the Oracle JDK Version 8, Update 72. For this build, SKCE was installed on a CentOS 7.4 server; therefore, these steps assume a Linux installation.

SKCE can be installed manually or with an installation script included in the download. SKCE depends on other software components, including an SQL database, an LDAP directory server, and the Glassfish Java app server. By default, the script will install MariaDB, OpenDJ, and Glassfish all on a single server. SKCE can also utilize AD for LDAP.

2007 For this build, the scripted installation was used with the default software components. The required
2008 software components, which are listed below, must be downloaded prior to running the installation
2009 script:

- 2010 ■ Glassfish 4.1
2011 ■ Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 8
2012 ■ JDK 8, Update 121
2013 ■ OpenDJ 3.0.0
2014 ■ MariaDB 10.1.22
2015 ■ MariaDB Java Client

2016 See StrongAuth's scripted installation instructions for details and download links:
2017 <https://sourceforge.net/p/skce/wiki/Install%20StrongAuth%20CryptoEngine%202.0%20%28Build%2016%29%20scripted/>.

2019 To download OpenDJ, you must register for a free account for *ForgeRock BackStage*.

2020 SKCE can also utilize an AD LDAP service. The LDAP directory contains system user accounts for
2021 managing the SKCE (generating cryptographic keys, etc.) Data pertaining to registered users and
2022 authenticators is stored in the SQL database, not in LDAP.

2023 **6.1.2 Hardware Requirements**

2024 StrongAuth recommends installing SKCE on a server with at least 10 GB of available disk space and 4 GB
2025 of RAM.

2026 **6.1.3 Network Requirements**

2027 The SKCE API is hosted on Transmission Control Protocol (TCP) Port 8181. Any apps that request U2F
2028 registration, authentication, or deregistration actions from the SKCE need to be able to connect on this
2029 port. Glassfish runs an HTTPS service on this port. Use firewall-cmd, iptables, or any other system utility
2030 for manipulating the firewall to open this port.

2031 Other network services listen on the ports listed below. For the scripted installation, where all these
2032 services are installed on a single server, there is no need to adjust firewall rules for these services
2033 because they are only accessed from localhost.

- 2034 ■ 3306 – MariaDB listener
2035 ■ 4848 – Glassfish administrative console
2036 ■ 1389 – OpenDJ LDAP service

2037 6.2 How to Install and Configure the FIDO U2F Authentication Server

2038 StrongAuth's scripted installation process is documented at
2039 <https://sourceforge.net/p/skce/wiki/Install%20StrongAuth%20CryptoEngine%202.0%20%28Build%2016%29%20scripted/>.

2041 The installation procedure consists of the following steps:

- 2042 ■ Downloading the software dependencies to the server where SKCE will be installed
- 2043 ■ Making any required changes to the installation script
- 2044 ■ Running the script as root/administrator
- 2045 ■ Performing post-installation configuration

2046 The installation script creates a “strongauth” Linux user and installs all software under
2047 /*usr/local/strongauth*. Rather than reproduce the installation steps here, this section provides some
2048 notes on the installation procedure:

- 2049 1. Download the software: Download and unzip the SKCE build to a directory on the server where
2050 SKCE is being installed. Download all installers as directed in the SKCE instructions to the same
2051 directory.
- 2052 2. Change software versions as required in the install script: If different versions of any of the
2053 software dependencies were downloaded, update the file names in the install script (*install-skce.sh*). Using different versions of the dependencies, apart from minor point-release versions,
2054 is not recommended. For the lab build, JDK Version 8u151 was used instead of the version
2055 referenced in the instructions. This required updating the `JDK` and `JDKVER` settings in the file.
- 2057 3. Change passwords in the install script: Changing the default passwords in the delivered script is
2058 strongly recommended. The defaults are readily discoverable, as they are distributed with the
2059 software. Passwords should be stored in a password vault or other agency-approved secure
2060 storage. Once the installation script has been run successfully, the script should be deleted or
2061 sanitized to remove passwords. The following lines in the install script contain passwords:

```
2062 LINUX_PASSWORD=ShaZam123          # For 'strongauth' account
2063 GLASSFISH_PASSWORD=adminadmin      # Glassfish Admin password
2064 MYSQL_ROOT_PASSWORD=BigKahuna    # MySQL 'root' password
2065 MYSQL_PASSWORD=AbracaDabra       # MySQL 'skles' password
2066 SKCE_SERVICE_PASS=Abcd1234!      # Webservice user 'service-cc-ce' password
2067 SAKA_PASS=Abcd1234!
2068 SERVICE_LDAP_BIND_PASS=Abcd1234!
```

- 2069 SEARCH_LDAP_BIND_PASS=Abcd1234 !
- 2070 4. Set the App ID URL: The App ID setting in *install-skce.sh* should point to a URL that will be
2071 accessible to clients where the *app.json* file can be downloaded. The default location is a URL on
2072 the SKCE server, but the SKCE would not be exposed to mobile clients in a typical production
2073 deployment. In the lab, *app.json* was hosted on the PingFederate server hosting the IdP in the
2074 following location:
- 2075 */usr/share/pingfederate-8.3.2/pingfederate/server/default/conf/template/assets/scripts*
- 2076 which enables the file to be accessed by clients at the following URL:
2077 <https://oidp1.slpsd.mso:9031/assets/scripts/app.json>.
- 2078 5. Run the script: *install-skce.sh* must be run as the root user. If the install script terminates with an
2079 error, troubleshoot and correct any problems before continuing.
- 2080 6. (For CentOS 7) create firewall rule: The install script attempts to open the required port using
2081 iptables, which does not work on CentOS 7. In that case, the following commands will open the
2082 port:
- 2083 # firewall-cmd --permanent --add-port 8181/tcp
2084 success
2085 # firewall-cmd --reload
2086 success
- 2087 7. Install additional libraries: Depending on how CentOS was installed, some additional libraries
2088 may be required to run the graphical key custodian setup tool. In the lab, the SKCE server did
2089 not include X11 or a graphical desktop, so the key custodian setup was run over Secure Shell
2090 (SSH) with X11 forwarding. To install additional libraries needed for this setup, run the following
2091 commands:
- 2092 # yum install libXrender
2093 # yum install libXtst
- 2094 Note that running the graphical configuration tool over SSH also requires configuring X11
2095 forwarding in the SSH daemon (**sshd**) on the server, and using the **-X** command line option
2096 when connecting from an SSH client.
- 2097 8. Run the key custodian setup tool: In production deployments, the use of a Hardware Security
2098 Module (HSM) and Universal Serial Bus (USB) drive for the security officer and key custodian
2099 credentials is strongly recommended. In the lab, the software security module was used. Also,
2100 the lab setup utilized a single SKCE server; in this case, all instructions pertaining to copying keys
2101 to a secondary appliance can be ignored.

2102 9. Restart Glassfish: On CentOS 7, run the following command:

2103 \$ sudo systemctl restart glassfishd

2104 10. Complete Step 3b in the SKCE installation instructions to activate the cryptographic module.

2105 11. Complete Step 3c in the SKCE installation instructions to create the domain signing key. When
2106 prompted for the App ID, use the URL referenced above in the App ID setting of the *install-*
2107 *skce.sh* script.

2108 12. Complete Step 4 if you are installing secondary SKCE instances; this was not done for this build,
2109 but is recommended for a production installation.

2110 13. Install a TLS certificate (optional): The SKCE installation script creates a self-signed certificate for
2111 the SKCE. It is possible to use the self-signed certificate, though PingFederate and any other
2112 servers that integrate with the SKCE would need to be configured to trust it. However, many
2113 organizations will have their own CAs, and will want to generate a trusted certificate for the
2114 SKCE for production use. To generate and install the certificate, follow the steps listed below:

2115 a. The keystore used by the SKCE Glassfish server is listed below:

2116 /usr/local/strongauth/glassfish4/glassfish/domains/domain1/config/keystor
2117 e.jks

2118 b. The default password for the keystore is “changeit”.

2119 c. Use keytool to generate a keypair and certificate signing request. For example, the fol-
2120 lowing commands generate a 2048-bit key pair with the alias “mss0,” and export a Cer-
2121 tificate Signing Request (CSR):

2122 \$ keytool -genkeypair -keyalg RSA -keysize 2048 -alias mss0 -keystore
2123 keystore.jks

2124 \$ keytool -certreq -alias mss0 -file strongauth.req -keystore
2125 keystore.jks

2126 d. Submit the CSR to your organization’s CA, and import the signed certificate along with
2127 the root and any intermediates:

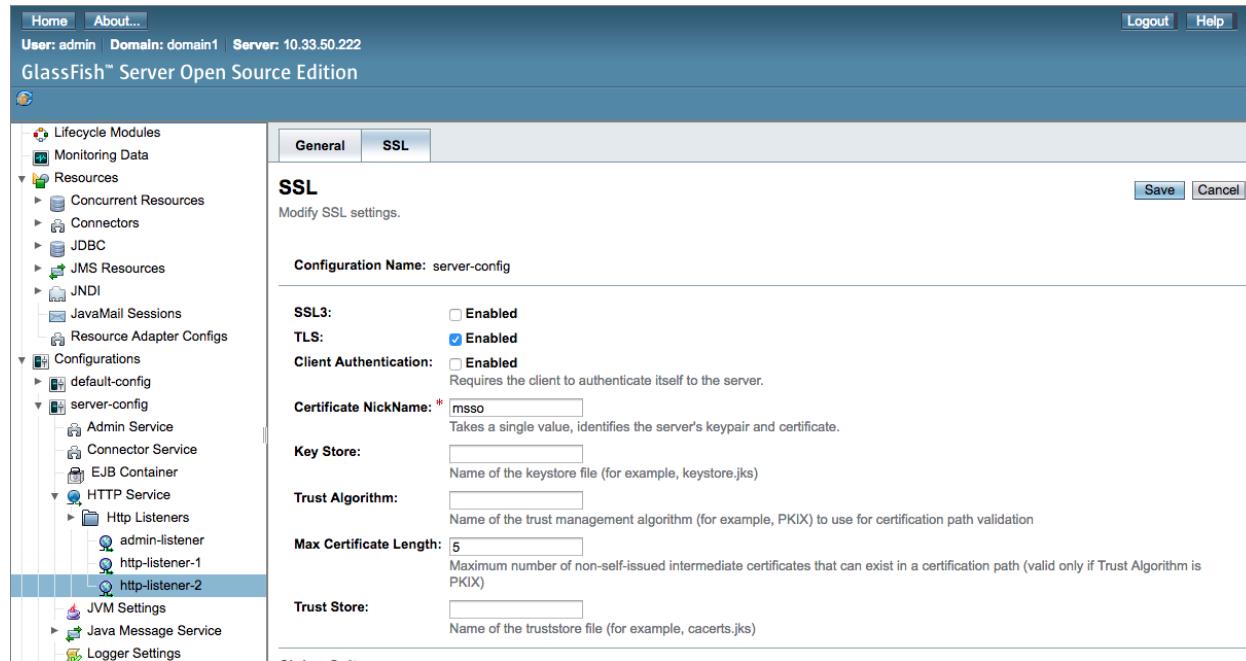
2128 \$ keytool -import -trustcacerts -alias mss0-root -file lab-certs/root.pem
2129 -keystore keystore.jks

2130 \$ keytool -import -alias mss0 -file lab-certs/strongauth.lpsd.mss0.cer -
2131 -keystore keystore.jks

2132 e. To configure the SKCE to use the new certificate, log into the Glassfish administrative
2133 console on the SKCE server. The console runs on Port 4848; the username is “admin,”
2134 and the password will be whatever was configured for `GLASSFISH_PASSWORD` in the
2135 *install-skce.sh* script.

- 2136 i. Navigate to *Configurations, server-config, HTTP Service, Http Listeners, http-*
 2137 *listener-2*, as shown in Figure 6-1. On the **SSL** tab, set the **Certificate NickName**
 2138 to the alias that was created with the “keytool -genkeypair” command above.

2139 **Figure 6-1 Glassfish SSL Settings**



- 2140 f. Click **Save**, and then restart glassfish. If logged on as the glassfish user, run the following command:
- 2141

```
$ sudo service glassfishd restart
```
- 2142 g. In a browser, access the SKCE web service on Port 8181, and ensure that it is using the newly created certificate.
- 2143 h. For the FIDO Engine tests below to complete successfully, the main CA trust store for the JDK will need to be updated with your organization’s CA certificate. This can also be done with keytool:
- 2144

```
$ keytool -import -trustcacerts -file lab-certs/root.pem -keystore $JAVA_HOME/jre/lib/security/cacerts
```
- 2145 14. Test the FIDO Engine: Follow the testing instructions under Step D at the following URL:
 2146 <https://sourceforge.net/p/skce/wiki/Test%20SKCE%202.0%20using%20a%20client%20program%20-%20Build%20163/>.

2154 There are additional tests on that web page to test the other cryptographic functions of the
2155 SKCE; however, only the FIDO Engine tests are critical for this build.

2156 If the FIDO Engine tests are completed without errors, proceed to Section 6.3 to integrate the SKCE with
2157 the IdP. If any errors are encountered, the Glassfish log file (located at
2158 `/usr/local/strongauth/glassfish4/glassfish/domains/domain1/logs/server.log`) should contain messages
2159 to aid in troubleshooting.

2160 **6.3 How to Install and Configure the FIDO U2F Adapter for the IdP**

2161 To incorporate FIDO U2F authentication into a login flow at the IdP, some integration is needed to
2162 enable the IdP to call the SKCE APIs. In the lab build architecture, FIDO U2F authentication was
2163 integrated into a SAML IdP. PingFederate has a plugin architecture that enables the use of custom and
2164 third-party adapters in the authentication flow. StrongAuth provides a PingFederate plugin to enable
2165 PingFederate IdPs (or AS) to support U2F authentication. This section describes the installation of the
2166 plugin on a PingFederate server. For details on how to integrate U2F authentication to a login flow, see
2167 [Section 4.2.1.3](#).

2168 The StrongAuth plugin for PingFederate is delivered in a zip file containing documentation and all of the
2169 required program files.

- 2170 1. To begin the installation process, upload the zip file to the PingFederate server where the
2171 StrongAuth plugin will be installed, and unzip the files.
- 2172 2. If Apache Ant is not already installed on the server, install it now by using the server's package
2173 manager. For CentOS, this can be done by running the following command:

2174 # yum install ant
- 2175 3. Once Apache Ant is installed, follow the "Installation" instructions in the *StrongAuth – Ping*
2176 *Federate FIDO IdP Adapter Installation Guide* [\[20\]](#), which consist of copying the plugin files to
2177 the required directories in the PingFederate installation, and running *build.sh*. If the script runs
2178 successfully, it will build the plugin using Ant and restart PingFederate.
- 2179 4. Follow the steps in "Table 2: Configure the SKCE" in the *Installation Guide*. For this build, the
2180 *app.json* file needs to be copied to a browser-accessible location on the PingFederate server
2181 where the plugin is being installed. In the lab, we placed it under the following location:

2182 `/usr/share/pingfederate-8.3.2/pingfederate/server/default/conf/template/assets/scripts`
- 2183 5. This enables the *app.json* to be accessed at the URL
2184 `https://idp1.spsd.mss:9031/assets/scripts/app.json`. Note that Steps 4 and 5 in Table 2 of the
2185 *Installation Guide* are only required if the SKCE is using the default self-signed certificate; if a
2186 trusted certificate was installed as described in [Section 6.2](#), then those steps can be skipped.

- 2187 6. Download the JQuery 2.2.0 library at the URL below, and save it to the scripts folder referenced
2188 above: <https://code.jquery.com/jquery-2.2.0.min.js>.
- 2189 7. Follow the steps in “Table 3: Configure the Ping Federate Instance” in the *Installation Guide*.
2190 Importing the SKCE self-signed certificate is not required if a trusted certificate was created.
2191 Installation of the JCE unlimited policy was described in the PingFederate installation
2192 instructions in [Section 3](#), so that too can be skipped at this point, if it has already been done.
2193 Steps 7–9 should be completed in any case.
- 2194 8. Follow the steps in “Table 4: Configuring the FIDO Adapter” in the *Installation Guide*. In Step 5,
2195 the Domain ID typically should be set to “1,” unless you have defined multiple domains in the
2196 SKCE. For the username and password, use the values configured earlier in *install-skce.sh*.
- 2197 9. “Table 5: Ping Federate OAuth Configuration Steps” in the *Installation Guide* provides an
2198 example of how to incorporate U2F into a login flow, along with username/password form login,
2199 by creating a composite adapter that includes the login form and U2F adapters, and using a
2200 selector to activate the composite adapter whenever an OAuth authorization request includes
2201 the scope value “Idap.” Alternatively, the individual adapters can be called directly in an
2202 authentication policy. See Chapter 4 of the *Installation Guide* for additional examples of using
2203 U2F in authentication policies.

2204 6.3.1 FIDO U2F Registration in Production

2205 By default, the StrongAuth Ping plugin enables the registration of U2F authenticators. In production, an
2206 authorized registration process should be established to provide adequate assurance in the binding of
2207 the authenticator to a claimed identity. If the FIDO adapter is accessible after single-factor password
2208 authentication, organizations may want to disable the registration functionality. See Section B.5 in
2209 Volume B of this guide for a discussion of FIDO enrollment.

2210 7 Functional Tests

2211 The MSSO architecture has a number of interoperate components, which can make troubleshooting
2212 difficult. This section describes tests than can be performed to validate that individual components are
2213 working as expected. If issues are encountered with the overall SSO flow, these tests may help identify
2214 the problem area.

2215 7.1 Testing FIDO Authenticators

2216 The FIDO Alliance implements a Functional Certification Program, in which products are evaluated for
2217 conformance to the UAF and U2F specifications. Purchasing FIDO-certified authenticators can help avoid
2218 potential authenticator implementation issues. Information on the certification program is available at
2219 <https://fidoalliance.org/certification/>, and the FIDO Alliance website also lists certified products.

2220 Some resources are available to help troubleshoot individual authenticators:

- 2221 ▪ The Yubico demonstration site provides an interface for testing registration and authentication
2222 with U2F authenticators: <https://demo.yubico.com/u2f>.
- 2223 ▪ The Nok Nok Labs Gateway Tutorial App supports testing of the registration, authentication, and
2224 transaction verification functions of FIDO UAF authenticators.

2225 **7.2 Testing FIDO Servers**

2226 The StrongAuth SKCE documentation includes instructions on testing U2F authenticator registration,
2227 authentication, de-registration, and other functions. See Step 14 in [Section 6.2](#).

2228 To test the NNAS, Nok Nok Labs provides the OnRamp mobile app in the Google Play Store and the
2229 Apple App Store to test the server APIs with UAF authenticators.

2230 **7.3 Testing IdPs**

2231 If federated authentication is failing, the issue may lie at the IdP or the AS. The PingFederate server log
2232 (located by default under `<pingfederate-directory>/log/server.log`), on both ends, should provide
2233 relevant messages.

2234 In some cases, it may be beneficial to look at the assertions being issued by the IdP and to check for the
2235 expected attributes. This could be done by integrating a demonstration app as a federation client and
2236 debugging the data returned in the assertion. For SAML, projects like SimpleSAMLphp
2237 (<https://simplesamlphp.org/>) provide an implementation that is easy to deploy. It is also possible to
2238 perform this testing without installing additional tools.

2239 One method for SAML is to use Chrome Remote Debugging for Android devices:
2240 <https://developers.google.com/web/tools/chrome-devtools/remote-debugging/>.

2241 By logging the authentication flow in the Network pane of Chrome's developer tools, the SAML response
2242 can be extracted and viewed. The authentication flow with the SAML IdP configured in this practice
2243 guide consists of a series of calls to the `SSO.ping` URL at the IdP. Because the SAML POST binding is used,
2244 the final `SSO.ping` response includes an HTML form that submits the SAML response back to the AS. The
2245 SAML response can be found in an input element in the page content:

```
2246 <input type="hidden" name="SAMLResponse"
2247 value="PHNhbWxwO1J1c3BvbnN1IFZlcnPb249IjIuMCIGSUQ9Ik01T2xNN1ZxZW51VnpBU2doSH1sakFLY1I
2248 uOCIGSXNdWVJbnN0YW50PSIyMDE3LTEzVDEzOjQ5OjE3LjEwMFoiIEluUmVzcG9uc2VUbz0iS2RwMXVfZ
2249 HFPMH1NX2Z0YWV1dWJnRj1vMFByIIiBEZXNOaW5hdGlvbj0iaHR0cHM6Ly9pZG0uc2FuZGJveC5tb3Rvcm9sYXN
2250 vbHV0aW9ucy5jb20vc3AvQUNTLnNhbWwyIiB4bWxucpzYW1scD0idXJuOm9hc21zOm5hbWVzOnRj0lNBTUw6M
2251 i4wOnByb3RvY29sIj48c2FtbDpjC3N1ZXIgeG1sbnM6c2FtbD0idXJuOm9hc21zOm5hbWVzOnRj0lNBTUw6Mi4
2252 wOmFzc2VydGlvbiI+aWRwMS5zchNkLm1zcz288L3NhbWw6SXNzdWVypjxkczpTaWduYXR1cmUgeG1sbnM6ZHM9I
2253 mh0dHA6Ly93d3cudzMub3JnLzIwMDAvMDkveG1sZHNpZyMiPgo8ZHM6U21nbmVksW5mbz4KPGRzOkNhbm9uaWN
2254 hbG16YXRpb25NZXR0b2QgQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzEwL3htbC1leGMTYzE0b
```

2255 iMiLz4KPGRzO1NpZ25hdHVyZU1ldGhvZCBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG1
 2256 sZHNPzyltb3J1I3JzYS1zaGEyNTYiLz4KPGRzO1J1ZmVyZW5jZSBVUkk9IiNKNU9sTTZWcWVuZVZ6QVNnaEh5b
 2257 GpBS2JSLjgiPgo8ZHM6VHJhbnNmb3Jtcz4KPGRzO1RyYW5zZm9ybSBBbGdvcml0aG09Imh0dHA6Ly93d3cudzM
 2258 ub3JnLzIwMDAvMDkveG1sZHNpZyN1bnZ1bG9wZWQtc21nbmF0dXJ1Ii8+CjxkczpUcmFuc2Zvcm0gQWxnb3Jpd
 2259 GhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzEwL3htbC1leGMtYzE0biMiLz4KPC9kczpUcmFuc2Zvcm1zPgo
 2260 8ZHM6RGlnZXN0TWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8wNC94bWx1bmMjc2hhM
 2261 jU2Ii8+CjxkczpEaWdlc3RWYWyx1ZT4xd1FpcUNVNmlZYTMzd1FtKzcxbEVsVm1pUUh6T2U5cytBTTdQYTk4Vlp
 2262 BPTwvZHM6RGlnZXN0VmFsdWU+CjwvZHM6UmVmZXJ1bmNlPgo8L2RzO1NpZ251ZE1uZm8+CjxkczpTaWduYXR1c
 2263 mVWYWyx1ZT4KTHpSbUJhc1k2bndGS3Zyjdjtl29WYWNJSWRJRUY4eUloV0JXT0NHZ3pyMWtONGVzVi9CU31LQ1N
 2264 XYjhKU1h3QzhWRHNNunRXOENMNQpVRFV0NTV1OXRca05Wanh2NWR0NSTOYXQ5eWtmdnhXbu9kcGVJVTBzMXNuM
 2265 UJHdytkOTROZU1CYVdJWE1ZOViRaD1nV3Q2S1100VFhCmRGdDzrRUY1S1NDS1FBQVN1bTEyT2xLV29GK2JSbG1
 2266 HNGVsbtVMTh1N0E3Wi9hRnZ1cDNDNmV5ZEpwK1IxaStaK0F6NH1XdmMvnMekYn1LMTBPZ05pLzBibnprazd3L
 2267 0psdHk0Z1VEcVd6bxJyRFpwSEJ4ZkFMVW5UV2RPVDVJe ko3bmqMqwtBYVN0NDYwWjUyblpBOGFBygpVbzA4T0t
 2268 EYnZVaS9UZ2xTcUzjcdJSYStCae9DbUR30WJvTG9udz09CjwvZHM6U21nbmF0dXJ1VmFsdWU+CjwvZHM6U21nb
 2269 mF0dXJ1PjxzYW1scDptDGF0dXM+PHNhbwXw01N0YXR1c0NvZGUgVmFsdWU9InVybjpvyXNpczpuYW1lczp0Yzp
 2270 TQU1MojIuMDpzdGF0dXM6U3VjY2VzcyIvpjwvc2FtbHA6U3RhdHVzPjxzYW1sOkFzc2VydG1vbiBJRD0iSF9tL
 2271 1dIR29UVUBELjNjV1A0MvhDVvh4YkdLIiBjC3N1ZU1uc3RhbnQ9IjIwMtctMTEtMTNUMTM6NDk6MTcuMTU1WiI
 2272 gVmVyc21vbj0iMi4wiB4bWxuczpZYw1sPSJ1cm46b2FzaXM6bmFtZXM6dGM6U0FNTDoyLjA6YXNzZXJ0aW9uI
 2273 j48c2FtbDpJc3N1ZXi+aWrwMS5zcHNkLm1zc288L3NhbWw6SXNzdWVYpjxzYw1s01N1Ymp1Y3Q+PHNhbw6TmF
 2274 tzuleIEZvcm1hdD0idXJu0m9hc21z0m5hbWVzOnRj01NBTUw6MS4x0m5hbWVpZC1mb3JtYXQ6dW5zcGVjaWzpZ
 2275 WQipnVuY2NvZXrlc3Q0PC9zYW1s0k5hbWVJRD48c2FtbDpTdwJqZWN0Q29uZmlybWF0aW9uIE11dGhvZD0iDxJ
 2276 uOm9hc21z0m5hbWVzOnRj01NBTUw6Mi4wOmNtOmJ1YXJ1ci+PHNhbw6U3ViamVjdeEnvbmZpcm1hdG1vbkRhd
 2277 GEgUmVjaXBpZW50PSJodHRwczovL21kbS5zYW5kYm94Lm1vdG9yb2xhc29sdXRpb25zLmNvbS9zcC9BQ1Muc2F
 2278 tbDIiIE5vdE9uT3JBznRlcj0iMjAxNy0xMS0xM1QxMz01NDoxNy4xNTVaiiBjblJ1c3BvbnN1VG89IktkcDF1X
 2279 2RxTzB5TV9mdGF1ZXViZ0Y5bzBQWCIVpjwvc2FtbDpTdwJqZWN0Q29uZmlybWF0aW9uPjwvc2FtbDpTdwJqZWN
 2280 0PjxzYW1s0kNvbmrpdG1vbnMgTm90QmVmb3J1PSIyMDE3LTExLTEzVDEzojQ00jE3LjE1NvoiIE5vdE9uT3JBZ
 2281 nRlcj0iMjAxNy0xMS0xM1QxMz01NDoxNy4xNTVaij48c2FtbDpBdWRpZW5jZVJ1c3RyaWN0aW9uPjxzYW1sOkF
 2282 1ZG11bmNlPmN0b1BpbmdGZWRfZw50aXR5SUQ8L3NhbWw6QXVkaWVuY2U+PC9zYW1sOkF1ZG11bmN1UmVzdHJpY
 2283 3RpB24+PC9zYW1s0kNvbmrpdG1vbnM+PHNhbw6QXV0aG5TdGF0Zw11bnQgU2Vzc21vbkluZGV4PSJIX20uV0h
 2284 Hb1VRUEQuM2NWUDQxWENVWHhI0s0iIEF1dGhuSw5zdGFudD0iMjAxNy0xMS0xM1QxMz00TOxNy4xNTNaIj48c
 2285 2FtbDpBdXRobkNvbR1eHQ+PHNhbw6QXV0aG5Db250Zxh0Q2xhc3NSZwy+dXJu0m9hc21z0m5hbWVzOnRj01N
 2286 BTUw6Mi4wOmFj0mNsYXNzZXM6dW5zcGVjaWzpZWQ8L3NhbWw6QXV0aG5Db250Zxh0Q2xhc3NSZwy+PC9zYW1sO
 2287 kF1dGhuQ29udGV4dD48L3NhbWw6QXV0aG5TdGF0Zw11bnQ+PHNhbw6QXR0cm1idxr1U3RhdGvtzw50PjxzYW1
 2288 s0kF0dHJpYnV0ZsBOYw11PSJ1aWQjIE5hbWVGb3JtYXQ9InVybjpvyXNpczpuYW1lczp0YzpTQU1MojIuMDphd
 2289 HRybmfTZs1mb3JtYXQ6YmFzaWMiPjxzYW1s0kF0dHJpYnV0ZVZhBHV1IHzaTp0eXB1PSJ4czpzdHJpbmciiHh
 2290 tbG5zOnhzPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxL1hNTFNjaGvtYStIgeG1sbnM6eHNpPSJodHRwOi8vd3d3L
 2291 ncZLm9yZy8yMDAxL1hNTFNjaGvtYStIgeG1sbnN0Yw5jZSI+dW5jY291dGVzdQ8L3NhbWw6QXR0cm1idxr1VmFsdWU
 2292 +PC9zYW1s0kF0dHJpYnV0ZT48c2FtbDpBdHryaWJ1dGUgTmFtZT0ibWFpbCigTmFtZUZvcm1hdD0idXJu0m9hc
 2293 21z0m5hbWVzOnRj01NBTUw6Mi4wOmF0dHJuYw11LWvcm1hdDpiYXNpYyI+PHNhbw6QXR0cm1idxr1VmFsdWU
 2294 geHNpOnR5cGU9InhzOnN0cmluZyIgeG1sbnM6eHM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvWE1MU2NoZW1hI
 2295 iB4bWxuczp4c2k9Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvWE1MU2NoZW1hLWluc3Rhbmn1Ij51bmNjb2V0ZxN
 2296 0NDwvc2FtbDpBdHryawJ1dGVWYwx1ZT48L3NhbWw6QXR0cm1idxr1Pjwvc2FtbDpBdHryaWJ1dGVTdGF0ZW1lb
 2297 nQ+PC9zYW1s0kFzc2VydG1vbj48L3NhbWxw01J1c3BvbnN1Pg=="/>

2298 The “value” string is the base64-encoded SAML response. A few lines of Python can get the SAML
 2299 response into a readable format. In this example, the value above has been saved to a file called
 2300 *samlresp.txt*:

2301 **\$ python**
 2302 Python 2.7.10 (default, Feb 7 2017, 00:08:15)
 2303 [GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin

```

2304 Type "help", "copyright", "credits" or "license" for more information.
2305 >>> import base64
2306 >>> import xml.dom.minidom
2307 >>> respFile = open("samlresp.txt", "r")
2308 >>> respStr = base64.b64decode(respFile.read())
2309 >>> respXml = xml.dom.minidom.parseString(respStr)
2310 >>> print(respXml.toprettyxml())
2311 <?xml version="1.0" ?>
2312 <samlp:Response Destination="https://idm.sandbox.motorolasolutions.com/sp/ACS.saml2"
2313 ID="J501M6VqeneVzASghHyljAKbR.8" InResponseTo="Kdp1u_dq00yM_ftaeueubgF9o0FX"
2314 IssueInstant="2017-11-13T13:49:17.100Z" Version="2.0"
2315 xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
2316     <saml:Issuer
2317     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">idp1.spdsd.msso</saml:Issuer>
2318         <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
2319             <ds:SignedInfo>
2320                 <ds:CanonicalizationMethod
2321                     Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2322
2323
2324             <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-
2325 more#rsa-sha256"/>
2326                 <ds:Reference URI="#J501M6VqeneVzASghHyljAKbR.8">
2327                     <ds:Transforms>
2328                         <ds:Transform
2329                             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
2330                         <ds:Transform
2331                             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2332                         </ds:Transforms>
2333                         <ds:DigestMethod
2334                             Algorithm="http://www.w3.org/2001/04/xmlenc#sha256" />
2335 <ds:DigestValue>1vQiqCU6iYa33vQm+711ElVmIqHzOe9s+AM7Pa98VZA=</ds:DigestValue>
2336             </ds:Reference>
2337         </ds:SignedInfo>
2338         <ds:SignatureValue>
2339 LzRmBarY6nwFKrvv7S/oVacIIIdIEF8yIhWBWOCGgzrlkN4esV/BSyKCSWb8JSXwC8VDsMRtW8CL5
2340 UDUt55u9tBkNVjxv5dt5+Nat9ykfvxWmOdpeIU0s1sn1BGw+d94heIBaWIXMY9YQh9gWt6JYt9Qa
2341 dFt6kEF5KSCKQAASe1201KWoF+bRlmG4elm5LM8u7A7Z/aFvup3C6eydJp+R1i+Z+Az4yWvc/6a
2342 byK10OgNi/0bnzkk7w/Jlty4fUDqWzmrrDZpHBxfALUnTWdOT5IzJ7njLAkAaSt460Z52nZA8aAb
2343 Uo8OKDbvUi/TglSqFcp2Ra+BhOCmdW9boLonw==
2344     </ds:SignatureValue>
2345     </ds:Signature>
2346     <samlp:Status>
2347         <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
2348     </samlp:Status>
2349     <saml:Assertion ID="H_m.WHGoUQPD.3cVP41XCUXxbGK" IssueInstant="2017-11-
2350 13T13:49:17.155Z" Version="2.0" xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
2351         <saml:Issuer>idp1.spdsd.msso</saml:Issuer>
2352         <saml:Subject>
2353             <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
2354 format:unspecified">unccoetest4</saml:NameID>
2355             <saml:SubjectConfirmation>
```

```

2356 Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
2357             <saml:SubjectConfirmationData
2358             InResponseTo="Kdp1u_dq00yM_ftaeeubgF9o0PX" NotOnOrAfter="2017-11-13T13:54:17.155Z"
2359             Recipient="https://idm.sandbox.motorolasolutions.com/sp/ACS.saml2"/>
2360                 </saml:SubjectConfirmation>
2361             </saml:Subject>
2362             <saml:Conditions NotBefore="2017-11-13T13:44:17.155Z" NotOnOrAfter="2017-
2363 11-13T13:54:17.155Z">
2364                 <saml:AudienceRestriction>
2365                 <saml:Audience>ctoPingFed_entityID</saml:Audience>
2366                     </saml:AudienceRestriction>
2367                 </saml:Conditions>
2368                 <saml:AuthnStatement AuthnInstant="2017-11-13T13:49:17.153Z"
2369                 SessionIndex="H_m.WHGoUQPD.3cVP41XCUXxbGK">
2370                     <saml:AuthnContext>
2371             <saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified</saml:AuthnContextClassRef>
2372                     </saml:AuthnContext>
2373             </saml:AuthnStatement>
2374             <saml:AttributeStatement>
2375                 <saml:Attribute Name="uid"
2376                 NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
2377                     <saml:AttributeValue
2378                         xmlns:xs="http://www.w3.org/2001/XMLSchema"
2379                         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2380                         xsi:type="xs:string">unccoetest4</saml:AttributeValue>
2381                     </saml:Attribute>
2382                     <saml:Attribute Name="mail"
2383                 NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
2384                     <saml:AttributeValue
2385                         xmlns:xs="http://www.w3.org/2001/XMLSchema"
2386                         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2387                         xsi:type="xs:string">unccoetest4</saml:AttributeValue>
2388                     </saml:Attribute>
2389                 </saml:AttributeStatement>
2390             </saml:Assertion>
2391         </samlp:Response>
2392     >>>
2393
2394 In the above example, two attributes, uid and mail, are asserted, but the mail attribute does not
2395 contain a valid email address.
2396

```

2397 For OIDC, because the ID Token is retrieved over a back-channel connection between the RP and the
2398 IdP, it cannot be observed in browser traffic. As with SAML, creating a test app is one method of testing,
2399 but manual testing is also possible by using a few software tools:

- 2400 1. Register an OIDC client with a client secret and a redirect URI that points to a nonexistent
2401 server. A redirect URI value like `https://127.0.0.1/test-url` will work, assuming that you do
2402 not have a web server running on your machine. In a desktop browser, submit an authentication
2403 request with a URL like the one listed below:

2404 `https://op1.lpsd.mssso:9031/as/authorization.oauth2?client_id=marktest&response_type=code&`
2405 `scope=openid%20address%20test%20phone%20openid%20profile%20name%20email`

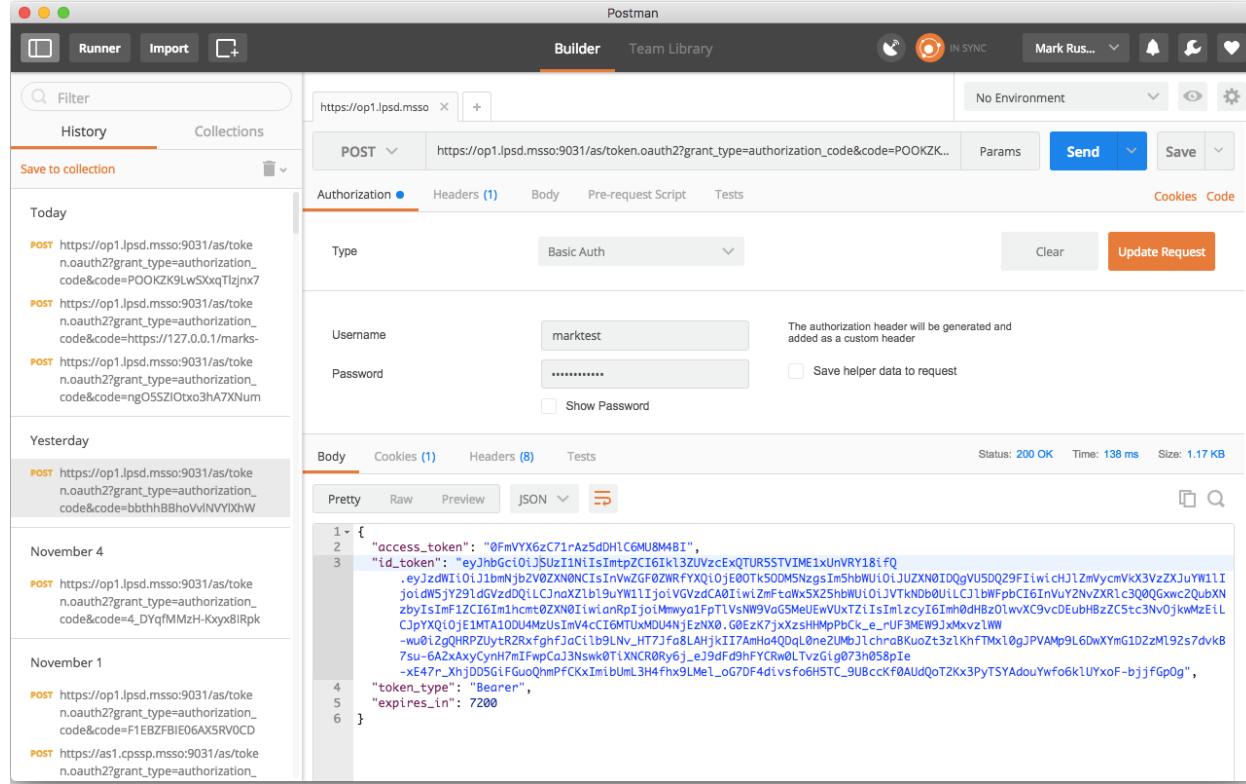
- 2406 2. Replace the server name and client ID with the correct values for your environment; also make
2407 sure that the scope parameter includes `openid` and any other expected scopes. Authenticate to
2408 the IdP. In this case, because the FIDO UAF adapter is in use but is being accessed through a
2409 desktop browser, it initiates an OOB authentication, which can be completed on the mobile
2410 device. Once authentication is completed, the browser will attempt to access the redirect URL,
2411 which will result in a connection error because no web server is running on localhost. However,
2412 the authorization code can be extracted from the URL:

2413 `https://127.0.0.1/test-url?code=lv-pND_3o7_aJ5nFMcD-WbrVENrW7w5V75Cupx9G`

2414 The authorization code can be submitted to the IdP's token endpoint in a POST to obtain the ID Token.
2415 There are numerous ways to do this. Postman is a simple graphical-user-interface tool for testing APIs,
2416 and can be used to submit the request: <https://www.getpostman.com>.

2417 Figure 7-1 shows Postman being used to retrieve an ID Token. A POST request is submitted to the OIDC
2418 IdP's token endpoint; by default, the token endpoint URL is the base URL, followed by `/as/token.oauth2`.
2419 The authorization code is included as a query parameter. The client ID and client secret are used as the
2420 HTTP basic authorization username and password.

2421 Figure 7-1 Using Postman to Obtain the ID Token



2422

2423 The response body is a JSON object, including the ID Token as well as an access token that can be used
2424 to access the userinfo endpoint. As with the SAML assertion, a few lines of Python can render the ID
2425 Token (which is a JWT) into a readable format:

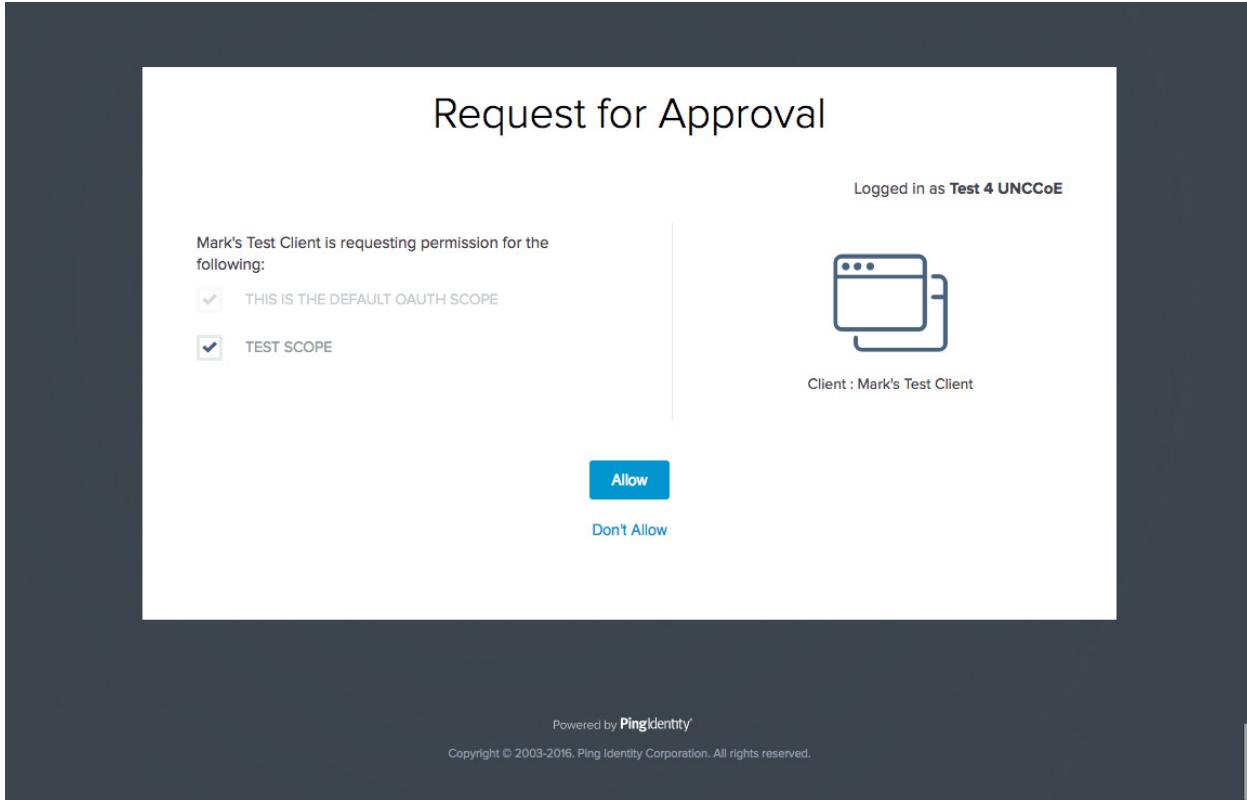
```
2426 $ python
2427 Python 2.7.10 (default, Feb 7 2017, 00:08:15)
2428 [GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
2429 Type "help", "copyright", "credits" or "license" for more information.
2430 >>> import jwt
2431 >>> import json
2432 >>> idTokenStr =
2433 "eyJhbGciOiJSUzIiNlIsImtpZCI6Ik13ZUVzcExQTUR5STVIME1xUnVRY18ifQ.eyJzdWIiOiJ1bmN
2434 jb2V0ZXN0NCIsInVwZGF0ZWRFYXQiOjE0OTk5ODM5NzgsIm5hbWUiOiJUZXN0IDQgVU5DQ29FIiwicH
2435 J1ZmVycmVkJx3VzZXJuYW11Ijoidw5jY291dGVzdDQiLCJnaXZlbl9uYW11IjoiVGVzdCA0IiwiZmFta
2436 Wx5X25hbWUiOjJVTkNDb0UiLCJlbWFpbCI6InVuY2NvZXRLc3Q0QGxwc2QubXNzbyIsImF1ZCI6Im1h
2437 cmt0ZXN0IiwianRpIjoiMnwyaiFpT1VsNW9VaG5MeUEwVUxTZiIsImlzcyI6Imh0dBHzO1wvXC9vcDE
2438 ubHBzzC5tc3NvOjkwmzEiLCJpYXQiOjE1MTA1ODU4MzUsImV4cCI6MTUxMDU4NjEzNX0.G0EzK7jxXz
2439 sHHMpPbCk_e_rUF3MEW9JxMxvz1WW-
2440 wu0i2gQHRPZUytR2RxfgfJaCilb9LNv-HT7Jfa8LAHjkII7AmHa4QDqL0ne2UMbJ1chraBKuoZt3z1
2441 KhfTMx10gJPVAMp9L16dwXYmG1D2zM192s7dvkB7su-
2442 6A2xAxYcynH7m1FwpCaJ3Nswk0TiXNCR0Ry6j_eJ9dFd9hFYCRw0LTvzGig073h058pIe-
2443 xe47r XhjDD5GiFGuoQhmPfCKxImibUmL3H4fhx9LMel oG7DF4divsfo6H5TC 9UBccKf0AUdQoT2K
```

```
2444     x3PyTSYAdouYwfo6k1UYxoF-bjjfGpOg"
2445     >>> idToken = jwt.decode(idTokenStr, verify=False)
2446     >>> print json.dumps(idToken, indent=4)
2447     {
2448         "family_name": "UNCCoE",
2449         "aud": "marktest",
2450         "sub": "unccoetest4",
2451         "iss": "https://op1.lpsd.mssso:9031",
2452         "preferred_username": "unccoetest4",
2453         "updated_at": 1499983978,
2454         "jti": "212kQiNU15oUhnLyA0ULsf",
2455         "given_name": "Test 4",
2456         "exp": 1510586135,
2457         "iat": 1510585835,
2458         "email": "unccoetest4@lpsd.mssso",
2459         "name": "Test 4 UNCCoE"
2460     }
2461     >>>
```

2462 This merely decodes the claims in the JWT without verifying the signature. If there is an issue with
2463 signature validation or trust in the signing key, these errors will be reported in the PingFederate server
2464 log.

2465 7.4 Testing the AS

2466 One simple step that can help identify problems at the AS is turning on the authorization prompts. This
2467 can be done on a per-client basis by deselecting the **BYPASS AUTHORIZATION APPROVAL** setting on the
2468 client configuration page, in the **OAuth Settings** section in the AS console. If the authorization prompt is
2469 displayed (Figure 7-2), this demonstrates that authentication has succeeded, and the list of scopes being
2470 requested by the client is displayed and can be verified.

2471 **Figure 7-2 Authorization Prompt**

2472

2473 It is also possible to manually obtain an access token by using the same procedure that was used in the
2474 previous section to obtain an ID Token; the only difference is that an OAuth request typically would not
2475 include the `openid` scope. If the issued access token is JWT, it can be analyzed using Python as described
2476 above.

2477 If the token is not a JWT (i.e., a Reference Token management scheme is in use), the access token can be
2478 submitted to the AS's introspection endpoint as specified in RFC 7662 [21]. The default location of the
2479 introspection endpoint for PingFederate is the base URL, followed by `/as/introspect.oauth2`. The request
2480 is submitted as a POST, with the access token in a query parameter called **token**. Basic authentication
2481 can be used with the client ID and secret as a username and password. The client must be authorized to
2482 call the introspection endpoint by selecting **Access Token Validation (Client is a Resource Server)** under
2483 **Allowed Grant Types** in the client configuration on the AS.

2484 Figure 7-3 shows a token introspection request and response in Postman.

2485 Figure 7-3 Token Introspection Request and Response

The screenshot shows the Postman application interface. In the left sidebar, there is a history of requests. The main area shows a POST request to <https://as1.cpssp.mss:9031/as/introspect.oauth2?token=9Lkun0IT0x71ZTaqHvtLo4bVD6l>. The Authorization tab is selected, showing basic auth with 'Username' as 'marktest' and 'Password' as '*****'. The Body tab shows a JSON response:

```

1 - {
2   "sub": "uncoetest4",
3   "scope": "testScope",
4   "active": true,
5   "token_type": "Bearer",
6   "exp": 1510597178,
7   "client_id": "marktest",
8   "username": "uncoetest4"
9 }

```

The status bar at the bottom indicates a 200 OK response with 98 ms time and 438 B size.

2486

2487 7.5 Testing the Application

2488 One last potential problem area in this SSO architecture is the back-end app, which must accept and
 2489 validate access tokens. Troubleshooting methods there will depend on the design of the app. Building
 2490 robust instrumentation and error reporting into RP apps will help identify problems. If the app validates
 2491 JWT access tokens, then establishing and maintaining trust in the AS's signing certificate, including
 2492 maintenance when the certificate is replaced, is essential to avoid validation problems. Clock
 2493 synchronization between the AS and the RP is also important; a time difference of five minutes or more
 2494 can cause validation errors as well.

2495 **Appendix A Abbreviations and Acronyms**

AD	Active Directory
API	Application Programming Interface
APNS	Apple Push Notification System
App	Application
App ID	Application Identification
AppAuth	Application Authentication System
AS	Authorization Server
ASM	Authenticator-Specific Module
BCP	Best Current Practice
BIND	Berkeley Internet Name Domain
CA	Certificate Authority
CPSSP	Central Public Safety Service Provider
CPU	Central Processing Unit
CRADA	Cooperative Research and Development Agreement
CSR	Certificate Signing Request
DN	Distinguished Name
DNS	Domain Name System
FIDO	Fast Identity Online
FOIA	Freedom of Information Act
FQDN	Fully Qualified Domain Name
GB	Gigabyte
GCM	Google Cloud Messenger
GHz	Gigahertz
HSM	Hardware Security Module
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identification
IdP	Identity Provider
IETF	Internet Engineering Task Force
iOS	iPhone Operating System
IP	Internet Protocol
IT	Information Technology
JCE	Java Cryptography Extension
JDK	Java Development Kit
JSON	JavaScript Object Notation
JWE	JSON Web Encryption
JWT	JSON Web Token
LDAP	Lightweight Directory Access Protocol
LES	Law Enforcement Sensitive

LGPL	Lesser General Public License
LPSD	Local Public Safety Department
MDM	Mobile Device Management
MFA	Multifactor Authentication
MSSO	Mobile Single Sign-On
NAT	Network Address Translation
NCCoE	National Cybersecurity Center of Excellence
NFC	Near Field Communication
NIST	National Institute of Standards and Technology
NNAS	Nok Nok Labs Authentication Server
NTP	Network Time Protocol
OIDC	OpenID Connect
OOB	Out-of-Band
OS	Operating System
PHI	Protected Health Information
PII	Personally Identifiable Information
PIN	Personal Identification Number
PKCE	Proof Key for Code Exchange
PSCR	Public Safety Communications Research lab
PSFR	Public Safety and First Responder
PSX	Public Safety Experience
QR	Quick Response
RAM	Random Access Memory
REST	Representational State Transfer
RFC	Request for Comments
RP	Relying Party
RPM	Red Hat Package Manager
SaaS	Software as a Service
SAML	Security Assertion Markup Language
SDK	Software Development Kit
SE	Standard Edition
SKCE	StrongKey CryptoEngine
SLO	Single Log-Out
SP	Service Provider
SPSD	State Public Safety Department
SQL	Structured Query Language
SSH	Secure Shell
SSO	Single Sign-On
TCP	Transmission Control Protocol
TEE	Trusted Execution Environment
TLS	Transport Layer Security
U2F	Universal Second Factor

UAF	Universal Authentication Framework
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
USB-C	Universal Serial Bus Type-C
VLAN	Virtual Local Area Network
VPN	Virtual Private Network
WAR	Web Archive

2496 **Appendix B References**

- [1] W. Denniss and J. Bradley, "OAuth 2.0 for Native Apps," BCP 212, RFC 8252, DOI 10.17487/RFC8252, October 2017. [Online]. Available: <https://www.rfc-editor.org/info/rfc8252>. [Accessed 25 February 2018].
- [2] FIDO Alliance, "FIDO Specifications Overview: UAF & U2F," 20 May 2016. [Online]. Available: <https://www.slideshare.net/FIDOAlliance/fido-specifications-overview-uaf-u2f>. [Accessed 25 February 2018].
- [3] Google, "Chrome custom tabs smooth the transition between apps and the web," Android Developers Blog, 2 September 2015. [Online]. Available: <https://android-developers.googleblog.com/2015/09/chrome-custom-tabs-smooth-transition.html>. [Accessed 25 February 2018].
- [4] Google, "Chrome Custom Tabs," 6 May 2016. [Online]. Available: <https://developer.chrome.com/multidevice/android/customtabs>. [Accessed 25 February 2018].
- [5] Google, "Google Chrome: Fast & Secure," Google Play, [Online]. Available: <https://play.google.com/store/apps/details?id=com.android.chrome>. [Accessed 25 February 2018].
- [6] Google, "Google Authenticator," Google Play, [Online]. Available: <https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2>. [Accessed 25 February 2018].
- [7] S. Machani, R. Philpott, S. Srinivas, J. Kemp and J. Hodges, "FIDO UAF Architectural Overview, FIDO Alliance Implementation Draft," 2 February 2017. [Online]. Available: <https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-uaf-overview-v1.1-id-20170202.html>. [Accessed 25 February 2018].
- [8] Nok Nok Labs Inc., "Nok Nok™ Passport," Google Play, [Online]. Available: <https://play.google.com/store/apps/details?id=com.noknok.android.passport2>. [Accessed 25 February 2018].
- [9] Motorola Solutions, "PSX App Suite," [Online]. Available: https://www.motorolasolutions.com/en_us/products/psx-app-suite.html. [Accessed 25 February 2018].
- [10] OpenID Foundation, "openid/AppAuth-Android," GitHub, [Online]. Available: <https://github.com/openid/AppAuth-Android>. [Accessed 25 February 2018].

- [11] D., Hardt, Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749," October 2012. [Online]. Available: <https://www.rfc-editor.org/info/rfc6749>. [Accessed 25 February 2018].
- [12] S. Cantor, J. Kemp, R. Philpott and E. Maler, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0," 15 March 2005. [Online]. Available: <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>. [Accessed 25 February 2018].
- [13] N. E. Sakimura, J. Bradley and N. Agarwal, "Proof Key for Code Exchange by OAuth Public Clients," RFC 7636, DOI 10.17487/RFC7636, September 2015. [Online]. Available: <https://www.rfc-editor.org/info/rfc7636>. [Accessed 25 February 2018].
- [14] M. Jones and J. Hildebrand, "JSON Web Encryption (JWE)," RFC 7516, May 2015. [Online]. Available: <https://tools.ietf.org/html/rfc7516>. [Accessed 25 February 2018].
- [15] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 1," 8 November 2014. [Online]. Available: http://openid.net/specs/openid-connect-core-1_0.html. [Accessed 25 February 2018].
- [16] Microsoft Corporation, "Active Directory Schema," [Online]. Available: [https://msdn.microsoft.com/en-us/library/ms675085\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms675085(v=vs.85).aspx). [Accessed 25 February 2018].
- [17] Nok Nok Labs, Inc., "Nok Nok Labs S3 Authentication Suite Solution Guide," v5.1.1, 2017.
- [18] Nok Nok Labs, Inc., "Nok Nok Authentication Server Administration Guide," v5.1.1, 2017.
- [19] Nok Nok Labs, Inc., "Nok Nok PingFederate Adapter Integration Guide," v1.0.1, 2017.
- [20] StrongAuth, Inc., "PingFederate FIDO IdP Adapter Installation Guide," Revision 2, 2017.
- [21] J. Richer, Ed., "OAuth 2.0 Token Introspection," RFC 7662, October 2015. [Online]. Available: <https://tools.ietf.org/html/rfc7662>. [Accessed 25 February 2018].