

NIST SPECIAL PUBLICATION 1800-13

Mobile Application Single Sign-On

Improving Authentication for Public Safety First Responders

Includes Executive Summary (A); Approach, Architecture, and Security Characteristics (B); and How-To Guides (C)

Bill Fisher
Paul Grassi
William C. Barker
Spike E. Dog
Santos Jha
William Kim
Taylor McCorkill
Joseph Portner
Mark Russell
Sudhi Umarji

May 2019

SECOND DRAFT

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

NIST SPECIAL PUBLICATION 1800-13

Mobile Application Single Sign-On Improving Authentication for Public Safety First Responders

*Includes Executive Summary (A); Approach, Architecture, and Security Characteristics (B);
and How-To Guides (C)*

**Bill Fisher
Paul Grassi***

Applied Cybersecurity Division
Information Technology Laboratory

**Spike E. Dog
Santos Jha
William Kim*
Taylor McCorkill
Joseph Portner*
Mark Russell
Sudhi Umarji**

The MITRE Corporation
McLean, Virginia

William C. Barker
Dakota Consulting
Silver Spring, Maryland

**Former employee; all work for this
publication was done while at employer.*

SECOND DRAFT

May 2019



U.S. Department of Commerce
Wilbur Ross, Secretary

National Institute of Standards and Technology
Walter Copan, NIST Director and Undersecretary of Commerce for Standards and Technology

NIST SPECIAL PUBLICATION 1800-13A

Mobile Application Single Sign-On

Improving Authentication for Public Safety First Responders

Volume A: Executive Summary

Bill Fisher

Paul Grassi*

Applied Cybersecurity Division

Information Technology Laboratory

Spike E. Dog

Santos Jha

William Kim*

Taylor McCorkill

Joseph Portner*

Mark Russell

Sudhi Umarji

The MITRE Corporation

McLean, Virginia

William C. Barker

Dakota Consulting

Silver Spring, Maryland

**Former employee; all work for this publication was done while at employer.*

May 2019

SECOND DRAFT

This publication is available free of charge from

<https://www.nccoe.nist.gov/projects/use-cases/mobile-sso>

1 Executive Summary

- 2 ▪ On-demand access to public safety data is critical to ensuring that public safety and first
3 responders (PSFRs) can protect life and property during an emergency.
- 4 ▪ This public safety information, often needing to be accessed via mobile or portable devices,
5 routinely includes sensitive information, such as personally identifiable information, law
6 enforcement sensitive information, and protected health information.
- 7 ▪ Because the communications are critical to public safety and may include sensitive information,
8 robust and reliable authentication mechanisms that do not hinder delivery of emergency
9 services are required.
- 10 ▪ In collaboration with the National Institute of Standards and Technology (NIST) Public Safety
11 Communications Research laboratory and industry stakeholders, the National Cybersecurity
12 Center of Excellence (NCCoE) at NIST built a laboratory environment to demonstrate standards-
13 based technologies that can enable PSFRs to gain access to public safety information efficiently
14 and securely by using mobile devices.
- 15 ▪ The technologies demonstrated are currently available and include (1) single sign-on (SSO)
16 capabilities that reduce the number of credentials that need to be managed by public safety
17 personnel, and reduce the time and effort that individuals spend authenticating themselves;
18 (2) identity federation that can improve the ability to authenticate personnel across public
19 safety organization (PSO) boundaries; and (3) multifactor authentication (MFA) that enables
20 authentication with a high level of assurance.
- 21 ▪ This NIST Cybersecurity Practice Guide describes how organizations can implement these
22 technologies to enhance public safety mission capabilities by using standards-based
23 commercially available or open-source products. The technologies described facilitate
24 interoperability among diverse mobile platforms, applications, relying parties, identity providers
25 (IdPs), and public-sector and private-sector participants, regardless of the application
26 development platform used in their construction.

27 CHALLENGE

- 28 Recent natural and man-made disasters and crises have highlighted the importance of efficient and
29 secure access to critical information by PSFRs. For decades, much of this information was broadcast to
30 PSFRs by voice over radio. More recently, many PSOs have transitioned to a hybrid model that includes
31 automated access to much of this information via ruggedized mobile laptops and tablets. Further
32 advances in technology have resulted in increasing reliance on smartphones or similar portable devices
33 for field access to public safety information. The increasing reliance on these devices has driven the use
34 of “native app”-based interfaces to access information, in addition to more traditional browser-based
35 methods.
- 36 Many PSOs are in the process of transitioning from traditional land-based mobile communications to
37 high-speed, regional or nationwide wireless broadband networks (e.g., FirstNet). These emerging “5G”
38 systems employ internet protocol-based communications to provide secure and interoperable public
39 safety communications to support initiatives such as Criminal Justice Information Services, Regional
40 Information Sharing Systems, and international justice and public safety services such as those provided

41 by Nlets. This transition will foster critically needed interoperability within and among jurisdictions, but
42 it will create a significant increase in the number of mobile devices that PSOs will need to manage.

43 Current PSO authentication services may not be sustainable in the face of this growth. There are needs
44 to improve security assurance, limit authentication requirements that are imposed on users
45 (e.g., reduce the number of passwords that are required), improve the usability and efficiency of user
46 account management, and share identities across jurisdictional boundaries. There is no single
47 management or administrative hierarchy spanning the PSFR population. PSFR organizations operate in a
48 variety of environments with different authentication requirements. Standards-based solutions are
49 needed to support technical interoperability and a diverse set of PSO environments.

50 **SOLUTION**

51 To address these challenges, the NCCoE brought together common identity and software application
52 providers to demonstrate how a PSO can implement mobile native and web application SSO, access
53 federated identity sources, and implement MFA. SSO limits the time and effort that PSFR personnel
54 spend authenticating, while MFA provides PSOs with adequate confidence that users who are accessing
55 their information are who they say they are. The architecture supports identity federation that allows
56 PSOs to share identity assertions between applications and across PSO jurisdictions. A combination of all
57 of these capabilities can allow PSFR personnel to authenticate—say, at the beginning of their shift—and
58 leverage that high-assurance authentication to gain cross-jurisdictional access to many other mobile
59 native and web applications while on duty.

60 The guide provides

- 61 ■ a detailed example solution and capabilities that address risk and security controls
- 62 ■ a demonstration of the approach using commercially available products
- 63 ■ “how to” instructions for implementers and security engineers on integrating and configuring
64 the example solution into their organization’s enterprise in a manner that achieves security
65 goals with minimal impact on operational efficiency and expense

66 The NCCoE assembled existing technologies that support the following standards:

- 67 ■ Internet Engineering Task Force Request for Comments 8252, *OAuth 2.0 for Native Apps*
- 68 ■ FIDO Universal Second Factor and Universal Authentication Framework
- 69 ■ Security Assertion Markup Language 2.0
- 70 ■ OpenID Connect 1.0

71 Commercial, standards-based products, such as the ones that we used, are readily available and
72 interoperable with existing information technology (IT) infrastructures. While the NCCoE used a suite of
73 commercial products to address this challenge, this guide does not endorse these particular products,
74 nor does it guarantee compliance with any regulatory initiatives. Your organization’s information
75 security experts should identify the products that will best integrate with your existing tools and IT
76 system infrastructure. Your organization can adopt this solution or one that adheres to these guidelines
77 in whole, or you can use this guide as a starting point for tailoring and implementing parts of a solution.

78 **BENEFITS**

79 The NCCoE's practice guide to *Mobile Application Single Sign-On* can help PSOs:

- 80 ▪ define requirements for mobile application SSO and MFA implementation
81 ▪ improve interoperability among mobile platforms, applications, and IdPs, regardless of the
82 application development platform used in their construction
83 ▪ enhance the efficiency of PSFRs by reducing the number of authentication steps, the time
84 needed to access critical data, and the number of credentials that need to be managed
85 ▪ support a diverse set of credentials, enabling a PSO to choose an authentication solution that
86 best meets its individual needs

87 **SHARE YOUR FEEDBACK**

88 You can view or download the guide at <https://www.nccoe.nist.gov/projects/use-cases/mobile-sso>. Help
89 the NCCoE make this guide better by sharing your thoughts with us as you read the guide. If you adopt
90 this solution for your own organization, please share your experience and advice with us. We recognize
91 that technical solutions alone will not fully enable the benefits of our solution, so we encourage
92 organizations to share lessons learned and best practices for transforming the processes associated with
93 implementing this guide.

94 To provide comments or to learn more by arranging a demonstration of this example implementation,
95 contact the NCCoE at psfr-nccoe@nist.gov.

96 **TECHNOLOGY PARTNERS/COLLABORATORS**

97 Organizations participating in this project submitted their capabilities in response to an open call in the
98 Federal Register for all sources of relevant security capabilities from academia and industry (vendors
99 and integrators). The following respondents with relevant capabilities or product components (identified
100 as "Technology Partners/Collaborators" herein) signed a Cooperative Research and Development
101 Agreement (CRADA) to collaborate with NIST in a consortium to build this example solution.



102

103 Certain commercial entities, equipment, products, or materials may be identified by name or company
104 logo or other insignia in order to acknowledge their participation in this collaboration or to describe an
105 experimental procedure or concept adequately. Such identification is not intended to imply special
106 status or relationship with NIST or recommendation or endorsement by NIST or NCCoE; neither is it
107 intended to imply that the entities, equipment, products, or materials are necessarily the best available
108 for the purpose.

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 challenges. Through this collaboration, the NCCoE develops modular, easily adaptable example cybersecurity solutions demonstrating how to apply standards and best practices using commercially available technology.

LEARN MORE

Visit <https://www.nccoe.nist.gov/nccoe@nist.gov>
301-975-0200

NIST SPECIAL PUBLICATION 1800-13B

Mobile Application Single Sign-On

Improving Authentication for Public Safety First Responders

Volume B: Approach, Architecture, and Security Characteristics

Bill Fisher

Paul Grassi*

Applied Cybersecurity Division

Information Technology Laboratory

Spike E. Dog

Santos Jha

William Kim*

Taylor McCorkill

Joseph Portner*

Mark Russell

Sudhi Umarji

The MITRE Corporation

McLean, Virginia

William C. Barker

Dakota Consulting

Silver Spring, Maryland

**Former employee; all work for this publication was done while at employer.*

May 2019

SECOND 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 by name or company logo or other insignia in order to acknowledge their participation in this collaboration or to describe an experimental procedure or concept adequately. Such identification is not intended to imply special status or relationship with NIST or recommendation or endorsement by NIST or NCCoE; neither 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 1800-13B, Natl. Inst. Stand. Technol. Spec. Publ. 1800-13B, 73 pages (May 2019), 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: May 29, 2019, through June 28, 2019

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

National Cybersecurity Center of Excellence
National Institute of Standards and Technology
100 Bureau Drive
Mailstop 2002
Gaithersburg, Maryland 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 information technology 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 Cybersecurity 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, Maryland.

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

NIST CYBERSECURITY PRACTICE GUIDES

NIST Cybersecurity Practice Guides (Special Publication 1800 series) 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, law enforcement sensitive information, and protected health information. 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 and industry stakeholders, the NCCoE aims to help PSFR personnel efficiently and securely gain access to mission data via mobile devices and applications. This practice guide describes a reference design for multifactor authentication (MFA) and mobile single sign-on (MSSO) for native and web applications while improving interoperability among mobile platforms, applications, and identity providers, regardless of the application 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 that uses commercially available and open-source products.

This guide discusses potential security risks facing organizations, benefits that may result from 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; relying party; single sign-on

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
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

Name	Organization
Arshad Noor	StrongKey
Pushkar Marathe	StrongKey
Max Smyth	StrongKey
Scott Wong	StrongKey
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
Motorola Solutions	Mobile Apps
Yubico	External Authenticators
Nok Nok Labs	Fast Identity Online (FIDO) Universal Authentication Framework Server
StrongKey	FIDO Universal Second Factor Server

1 **Contents**

2	1 Summary	1
3	1.1 Challenge.....	1
4	1.1.1 Easing User Authentication Requirements	2
5	1.1.2 Improving Authentication Assurance	2
6	1.1.3 Federating Identities and User Account Management	2
7	1.2 Solution.....	3
8	1.3 Benefits.....	4
9	2 How to Use This Guide	4
10	2.1 Typographic Conventions.....	6
11	3 Approach	6
12	3.1 Audience.....	6
13	3.2 Scope	7
14	3.3 Assumptions	8
15	3.4 Business Case.....	9
16	3.5 Risk Assessment	9
17	3.5.1 PSFR Risks	10
18	3.5.2 Mobile Ecosystem Threats	10
19	3.5.3 Authentication and Federation Threats	13
20	3.6 Systems Engineering.....	15
21	3.7 Technologies.....	15
22	4 Architecture	17
23	4.1 General Architectural Considerations.....	17
24	4.1.1 SSO with OAuth 2.0, IETF RFC 8252, and AppAuth Open-Source Libraries	18
25	4.1.2 Identity Federation	19
26	4.1.3 FIDO and Authenticator Types	19
27	4.2 High-Level Architecture.....	19
28	4.3 Detailed Architecture Flow.....	22

29	4.3.1	SAML and U2F Authentication Flow	22
30	4.3.2	OpenID Connect and UAF Authentication Flow	27
31	4.4	Single Sign-On with the OAuth Authorization Flow	31
32	4.5	Application Developer Perspective of the Build	32
33	4.6	Identity Provider Perspective of the Build	32
34	4.7	Token and Session Management	33
35	5	Security Characteristic Analysis.....	33
36	5.1	Assumptions and Limitations	34
37	5.2	Threat Analysis	34
38	5.2.1	Mobile Ecosystem Threat Analysis	34
39	5.2.2	Authentication and Federation Threat Analysis.....	36
40	5.3	Scenarios and Findings	38
41	6	Future Build Considerations.....	39
42	6.1	Single Logout	39
43	6.2	Shared Devices	40
44	6.3	Step-Up Authentication.....	40
45	Appendix A	Mapping to Cybersecurity Framework Core.....	41
46	Appendix B	Assumptions Underlying the Build	45
47	B.1	Identity Proofing.....	45
48	B.2	Mobile Device Security.....	45
49	B.3	Mobile Application Security	45
50	B.4	Enterprise Mobility Management	47
51	B.5	FIDO Enrollment Process.....	48
52	Appendix C	Architectural Considerations for the Mobile Application Single Sign-On Build.....	49
54	C.1	SSO with OAuth 2.0, IETF RFC 8252, and AppAuth Open-Source Libraries	49
55	C.1.1	Attributes and Authorization.....	51
56	C.2	Federation	52

57	C.3 Authenticator Types	53
58	C.3.1 UAF Protocol	56
59	C.3.2 U2F Protocol	57
60	C.3.3 FIDO 2	57
61	C.3.4 FIDO Key Registration	57
62	C.3.5 FIDO Authenticator Attestation	58
63	C.3.6 FIDO Deployment Considerations	59
64	Appendix D Acronyms.....	61
65	Appendix E References	63

66 **List of Figures**

67	Figure 3-1 The Mobile Ecosystem	13
68	Figure 4-1 High-Level U2F Architecture	20
69	Figure 4-2 High-Level UAF Architecture.....	21
70	Figure 4-3 SAML and U2F Sequence Diagram	23
71	Figure 4-4 OIDC and UAF Sequence Diagram	27
72	Figure 5-1 Mobile Device Technology Stack.....	35

73 **List of Tables**

74	Table 3-1 Threat Classes and Categories	11
75	Table 3-2 Products and Technologies	15
76	Table A-1 Cybersecurity Framework Categories	41
77	Table C-1 FAL Requirements.....	53
78	Table C-2 AAL Summary of Requirements	55

79 1 Summary

80 The National Cybersecurity Center of Excellence (NCCoE), with the National Institute of Standards and
81 Technology's (NIST's) Public Safety Communications Research lab, is helping the public safety and first
82 responder (PSFR) community address the challenge of securing sensitive information accessed on
83 mobile applications. The Mobile Application Single Sign-On (SSO) Project is a collaborative effort with
84 industry and the information technology (IT) community, including vendors of cybersecurity solutions.

85 This project aims to help PSFR personnel efficiently and securely gain access to mission-critical data via
86 mobile devices and applications through mobile SSO, identity federation, and multifactor authentication
87 (MFA) solutions for native and web applications by using standards-based commercially available and
88 open-source products.

89 The reference design herein

- 90 ▪ provides a detailed example solution and capabilities that address risk and security controls
- 91 ▪ demonstrates standards-based MFA, identity federation, and mobile SSO for native and web
92 applications
- 93 ▪ supports multiple authentication methods, considering unique environmental constraints faced
94 by first responders in emergency medical services, law enforcement, and fire services

95 1.1 Challenge

96 On-demand access to public safety data is critical to ensuring that PSFR personnel can protect life and
97 property during an emergency. Mobile platforms offer a significant operational advantage to public
98 safety stakeholders by providing access to mission-critical information and services while deployed in
99 the field, during training and exercises, or when participating in day-to-day business and preparing for
100 emergencies during nonemergency periods. These advantages can be limited if complex authentication
101 requirements hinder PSFR personnel, especially when a delay—even seconds—is a matter of containing
102 or exacerbating an emergency. PSFR communities are challenged with implementing efficient and
103 secure authentication mechanisms to protect access to this sensitive information while meeting the
104 demands of their operational environment.

105 Many public safety organizations (PSOs) are in the process of transitioning from traditional land-based
106 mobile communications to high-speed, regional or nationwide wireless broadband networks (e.g., First
107 Responder Network Authority [FirstNet]). These emerging 5G systems employ internet protocol-based
108 communications to provide secure and interoperable public safety communications to support
109 initiatives such as Criminal Justice Information Services; Regional Information Sharing Systems; and
110 international justice and public safety services, such as those provided by Nlets. This transition will
111 foster critically needed interoperability within and among jurisdictions but will create a significant

112 increase in the number of mobile Android and iPhone operating system (iOS) devices that PSOs will need
113 to manage.

114 Current PSO authentication services may not be sustainable in the face of this growth. There are needs
115 to improve security assurance, limit authentication requirements that are imposed on users (e.g., avoid
116 the number of passwords that are required), improve the usability and efficiency of user account
117 management, and share identities across jurisdictional boundaries. There is no single management or
118 administrative hierarchy spanning the PSFR population. PSFR organizations operate in a variety of
119 environments with different authentication requirements. Standards-based solutions are needed to
120 support technical interoperability and this diverse set of PSO environments.

121 **1.1.1 Easing User Authentication Requirements**

122 Many devices that digitally access public safety information employ different software applications to
123 access different information sources. Single-factor authentication processes, usually passwords, are
124 most commonly required to access each of these applications. Users often need different passwords or
125 personal identification numbers (PINs) for each application used to access critical information.
126 Authentication prompts, such as entering complex passwords on a small touchscreen for each
127 application, can hinder PSFRs. There is an operational need for the mobile systems on which they rely to
128 support a single authentication process that can be used to access multiple applications. This is referred
129 to as single sign-on, or SSO.

130 **1.1.2 Improving Authentication Assurance**

131 Single-factor password authentication mechanisms for mobile native and web applications may not
132 provide sufficient protection for control of access to law enforcement-sensitive information, protected
133 health information, and personally identifiable information (PII). Replacement of passwords by
134 multifactor technology (e.g., a PIN plus some physical token or biometric) is widely recognized as
135 necessary for access to sensitive information. Technology for these capabilities exists, but budgetary,
136 contractual, and operational considerations have impeded implementation and use of these
137 technologies. PSOs need a solution that supports differing authenticator requirements across the
138 community (e.g., law enforcement, fire response, emergency medical services) and a “future proof”
139 solution allowing for adoption of evolving technologies that may better support PSFRs in the line of
140 duty.

141 **1.1.3 Federating Identities and User Account Management**

142 PSFRs need access to a variety of applications and databases to support routine activities and
143 emergency situations. These resources may be accessed by portable mobile devices or mobile data
144 terminals in vehicles. It is not uncommon for these resources to reside within neighboring jurisdictions
145 at the federal, state, county, or local level. Even when the information is within the same jurisdiction, it
146 may reside in a third-party vendor’s cloud service. This environment results in issuance of many user

147 accounts to each PSFR that are managed and updated by those neighboring jurisdictions or cloud service
148 providers. When a PSFR leaves or changes job functions, the home organization must ensure that
149 accounts are deactivated, avoiding any orphaned accounts managed by third parties. PSOs need a
150 solution that reduces the number of accounts managed and allows user accounts and credentials issued
151 by a PSFR's home organization to access information across jurisdictions and with cloud services. The
152 ability of one organization to accept the identity and credentials from another organization in the form
153 of an identity assertion is called identity federation. Current commercially available standards support
154 this functionality.

155 **1.2 Solution**

156 This NIST Cybersecurity Practice Guide demonstrates how commercially available technologies,
157 standards, and best practices implementing SSO, identity federation, and MFA can meet the needs of
158 public safety first responder communities when accessing services from mobile devices.

159 In our lab at the NCCoE, we built an environment that simulates common identity providers (IdPs) and
160 software applications found in PSFR infrastructure. In this guide, we show how a PSFR entity can
161 leverage this infrastructure to implement SSO, identity federation, and MFA for native and web
162 applications on mobile platforms. SSO, federation, and MFA capabilities can be implemented
163 independently, but implementing them together would achieve maximum improvement with respect to
164 usability, interoperability, and security.

165 At its core, the architecture described in [Section 4](#) implements the Internet Engineering Task Force's
166 (IETF's) best current practice (BCP) guidance found in Request for Comments (RFC) 8252, *OAuth 2.0 for*
167 *Native Apps* [\[1\]](#). Leveraging technology newly available in modern mobile operating systems (OSes), RFC
168 8252 defines a specific flow allowing for authentication to mobile native applications without exposing
169 user credentials to the client application. This authentication can be leveraged by additional mobile
170 native and web applications to provide an SSO experience, avoiding the need for the user to manage
171 credentials independently for each application. Using the Fast Identity Online (FIDO) Universal
172 Authentication Framework (UAF) [\[2\]](#) and Universal Second Factor (U2F) [\[3\]](#) protocols, this solution
173 supports MFA on mobile platforms that use a diverse set of authenticators. The use of security assertion
174 markup language (SAML) 2.0 [\[4\]](#) and OpenID Connect (OIDC) 1.0 [\[5\]](#) federation protocols allows PSOs to
175 share identity assertions between applications and across PSO jurisdictions. Using this architecture
176 allows PSFR personnel to authenticate once—say, at the beginning of their shift—and then leverage that
177 single authentication to gain access to many other mobile native and web applications while on duty,
178 reducing the time needed for authentication.

179 The PSFR community comprises tens of thousands of different organizations across the United States,
180 many of which may operate their own IdPs. Today, most IdPs use SAML 2.0, but OIDC is rapidly gaining
181 market share as an alternative for identity federation. As this build architecture demonstrates, an OAuth
182 authorization server (AS) can integrate with both OIDC and SAML IdPs.

183 The guide provides:

- 184 ▪ a detailed example solution and capabilities that may be implemented independently or in
185 combination to address risk and security controls
- 186 ▪ a demonstration of the approach, which uses commercially available products
- 187 ▪ how-to instructions for implementers and security engineers on integrating and configuring the
188 example solution into their organization's enterprise in a manner that achieves security goals
189 with minimal impact on operational efficiency and expense

190 Organizations can adopt this solution or a different one that adheres to these guidelines in whole, or an
191 organization can use this guide as a starting point for tailoring and implementing parts of a solution.

192 **1.3 Benefits**

193 The NCCoE, in collaboration with our stakeholders in the PSFR community, identified the need for a
194 mobile SSO and MFA solution for native and web applications. This NCCoE practice guide, *Mobile*
195 *Application Single Sign-On*, can help PSOs:

- 196 ▪ define requirements for mobile application SSO and MFA implementation
- 197 ▪ improve interoperability among mobile platforms, applications, and IdPs, regardless of the
198 application development platform used in their construction
- 199 ▪ enhance the efficiency of PSFRs by reducing the number of authentication steps, the time
200 needed to access critical data, and the number of credentials that need to be managed
- 201 ▪ support a diverse set of credentials, enabling a PSO to choose an authentication solution that
202 best meets its individual needs
- 203 ▪ enable cross-jurisdictional information sharing by identity federation

204 **2 How to Use This Guide**

205 This NIST Cybersecurity Practice Guide demonstrates a standards-based reference design and provides
206 users with the information they need to replicate an MFA and mobile SSO solution for mobile native and
207 web applications. This reference design is modular and can be deployed in whole or in part.

208 This guide contains three volumes:

- 209 ▪ NIST Special Publication (SP) 1800-13A: *Executive Summary*
- 210 ▪ NIST SP 1800-13B: *Approach, Architecture, and Security Characteristics*—what we built and why
211 (**you are here**)
- 212 ▪ NIST SP 1800-13C: *How-To Guides*—instructions for building the example solution

213 Depending on your role in your organization, you might use this guide in different ways:

214 **Business decision makers, including chief security and technology officers**, will be interested in the
215 *Executive Summary* (NIST SP 1800-13A), which describes the following topics:

- 216 ▪ challenges that enterprises face in MFA and mobile SSO for native and web applications
217 ▪ example solution built at the NCCoE
218 ▪ benefits of adopting the example solution

219 **Technology or security program managers** who are concerned with how to identify, understand, assess,
220 and mitigate risk will be interested in this part of the guide, NIST SP 1800-13B, which describes what we
221 did and why. The following sections will be of particular interest:

- 222 ▪ [Section 3.5](#), Risk Assessment, provides a description of the risk analysis we performed.
223 ▪ [Appendix A](#), Mapping to Cybersecurity Framework Core, maps the security characteristics of this
224 example solution to cybersecurity standards and best practices.

225 You might share the *Executive Summary*, NIST SP 1800-13A, with your leadership team members to help
226 them understand the importance of adopting a standards-based MFA and mobile SSO solution for native
227 and web applications.

228 **Information Technology (IT) professionals** who want to implement an approach like this will find the
229 whole practice guide useful. You can use the how-to portion of the guide, NIST SP 1800-13C, to replicate
230 all or parts of the build created in our lab. The how-to portion of the guide provides specific product
231 installation, configuration, and integration instructions for implementing the example solution. We do
232 not re-create the product manufacturer's documentation, which is generally widely available. Rather,
233 we show how we incorporated the products together in our environment to create an example solution.

234 This guide assumes that IT professionals have experience implementing security products within the
235 enterprise. While we have used a suite of commercial products to address this challenge, this guide does
236 not endorse these particular products. Your organization can adopt this solution or one that adheres to
237 these guidelines in whole, or you can use this guide as a starting point for tailoring and implementing
238 SSO or MFA separately. Your organization's security experts should identify the products that will best
239 integrate with your existing tools and IT system infrastructure. We hope you will seek products that are
240 congruent with applicable standards and best practices. [Section 3.7](#), Technologies, lists the products we
241 used and maps them to the cybersecurity controls provided by this reference solution.

242 A NIST Cybersecurity Practice Guide does not describe "the" solution, but a possible solution. This is a
243 draft guide. We seek feedback on its contents and welcome your input. Comments, suggestions, and
244 success stories will improve subsequent versions of this guide. Please contribute your thoughts to psfr-nccoe@nist.gov.

246 2.1 Typographic Conventions

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

Typeface/Symbol	Meaning	Example
<i>Italics</i>	file names 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 .
Monospace	command-line input, onscreen computer output, sample code examples, and status codes	<code>mkdir</code>
Monospace Bold	command-line user input contrasted with computer output	service sshd start
<u>blue text</u>	link to other parts of the document, a web URL, or an email address	All publications from NIST's NCCoE are available at https://www.nccoe.nist.gov .

248 3 Approach

249 In conjunction with the PSFR community, the National Cybersecurity Center of Excellence developed a
 250 project description identifying MFA and SSO for mobile native and web applications as a critical need for
 251 PSFR organizations. The NCCoE then engaged subject matter experts from industry organizations,
 252 technology vendors, and standards bodies to develop an architecture and reference design leveraging
 253 new capabilities in modern mobile OSes and best current practices in SSO and MFA.

254 3.1 Audience

255 This guide is intended for individuals or entities that are interested in understanding the mobile native
 256 and web application SSO and MFA reference designs that the NCCoE has implemented to allow PSFR

257 personnel to securely and efficiently gain access to mission-critical data by using mobile devices. Though
258 the NCCoE developed this reference design with the PSFR community, any party interested in SSO and
259 MFA for native mobile and web applications can leverage the architecture and design principles
260 implemented in this guide.

261 The overall build architecture addresses three different audiences with somewhat separate concerns:

- 262 ■ IdPs—PSFR organizations that issue and maintain user accounts for their users. Larger PSFR
263 organizations may operate their own IdP infrastructures and may federate by using SAML or
264 OIDC services, while others may seek to use an IdP service provider. IdPs are responsible for
265 identity proofing, account creation, account and attribute management, and credential
266 management.
- 267 ■ Relying parties (RPs)—organizations providing application services to multiple PSFR
268 organizations. RPs may be software as a service (SaaS) providers or PSFR organizations providing
269 shared services consumed by other organizations. The RP operates an OAuth 2.0 AS, which
270 integrates with users' IdPs and issues access tokens to enable mobile applications to make
271 requests to the back-end application servers.
- 272 ■ Application developers—mobile application developers. Today, mobile client applications are
273 typically developed by the same software provider as the back-end RP applications. However,
274 the OAuth framework enables interoperability between RP applications and third-party client
275 applications. In any case, mobile application development is a specialized skill with unique
276 considerations and requirements. Mobile application developers should consider implementing
277 the AppAuth library for IETF RFC 8252 to enable standards-based SSO.

278 **3.2 Scope**

279 The focus of this project is to address the need for secure and efficient mobile native and web
280 application SSO. The NCCoE drafted a use case that identified numerous desired solution characteristics.
281 After an open call in the Federal Register for vendors to help develop a solution, we chose participating
282 technology collaborators on a first-come, first-served basis. We scoped the project to produce the
283 following high-level desired outcomes:

- 284 ■ Provide a standards-based solution architecture that selects an effective and secure approach to
285 implementing mobile SSO, leveraging native capabilities of the mobile OS.
- 286 ■ Ensure that mobile applications do not have access to user credentials.
- 287 ■ Support MFA and multiple authentication protocols.
- 288 ■ Support multiple authenticators, considering unique environmental constraints faced by first
289 responders in emergency medical services, law enforcement, and fire services.
- 290 ■ Support cross-jurisdictional information sharing through identity federation.

291 To maintain the project’s focus on core SSO and MFA requirements, the following subjects are out of
292 scope. These technologies and practices are critical to a successful implementation, but they do not
293 directly affect the core design decisions.

- 294 ■ Identity proofing—The solution creates synthetic digital identities that represent the identities
295 and attributes of public safety personnel to test authentication assertions. This includes the
296 usage of a lab-configured identity repository—not a genuine repository and schema provided by
297 any PSO. This guide will not demonstrate an identity proofing process.
- 298 ■ Access control—This solution supports the creation and federation of attributes but will not
299 discuss or demonstrate access control policies that an RP might implement to govern access to
300 specific resources.
- 301 ■ Credential storage—This solution is agnostic to where credentials are stored on the mobile
302 device. For example, this use case is not affected by storing a certificate in software versus
303 hardware, such as a trusted platform module.
- 304 ■ Enterprise Mobility Management (EMM)—The solution assumes that all applications involved in
305 the SSO experience are allowable via an EMM. This implementation may be supported by using
306 an EMM (for example, to automatically provision required mobile applications to the device),
307 but it does not strictly depend on using an EMM.
- 308 ■ Fallback authentication mechanisms—This solution involves the use of multifactor
309 authenticators, which may consist of physical authentication devices or cryptographic keys
310 stored directly on mobile devices. Situations may arise where a user’s authenticator or device
311 has been lost or stolen. This practice guide recommends registering multiple authenticators for
312 each user as a partial mitigation, but in some cases, it may be necessary to either enable users
313 to fall back to single-factor authentication or provide other alternatives. Such fallback
314 mechanisms must be evaluated considering the organization’s security and availability
315 requirements.

316 **3.3 Assumptions**

317 Before implementing the capabilities described in this practice guide, organizations should review the
318 assumptions underlying the NCCoE build. These assumptions are detailed in [Appendix B](#). Though not in
319 scope for this effort, implementers should consider whether the same assumptions can be made based
320 on current policy, process, and IT infrastructure. As detailed in [Appendix B](#), applicable and appropriate
321 guidance is provided to assist this process for the following functions:

- 322 ■ identity proofing
- 323 ■ mobile device security
- 324 ■ mobile application security
- 325 ■ EMM

326 ▪ FIDO enrollment process

327 3.4 Business Case

328 Any decision to implement IT systems within an organization must begin with a solid business case. This
329 business case could be an independent initiative or a component of the organization's strategic planning
330 cycle. Individual business units or functional areas typically derive functional or business unit strategies
331 from the overall organization's strategic plan. The business drivers for any IT project must originate in
332 these strategic plans, and the decision to determine if an organization will invest in mobile SSO, identity
333 federation, or MFA by implementing the solution in this practice guide will be based on the
334 organization's decision-making process for initiating new projects.

335 Important inputs to the business case are the risks to the organization from mobile authentication and
336 identity management, as outlined in Section 3.5. Apart from addressing cybersecurity risks, SSO also
337 improves the user experience and alleviates the overhead associated with maintaining and using
338 passwords for multiple applications. This provides a degree of convenience to all types of users, but
339 reducing the authentication overhead for PSFR users and reducing barriers to getting the information
340 and applications that they need could have a tremendous effect. First responder organizations and
341 application providers also benefit by using interoperable standards that provide easy integration across
342 disparate technology platforms. In addition, the burden of account management is reduced by using a
343 single user account managed by the organization to access multiple applications and services.

344 3.5 Risk Assessment

345 NIST SP 800-30 Revision 1 [\[6\]](#), *Guide for Conducting Risk Assessments*, states that risk is “a measure of
346 the extent to which an entity is threatened by a potential circumstance or event, and typically a function
347 of (i) the adverse impacts that would arise if the circumstance or even occurs; and (ii) the likelihood of
348 occurrence.” The guide further defines risk assessment as “the process of identifying, estimating, and
349 prioritizing risks to organizational operations (including mission, functions, image, reputation),
350 organizational assets, individuals, other organizations, and the Nation, resulting from the operation of
351 an information system. Part of risk management incorporates threat and vulnerability analyses, and
352 considers mitigations provided by security controls planned or in place.”

353 The NCCoE recommends that any discussion of risk management, particularly at the enterprise level,
354 begins with a comprehensive review of NIST SP 800-37 Revision 2, *Guide for Applying the Risk*
355 *Management Framework to Federal Information Systems* [\[7\]](#)—material that is available to the public.
356 The risk management framework guidance, as a whole, proved invaluable in giving us a baseline to
357 assess risks, from which we developed the project, the security characteristics of the build, and this
358 guide.

359 **3.5.1 PSFR Risks**

360 As PSFR communities adopt mobile platforms and applications, organizations should consider potential
361 risks that these new devices and ecosystems introduce that may negatively affect PSFR organizations
362 and the ability of PSFR personnel to operate. These are some of the risks:

- 363 ■ The reliance on passwords alone by many PSFR entities effectively expands the scope of a single
364 application/database compromise when users fall back to reusing a small set of easily
365 remembered passwords across multiple applications.
- 366 ■ Complex passwords are harder to remember and input to IT systems. Mobile devices exacerbate
367 this issue with small touchscreens that may not work with gloves or other PSFR equipment, and
368 with three separate keyboards among which the user must switch. In an emergency response,
369 any delay in accessing information may prove critical to containing a situation.
- 370 ■ Social engineering, man-in-the-middle attacks, replay attacks, and phishing all present real
371 threats to password-based authentication systems.
- 372 ■ Deterministic, cryptographic authentication mechanisms have security benefits yet come with
373 the challenge of cryptographic key management. Loss or misuse of cryptographic keys could
374 undermine an authentication system, leading to unauthorized access or data leakage.
- 375 ■ Biometric authentication mechanisms may be optimal for some PSFR personnel, yet
376 organizations need to ensure that PII, such as fingerprint templates, is protected.
- 377 ■ Credentials exposed to mobile applications could be stolen by malicious applications or misused
378 by nonmalicious applications. Previously, it was common for native applications to use
379 embedded user-agents (commonly implemented with web views) for OAuth requests. That
380 approach has many drawbacks, including the host application being able to copy user
381 credentials and cookies, as well as the user needing to authenticate again in each application.

382 **3.5.2 Mobile Ecosystem Threats**

383 Any discussion of risks and vulnerabilities is incomplete without considering the threats that are
384 involved. NIST SP 800-150, *Guide to Cyber Threat Information Sharing* [8], states that a cyber threat is
385 “any circumstance or event with the potential to adversely impact organizational operations (including
386 mission, functions, image, or reputation), organizational assets, individuals, other organizations, or the
387 Nation through an information system via unauthorized access, destruction, disclosure, or modification
388 of information, and/or denial of service.”

389 To simplify this concept, a *threat* is anything that can exploit a *vulnerability* to damage an *asset*. Finding
390 the intersection of these three will yield a *risk*. Understanding the applicable threats to a system is the
391 first step in determining its risks.

392 However, identifying and delving into mobile threats is not the primary goal of this practice guide.
393 Instead, we rely on prior work from NIST’s [Mobile Threat Catalogue](#) (MTC), along with its associated

394 NIST Interagency Report 8144, *Assessing Threats to Mobile Devices & Infrastructure* [9]. Each entry in
 395 the MTC contains several pieces of information: an identifier, a category, a high-level description, details
 396 on its origin, exploit examples, examples of common vulnerabilities and exposures, possible
 397 countermeasures, and academic references. For the purposes of this practice guide, we are primarily
 398 interested in threat identifiers, categories, descriptions, and countermeasures.

399 In broad strokes, the MTC covers 32 threat categories that are grouped into 12 distinct classes, as shown
 400 in Table 3-1. Of these categories, three in particular, highlighted in green in the table, are covered by the
 401 guidance in this practice guide. If implemented correctly, this guidance will help mitigate those threats.

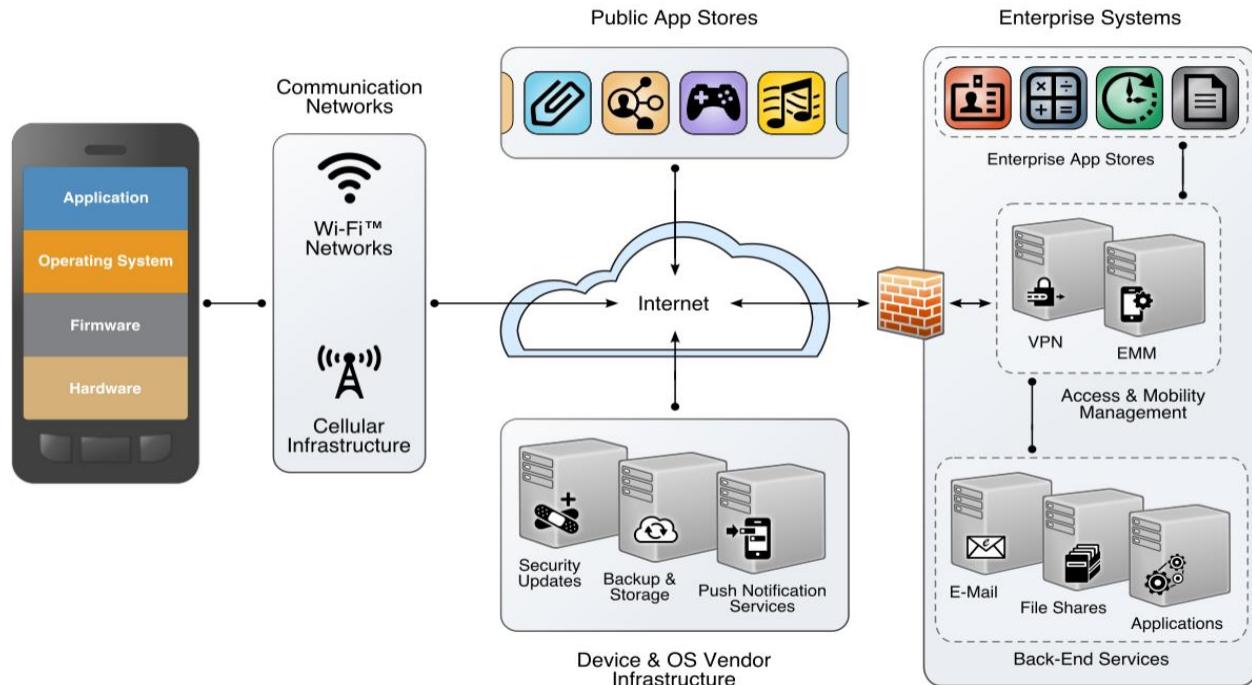
402 **Table 3-1 Threat Classes and Categories**

Threat Class	Threat Category	Threat Class	Threat Category
Application	Malicious or Privacy-Invasive Applications Vulnerable Applications	Local Area Network and Personal Area Network	Network Threats: Bluetooth Network Threats: Near Field Communication (NFC) Network Threats: Wi-Fi
Authentication	Authentication: User or Device to Network Authentication: User or Device to Remote Service Authentication: User to Device	Payment	Application-Based In-Application Purchases
Cellular	Carrier Infrastructure Carrier Interoperability Cellular Air Interface Consumer-Grade Femtocell SMS/MMS/RCS USSD VoLTE	Physical Access	Physical Access
		Privacy	Behavior Tracking
		Supply Chain	Supply Chain
		Stack	Baseband Subsystem Boot Firmware Device Drivers

Threat Class	Threat Category	Threat Class	Threat Category
Ecosystem	Mobile Application Store		Isolated Execution Environments
	Mobile OS & Vendor Infrastructure		Mobile Operating System
EMM	EMM		SD Card
Global Positioning System (GPS)	GPS		USIM/SIM/UICC Security

403 The other categories, while still important elements of the mobile ecosystem and critical to the health of
 404 an overall mobility architecture, are out of scope for this document. The entire mobile ecosystem should
 405 be considered when analyzing threats to the architecture; this ecosystem is depicted in Figure 3-1, taken
 406 from NIST Interagency Report 8144. Each player in the ecosystem—the mobile device user, the
 407 enterprise, the network operator, the application developer, and the original equipment manufacturer
 408 (OEM)—can find suggestions to deter other threats by reviewing the MTC and NIST Interagency Report
 409 8144. Many of these share common solutions, such as using EMM software to monitor device health,
 410 and installing applications from only authorized sources.

411 Figure 3-1 The Mobile Ecosystem



412

413

3.5.3 Authentication and Federation Threats

414 The MTC is a useful reference from the perspective of mobile devices, applications, and networks. In the
 415 context of mobile SSO, specific threats to authentication and federation systems must also be
 416 considered. Table 8-1 in NIST SP 800-63B [10] lists several categories of threats against authenticators:

- 417 ■ theft—stealing a physical authenticator, such as a smart card or U2F device
- 418 ■ duplication—unauthorized copying of an authenticator, such as a password or private key
- 419 ■ eavesdropping—interception of an authenticator secret when in use
- 420 ■ offline cracking—attacks on authenticators that do not require interactive authentication
 421 attempts, such as brute-force attacks on passwords used to protect cryptographic keys
- 422 ■ side-channel attack—exposure of an authentication secret through observation of the
 423 authenticator's physical characteristics
- 424 ■ phishing or pharming—capturing authenticator output through impersonation of the RP or IdP
- 425 ■ social engineering—using a pretext to convince the user to subvert the authentication process

- 426 ▪ online guessing—attempting to guess passwords through repeated online authentication
427 attempts with the RP or IdP
- 428 ▪ end point compromise—malicious code on the user’s device, which is stealing authenticator
429 secrets, redirecting authentication attempts to unintended RPs, or otherwise subverting the
430 authentication process
- 431 ▪ unauthorized binding—binding an attacker-controlled authenticator with the user’s account by
432 intercepting the authenticator during provisioning or impersonating the user in the enrollment
433 process

434 These threats undermine the basic assumption that use of an authenticator in an authentication
435 protocol demonstrates that the user initiating the protocol is the individual referenced by the claimed
436 user identifier. Mitigating these threats is the primary design goal of MFA, and the FIDO specifications
437 address many of these threats.

438 An additional set of threats concerns federation protocols. Authentication threats affect the process of
439 direct authentication of the user to the RP or IdP, whereas federation threats affect the assurance that
440 the IdP can deliver assertions that are genuine and unaltered, only to the intended RP. Table 8-1 in NIST
441 SP 800-63C [\[11\]](#) lists the following federation threats:

- 442 ▪ assertion manufacture or modification—generation of a false assertion or unauthorized
443 modification of a valid assertion
- 444 ▪ assertion disclosure—disclosure of sensitive information contained in an assertion to an
445 unauthorized third party
- 446 ▪ assertion repudiation by the IdP—IdP denies having authenticated a user after the fact
- 447 ▪ assertion repudiation by the subscriber—subscriber denies having authenticated and performed
448 actions on the system
- 449 ▪ assertion redirect—subversion of the federation protocol flow to enable an attacker to obtain
450 the assertion or to redirect it to an unintended RP
- 451 ▪ assertion reuse—attacker obtains a previously used assertion to establish his own session with
452 the RP
- 453 ▪ assertion substitution—attacker substitutes an assertion for a different user in the federation
454 flow, leading to session hijacking or fixation

455 Federation protocols are complex and require interaction among multiple systems, typically under
456 different management. Implementers should carefully apply best security practices relevant to the
457 federation protocols in use. Most federation protocols can incorporate security measures to address
458 these threats, but this may require specific configuration and enabling optional features.

459 3.6 Systems Engineering

460 Some organizations use a systems engineering-based approach to plan and implement their IT projects.
 461 Organizations wishing to implement IT systems should develop robust requirements, taking into
 462 consideration the operational needs of each system stakeholder. Standards such as International
 463 Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) ISO/IEC/IEEE
 464 15288:2015, *Systems and software engineering—System life cycle processes* [\[12\]](#); and NIST SP 800-160,
 465 *Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of*
 466 *Trustworthy Secure Systems* [\[13\]](#) provide guidance for applying security in systems development. With
 467 both standards, organizations can choose to adopt only those sections of the standard that are relevant
 468 to their development approach, environment, and business context. NIST SP 800-160 recommends a
 469 thorough analysis of alternative solution classes accounting for security objectives, considerations,
 470 concerns, limitations, and constraints. This advice applies to both new system developments and
 471 integration of components into existing systems, the focus of this practice guide. [Section 4.1](#), General
 472 Architecture Considerations, may assist organizations with this analysis.

473 3.7 Technologies

474 Table 3-2 lists all of the technologies used in this project and provides a mapping among the generic
 475 application term, the specific product used, and the NIST Cybersecurity Framework Subcategory that the
 476 product provides. For a mapping of Cybersecurity Framework Subcategories to security controls, please
 477 refer to [Appendix A](#), Mapping to Cybersecurity Framework Core. Refer to Table A-1 for an explanation of
 478 the Cybersecurity Framework Category and Subcategory codes.

479 **Table 3-2 Products and Technologies**

Component	Specific Product Used	How the Component Functions in the Build	Applicable Cybersecurity Framework Subcategories
Federation Server	Ping Federate 8.2	OAuth 2.0 AS OIDC provider SAML 2 IdP	PR.AC-3: Remote access is managed.
FIDO U2F Server	StrongKey Crypto Engine (SKCE) 2.0	FIDO U2F server	PR.AC-1: Identities and credentials are managed for authorized devices and users.

Component	Specific Product Used	How the Component Functions in the Build	Applicable Cybersecurity Framework Subcategories
External Authenticator	YubiKey Neo	FIDO U2F token supporting authentication over NFC	PR.AC-1: Identities and credentials are managed for authorized devices and users.
FIDO UAF Server	Nok Nok Labs FIDO UAF Server	UAF authenticator enrollment, authentication, and transaction confirmation	PR.AC-1: Identities and credentials are managed for authorized devices and users.
Mobile Applications (including SaaS back end)	Motorola Solutions Public Safety Experience (PSX) Cockpit, PSX Messenger, and PSX Mapping 5.2; custom demo applications developed by the build team	Provide application programming interfaces (APIs) for mobile client applications to access cloud-hosted services and data; consume OAuth tokens	PR.AC-3: Remote access is managed.
SSO Implementing Best Current Practice	AppAuth Software Development Kit (SDK) for iOS and Android	Library used by mobile applications, providing an IETF RFC 8252-compliant OAuth 2.0 client implementation; implements authorization requests, Proof Key for Code Exchange (PKCE), and token refresh	PR.AC-3: Remote access is managed.

480 4 Architecture

481 The NCCoE worked with industry subject matter experts to develop an open, standards-based,
482 commercially available architecture demonstrating three main capabilities:

- 483 ▪ SSO to RP applications using OAuth 2.0 implemented in accordance with RFC 8252 (the *OAuth
484 2.0 for Native Apps BCP*)
485 ▪ identity federation to RP applications using both SAML 2.0 and OIDC 1.0
486 ▪ MFA to mobile native and web applications using FIDO UAF and U2F

487 Though these capabilities are implemented as an integrated solution in this guide, organizational
488 requirements may dictate that only a subset of these capabilities be implemented. The modular
489 approach of this architecture is designed to support such use cases.

490 Additionally, the authors of this document recognize that PSFR organizations will have diverse IT
491 infrastructures, which may include previously purchased authentication, federation, or SSO capabilities,
492 and legacy technology. For this reason, Section 4.1 and [Appendix C](#) outline general considerations that
493 any organization may apply when designing an architecture tailored to organizational needs. [Section 4.2](#)
494 follows with considerations for implementing the architecture specifically developed by the NCCoE for
495 this project.

496 Organizations are encouraged to read [Section 3.2](#), [Section 3.3](#), [Section 3.5](#), and [Appendix B](#) to
497 understand context for this architecture design.

498 4.1 General Architectural Considerations

499 The PSFR community is large and diverse, comprising numerous state, local, tribal, and federal
500 organizations with individual missions and jurisdictions. PSFR personnel include police, firefighters,
501 emergency medical technicians, public health officials, and other skilled support personnel. There is no
502 single management or administrative hierarchy spanning the PSFR population. PSFR organizations
503 operate in a variety of environments with different technology requirements and wide variations in IT
504 staffing and budgets.

505 Cooperation and communication among PSFR organizations at multiple levels is crucial to addressing
506 emergencies that span organizational boundaries. Examples include coordination among multiple
507 services within a city (e.g., fire and police services), among different state law enforcement agencies to
508 address interstate crime, and among federal agencies like the Department of Homeland Security and its
509 state and local counterparts. This coordination is generally achieved through peer-to-peer interaction
510 and agreement or through federation structures, such as the National Identity Exchange Federation.
511 Where interoperability is achieved, it is the result of the cooperation of willing partners rather than
512 adherence to central mandates.

513 Enabling interoperability across the heterogeneous, decentralized PSFR user base requires a standards-based solution; a proprietary solution might not be uniformly adopted and could not be mandated. The
514 solution must also support identity federation and federated authentication, as user accounts and
515 authenticators are managed by several different organizations. The solution must also accommodate
516 organizations of different sizes, levels of technical capabilities, and budgets. Compatibility with the
517 existing capabilities of fielded identity systems can reduce the barrier to entry for smaller organizations.
518

519 Emergency response and other specialized work performed by PSFR personnel often require that they
520 wear personal protective equipment, such as gloves, masks, respirators, and helmets. This equipment
521 renders some authentication methods impractical or unusable. Fingerprint scanners cannot be used
522 with gloves, authentication using a mobile device camera to analyze the user's face or iris may be
523 hampered by masks or goggles, and entering complex passwords on small virtual keyboards is also
524 impractical for gloved users. In addition, PSFR work often involves urgent and hazardous situations
525 requiring the ability to quickly perform mission activities like driving, firefighting, and administering
526 urgent medical aid. Therefore, the solution must support a variety of authenticators in an interoperable
527 way so that individual user groups can select authenticators suited to their operational constraints.

528 In considering these requirements, the NCCoE implemented a standards-based architecture and
529 reference design. Section 4.1.1 through [Section 4.1.3](#) detail the primary standards used, while
530 [Appendix C](#) goes into great depth on architectural consideration when implementing these standards.

531 [4.1.1 SSO with OAuth 2.0, IETF RFC 8252, and AppAuth Open-Source Libraries](#)

532 SSO enables a user to authenticate once and to subsequently access different applications without
533 having to authenticate again. SSO on mobile devices is complicated by the sandboxed architecture,
534 which makes it difficult to share the session state with back-end systems between individual
535 applications. EMM vendors have provided solutions through proprietary SDKs, but this approach
536 requires integrating the SDK with each individual application and does not scale to a large and diverse
537 population, such as the PSFR user community.

538 OAuth 2.0 is an IETF standard that has been widely adopted to provide delegated authorization of
539 clients accessing representational state transfer interfaces, including mobile applications. OAuth 2.0,
540 when implemented in accordance with RFC 8252 (the *OAuth 2.0 for Native Apps* BCP), provides a
541 standards-based SSO pattern for mobile applications. The OpenID Foundation's AppAuth libraries [\[14\]](#)
542 can facilitate building mobile applications in full compliance with IETF RFC 8252, but any mobile
543 application that follows RFC 8252's core recommendation of using a shared external user-agent for the
544 OAuth authorization flow will have the benefit of SSO. OAuth considerations and recommendations are
545 detailed in [Section C.1 of Appendix C](#).

546 **4.1.2 Identity Federation**

547 SAML 2.0 [4] and OIDC 1.0 [5] are two standards that enable an application to redirect users to an IdP
548 for authentication and to receive an assertion of the user's identity and other optional attributes.
549 Federation is important in a distributed environment like the PSFR community, where user management
550 occurs in numerous local organizations. Federated authentication relieves users of having to create
551 accounts in each application that they need to access, and it frees application owners from managing
552 user accounts and credentials. OIDC is a more recent protocol, but many organizations have existing
553 SAML deployments. The architecture supports both standards to facilitate adoption without requiring
554 upgrades or modifications to existing SAML IdPs. Federation considerations and recommendations are
555 detailed in [Section C.2 of Appendix C](#).

556 **4.1.3 FIDO and Authenticator Types**

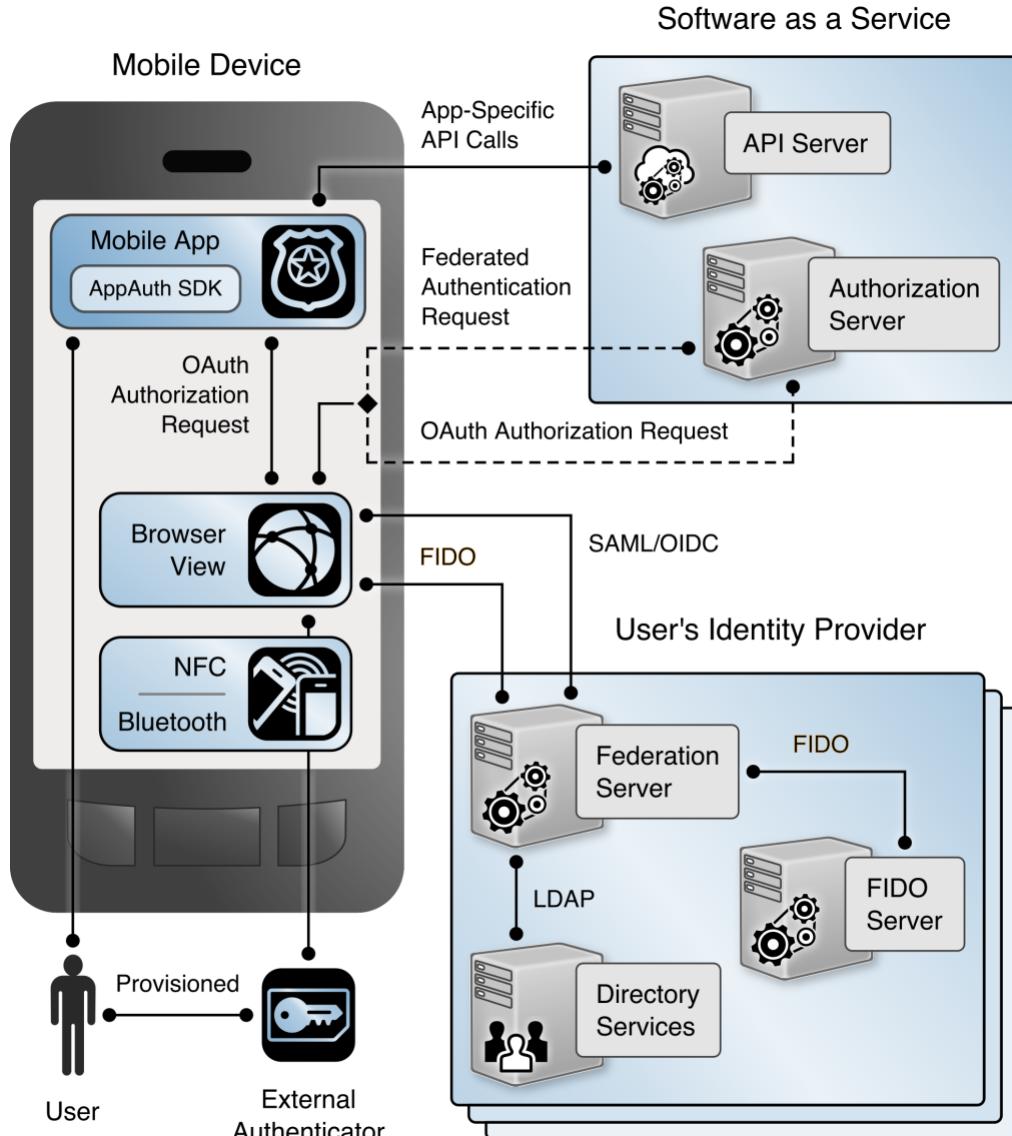
557 When considering MFA implementations, PSFR organizations should carefully consider organizationally
558 defined authenticator requirements. These requirements are detailed in [Section C.3 of Appendix C](#).

559 FIDO provides a standard framework within which vendors have produced a wide range of interoperable
560 biometric, hardware, and software authenticators. This will enable PSFR organizations to choose
561 authenticators suitable to their operational constraints. The FIDO Alliance has published specifications
562 for two types of authenticators based on UAF and U2F. These protocols operate agnostic of the FIDO
563 authenticator, allowing PSOs to choose any FIDO-certified authenticator that meets operational
564 requirements and to implement it with this solution. The protocols, FIDO key registration, FIDO
565 authenticator attestation, and FIDO deployment considerations are also detailed in [Section C.3 of](#)
566 [Appendix C](#).

567 **4.2 High-Level Architecture**

568 The NCCoE implemented both FIDO UAF and U2F for this project. The high-level architecture varies
569 somewhat between the two implementations. Figure 4-1 depicts the interactions between the key
570 elements of the build architecture with the U2F implementation.

571 Figure 4-1 High-Level U2F Architecture

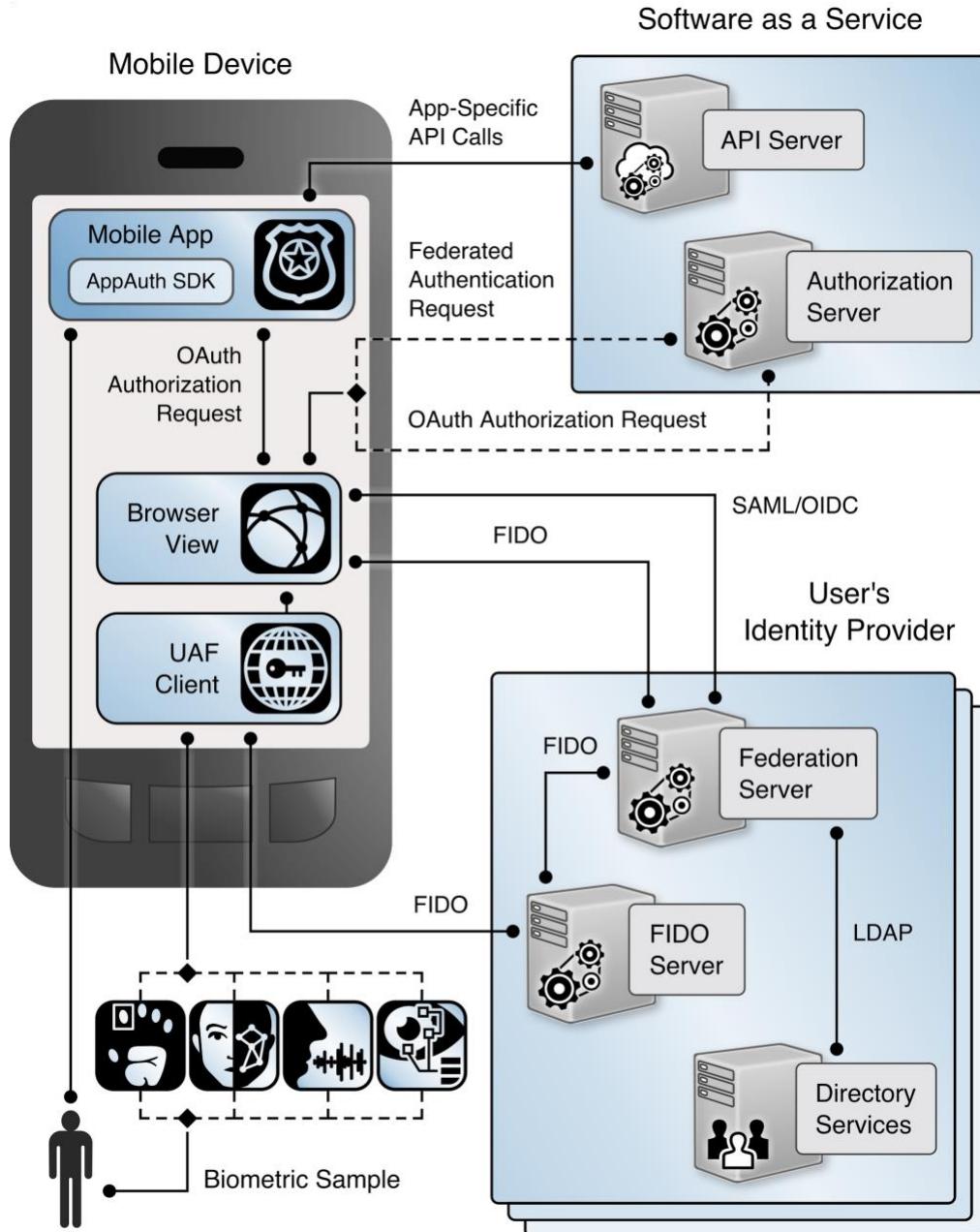


572

573 On the mobile device, the mobile application includes the OpenID Foundation's AppAuth library, which
 574 streamlines implementation of the OAuth client functionality in accordance with the IETF RFC 8252,
 575 *OAuth 2.0 for Native Apps*, guidance. AppAuth orchestrates the authorization request flow by using the
 576 device's native browser capabilities, including in-application browser tabs on devices that support them.
 577 The mobile device also supports the two FIDO authentication schemes, UAF and U2F. UAF typically
 578 involves an internal (on-device) authenticator that authenticates the user directly to the device by using
 579 biometrics, other hardware capabilities, or a software client. U2F typically involves an external hardware
 580 authenticator token, which communicates with the device over NFC or Bluetooth.

581 Figure 4-2 shows the corresponding architecture view with the FIDO UAF components.

582 **Figure 4-2 High-Level UAF Architecture**



583 User

584 The SaaS provider hosts application servers that provide APIs consumed by mobile applications, as well
585 as an OAuth AS. The browser on the mobile device connects to the AS to initiate the OAuth

586 authorization code flow. The AS redirects the browser to the IdP of the user's organization to
587 authenticate the user. Once the user has authenticated, the AS will issue an access token, which is
588 returned to the mobile application through a browser redirect and can be used to authorize requests to
589 the application servers.

590 The user's IdP includes a federation server that implements SAML or OIDC, directory services containing
591 user accounts and attributes, and a FIDO authentication service that can issue authentication challenges
592 and validate the responses that are returned from FIDO authenticators. The FIDO authentication service
593 may be built into the IdP but is more commonly provided by a separate server.

594 A SaaS provider may provide multiple applications, which may be protected by the same AS. For
595 example, Motorola Solutions provides both the PSX Mapping and PSX Messaging applications, which are
596 protected by a shared AS. Users may also use services from different SaaS providers, which would have
597 separate ASes. This build architecture can provide SSO between applications hosted by a single SaaS
598 provider as well as across applications provided by multiple SaaS vendors.

599 Support for these two scenarios differs between the Android and iOS platforms. Today, U2F is not
600 supported on iOS devices, while UAF is supported on both Android and iOS. The build team has only
601 built and tested the U2F implementation on Android devices.

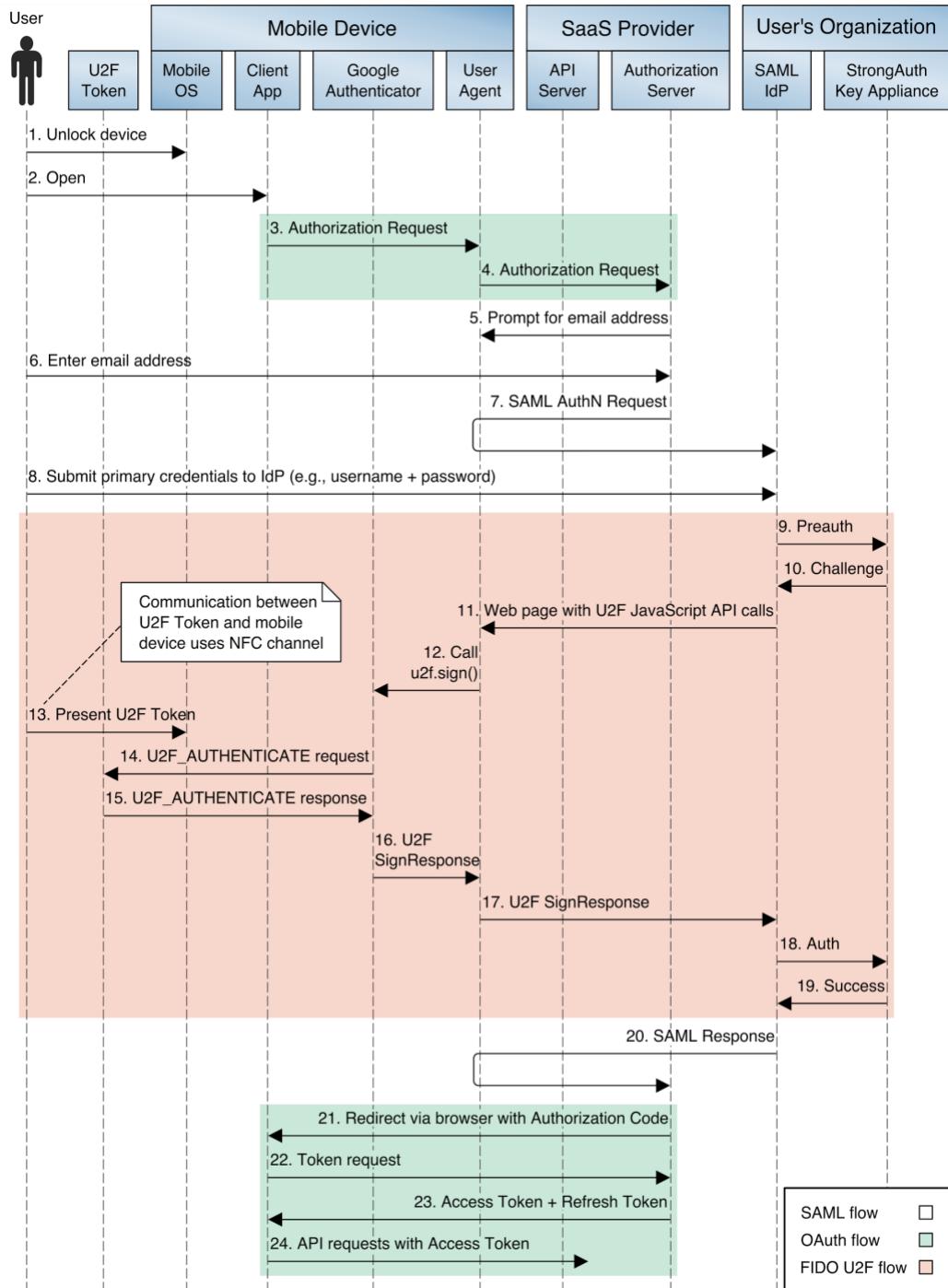
602 **4.3 Detailed Architecture Flow**

603 The mobile SSO lab implementation demonstrates two authentication flows: one in which the user
604 authenticates to a SAML IdP with a YubiKey Neo U2F token and a PIN, and one in which the user
605 authenticates to an OIDC IdP by using UAF with a fingerprint. These pairings of federation and
606 authentication protocols are purely arbitrary; U2F could just as easily be used with OIDC, for example.

607 **4.3.1 SAML and U2F Authentication Flow**

608 The authentication flow using SAML and U2F is depicted in Figure 4-3. As explained in Section 4.2, at the
609 time of publication this implementation is not supported on iOS devices. This figure depicts the message
610 flows among different components on the mobile device or hosted by the SaaS provider or user
611 organization. In the figure, colored backgrounds differentiate the SAML, OAuth, and FIDO U2F protocol
612 flows. Prior to this authentication flow, the user must have registered a FIDO U2F token with the IdP,
613 and the AS and IdP must have exchanged metadata and established an RP trust.

614 Figure 4-3 SAML and U2F Sequence Diagram



615

616 The detailed steps are as follows:

- 617 1. The user unlocks the mobile device. Any form of lock-screen authentication can be used; it is not
618 directly tied to the subsequent authentication or authorization.
- 619 2. The user opens a mobile application that connects to the SaaS provider's back-end services. The
620 mobile application determines that an OAuth token is needed. This may occur because the
621 application has no access or refresh tokens cached or it has an existing token known to be
622 expired based on token metadata, or it may submit a request to the API server with a cached
623 bearer token and receive an HTTP 401 status code in the response.
- 624 3. The mobile application initiates an OAuth authorization request using the authorization code
625 flow by invoking the system browser (or an in-application browser tab) with the uniform
626 resource locator (URL) of the SaaS provider AS's authorization end point.
- 627 4. The browser submits the request to the AS over a hypertext transfer protocol secure (https)
628 connection. This begins the OAuth 2 authorization flow.
- 629 5. The AS returns a page that prompts for the user's email address.
- 630 6. The user submits the email address. The AS uses the domain of the email address for IdP
631 discovery. The user needs to specify the email address only one time; the address is stored in a
632 cookie in the device browser and will be used to automatically determine the user's IdP on
633 subsequent visits to the AS.
- 634 7. The AS redirects the device browser to the user's IdP with a SAML authentication request. This
635 begins the SAML authentication flow.
- 636 8. The IdP returns a login page. The user submits a username and PIN. The IdP validates these
637 credentials against the directory service. If the credentials are invalid, the IdP redirects back to
638 the login page with an error message and prompts the user to authenticate again. If the
639 credentials are valid, the IdP continues to step 9.
- 640 9. The IdP submits a "preauth" API request to the StrongKey SKCE server. The preauth request
641 includes the authenticated username obtained in step 8. This begins the FIDO U2F
642 authentication process.
- 643 10. The SKCE responds with a U2F challenge that must be signed by the user's registered key in the
644 U2F token to complete authentication. If the user has multiple keys registered, the SKCE returns
645 a challenge for each key so that the user can authenticate with any registered authenticator.

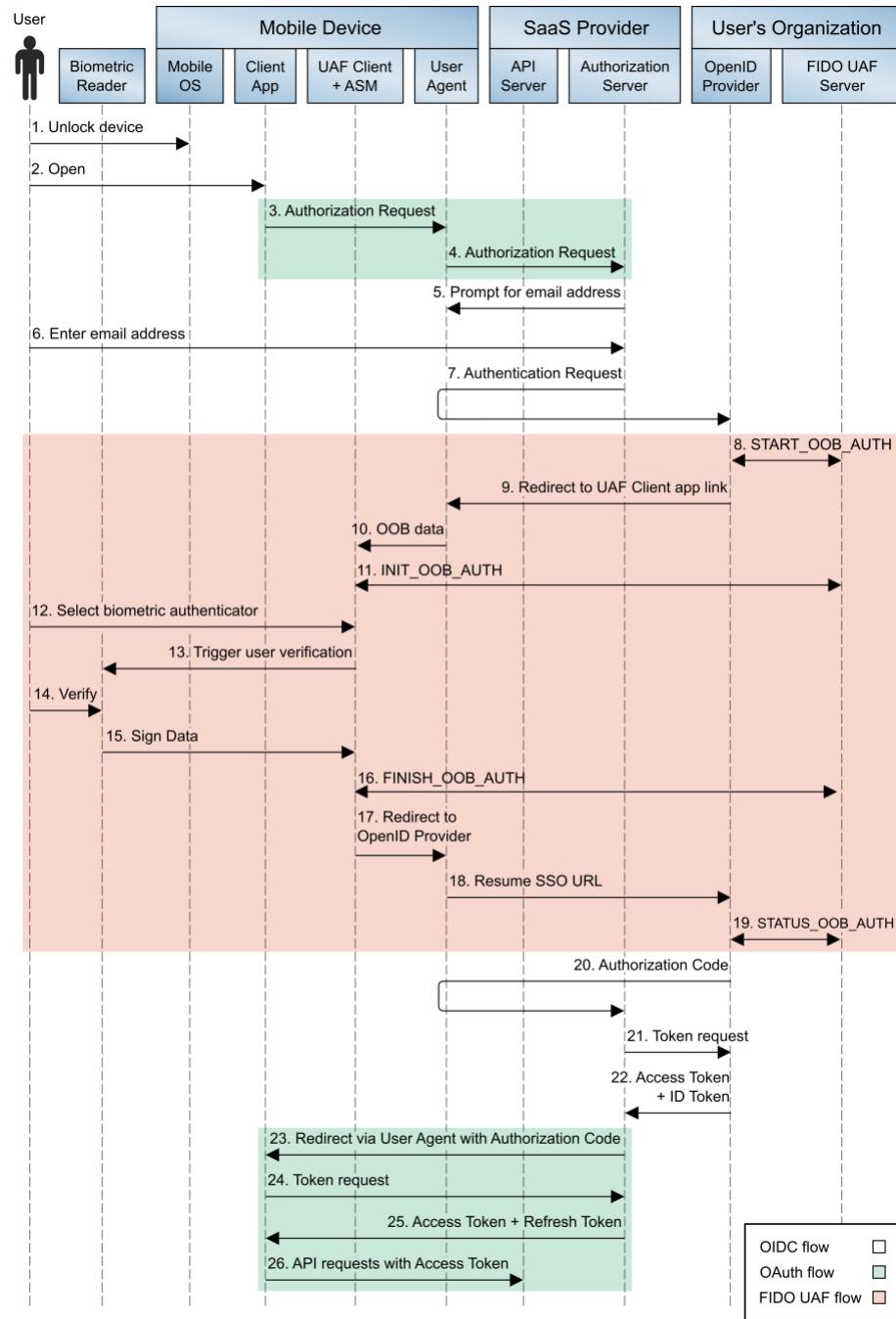
- 646 11. The IdP returns a page to the user’s browser that includes Google’s JavaScript U2F API and the
647 challenge obtained from the SKCE in step 10. The user taps a button on the page to initiate U2F
648 authentication, which triggers a call to the u2f.sign JavaScript function.
- 649 12. The u2f.sign function invokes the Google Authenticator application, passing it the challenge, the
650 appId (typically the domain name of the IdP), and an array of the user’s registered key.
- 651 13. Google Authenticator prompts the user to hold the U2F token against the NFC radio of the
652 mobile device, which the user does.
- 653 14. Google Authenticator connects to the U2F token over the NFC channel and sends an applet
654 selection command to activate the U2F applet on the token. Google Authenticator then submits
655 a U2F_AUTHENTICATE message to the token.
- 656 15. Provided that the token has one of the keys registered at the IdP, it signs the challenge and
657 returns the signature in an authentication success response over the NFC channel.
- 658 16. Google Authenticator returns the signature to the browser in a SignResponse object.
- 659 17. The callback script on the authentication web page returns the SignResponse object to the IdP.
- 660 18. The IdP calls the “authenticate” API on the SKCE, passing the SignResponse as a parameter.
- 661 19. The SKCE validates the signature of the challenge by using the registered public key and verifies
662 that the appId matches the IdP’s and that the response was received within the configured time-
663 out. The API returns a response to the IdP, indicating success or failure and any error messages.
664 This concludes the U2F authentication process; the user has now authenticated to the IdP,
665 which sets a session cookie.
- 666 20. The IdP returns a SAML response indicating the authentication success or failure to the AS
667 through a browser redirect. If authentication has succeeded, the response will include the user’s
668 identifier and, optionally, additional attribute assertions. This concludes the SAML
669 authentication flow. The user is now authenticated to the AS, which sets a session cookie.
670 Optionally, the AS could prompt the user to approve the authorization request, displaying the
671 scopes of access being requested at this step.
- 672 21. The AS sends a redirect to the browser with the authorization code. The target of the redirect is
673 the mobile application’s redirect_uri, a link that opens in the mobile application through a
674 mechanism provided by the mobile OS (e.g., custom request scheme or Android AppLink).
- 675 22. The mobile application extracts the authorization code from the URL and submits it to the AS’s
676 token end point.

- 677 23. The AS responds with an access token and, optionally, a refresh token that can be used to obtain
678 an additional access token when the original token expires. This concludes the OAuth
679 authorization flow.
- 680 24. The mobile application can now submit API requests to the SaaS provider's back-end services by
681 using the access token in accordance with the bearer token authorization scheme defined in
682 RFC 6750, *The OAuth 2.0 Authorization Framework: Bearer Token Usage* [\[15\]](#).

683 **4.3.2 OpenID Connect and UAF Authentication Flow**

684 The authentication flow involving OIDC and UAF is depicted in Figure 4-4.

685 **Figure 4-4 OIDC and UAF Sequence Diagram**



686

687 Figure 4-4 uses the same conventions and color coding as the earlier SAML/U2F diagram (Figure 4-3) to
688 depict components on the device, at the SaaS provider, and at the user's organization. Prior to this
689 authentication flow, the user must have registered a FIDO UAF authenticator with the IdP, and the AS
690 must be registered as an OIDC client at the IdP. The detailed steps are listed below. For ease of
691 comparison, steps that are identical to the corresponding step in Figure 4-3 are shown in italics.

- 692 1. *The user unlocks the mobile device. Any form of lock-screen authentication can be used; it is not*
693 *directly tied to the subsequent authentication or authorization.*
- 694 2. *The user opens a mobile application that connects to the SaaS provider's back-end services. The*
695 *mobile application determines that an OAuth token is needed. This may occur because the*
696 *application has no access or refresh tokens cached or it has an existing token known to be*
697 *expired based on token metadata, or it may submit a request to the API server with a cached*
698 *bearer token and receive an HTTP 401 status code in the response.*
- 699 3. *The mobile application initiates an OAuth authorization request by using the authorization code*
700 *flow by invoking the system browser (or an in-application browser tab) with the URL of the SaaS*
701 *provider AS's authorization end point.*
- 702 4. *The in-application browser tab submits the request to the AS over an https connection. This*
703 *begins the OAuth 2 authorization flow.*
- 704 5. *The AS returns a page that prompts for the user's email address.*
- 705 6. *The user submits the email address. The AS uses the domain of the email address for IdP*
706 *discovery. The user needs to specify the email address only one time; the address is stored in a*
707 *cookie in the device browser and will be used to automatically determine the user's IdP on*
708 *subsequent visits to the AS.*
- 709 7. The AS redirects the device browser to the user's IdP with an OIDC authentication request. This
710 begins the OIDC authentication flow.
- 711 8. The IdP submits a START_OOB_AUTH request to the UAF authentication server. The server
712 responds with a data structure containing the necessary information for a UAF client to initiate
713 an Out-of-Band (OOB) authentication, including a transaction identifier linked to the user's
714 session at the IdP.
- 715 9. The IdP returns an HTTP redirect to the browser. The redirect target URL is an application link
716 that will pass the OOB data to the Nok Nok Labs Passport application on the device.
- 717 10. The Nok Nok Passport application opens and extracts the OOB data from the application link
718 URL.

- 719 11. Passport sends an INIT_OOB_AUTH request to the UAF authentication server, including the OOB
720 data and a list of authenticators available on the device that the user has registered for use at
721 the IdP. The server responds with a set of UAF challenges for the registered authenticators.
- 722 12. If the user has multiple registered authenticators (e.g., fingerprint and voice authentication),
723 Passport prompts the user to select which authenticator to use.
- 724 13. Passport activates the authenticator, which prompts the user to perform the required steps for
725 verification. For example, if the selected authenticator is the Android Fingerprint authenticator,
726 the standard Android fingerprint user interface (UI) overlay will pop over the browser and
727 prompt the user to scan an enrolled fingerprint. The authenticator UI may be presented by
728 Passport (for example, the PIN authenticator), or it may be provided by an OS component such
729 as Apple Touch ID or Face ID.
- 730 14. The user completes the biometric scan or other user verification activity. Verification occurs
731 locally on the device; biometrics and secrets are not transmitted to the server.
- 732 15. The authenticator signs the UAF challenge by using the private key that was created during
733 initial UAF enrollment with the IdP. The authenticator returns control to the Passport
734 application through an application link with the signed UAF challenge.
- 735 16. The Passport application sends a FINISH_OOB_AUTH API request to the UAF authentication
736 server. The server extracts the username and registered public key and validates the signed
737 response. The server can also validate the authenticator's attestation signature and check that
738 the security properties of the authenticator satisfy the IdP's security policy. The server caches
739 the authentication result.
- 740 17. The Passport application closes, returning control to the browser, which is redirected to the
741 "resume SSO" URL at the IdP. This URL is defined on the Ping server to enable multistep
742 authentication flows and allow the browser to be redirected back to the IdP after completing
743 required authentication steps with another application.
- 744 18. The browser requests the Resume SSO URL at the IdP.
- 745 19. The IdP sends a STATUS_OOB_AUTH API request to the UAF authentication server. The UAF
746 server responds with the success/failure status of the out-of-band authentication and any
747 associated error messages. (Note: The IdP begins sending STATUS_OOB_AUTH requests
748 periodically, following step 9 in the flow, and continues to do so until a final status is returned or
749 the transaction times out.) This concludes the UAF authentication process; the user has now
750 authenticated to the IdP, which sets a session cookie.
- 751 20. The IdP returns an authorization code to the AS through a browser redirect.

- 752 21. The AS submits a token request to the IdP's token end point, authenticating with its credentials
753 and including the authorization code.
- 754 22. The IdP responds with an identification (ID) token and an access token. The ID token includes
755 the user's identifier and, optionally, additional attribute assertions. The access token can
756 optionally be used to request additional user claims at the IdP's user information end point. This
757 concludes the OIDC authentication flow. The user is now authenticated to the AS, which sets a
758 session cookie. Optionally, the AS could prompt for the user to approve the authorization
759 request, displaying the scopes of access being requested at this step.
- 760 23. *The AS sends a redirect to the browser with the authorization code. The target of the redirect is
761 the mobile application's redirect_uri, a link that opens in the mobile application through a
762 mechanism provided by the mobile OS (e.g., custom request scheme or Android AppLink).*
- 763 24. *The mobile application extracts the authorization code from the URL and submits it to the AS's
764 token end point.*
- 765 25. *The AS responds with an access token and, optionally, a refresh token that can be used to obtain
766 an additional access token when the original token expires. This concludes the OAuth
767 authorization flow.*
- 768 26. *The mobile application can now submit API requests to the SaaS provider's back-end services by
769 using the access token in accordance with the bearer token authorization scheme.*
- 770 Both authentication flows end with a single application obtaining an access token to access back-end
771 resources. At this point, traditional OAuth token life-cycle management would begin. Access tokens
772 have an expiration time. Depending on the application's security policy, refresh tokens may be issued
773 along with the access token and used to obtain a new access token when the initial token expires.
774 Refresh tokens and access tokens can continue to be issued in this manner for as long as the security
775 policy allows. When the current access token has expired and no additional refresh tokens are available,
776 the mobile application would submit a new authorization request to the AS.
- 777 Apart from obtaining an access token, the user has established sessions with the AS and IdP that can be
778 used for SSO.
- 779 Implementation details for this scenario were slightly different on iOS and Android devices. On Android
780 devices, a Chrome Custom Tab was used as the user-agent. On iOS, however, the team encountered
781 issues using the custom tabs implementation in iOS 12 (provided by the ASWebAuthenticationSession
782 API) in conjunction with Passport. At step 17 in the above sequence, where the Passport application
783 should close and control should return to the in-application browser tab, instead a second Safari
784 window opened, and the user was prompted again to authenticate using Passport. The team
785 determined that ASWebAuthenticationSession does not seem to support opening a different application
786 like Passport and then returning to the same ASWebAuthenticationSession instance once the other

787 application closes. This issue was resolved by configuring AppAuth to use Safari instead of
788 ASWebAuthenticationSession.

789 **4.4 Single Sign-On with the OAuth Authorization Flow**

790 When multiple applications invoke a common user-agent to perform the OAuth authorization flow, the
791 user-agent maintains the session state with the AS and IdP. In the build architecture, this can enable SSO
792 in two scenarios.

793 In the first case, assume that a user has launched a mobile application, has been redirected to an IdP to
794 authenticate, and has completed the OAuth flow to obtain an access token. Later, the user launches a
795 second application that connects to the same AS used by the first application. The application will
796 initiate an authorization request using the same user-agent as the first application. Provided that the
797 user has not logged out at the AS, this request will be sent with the session cookie that was established
798 when the user authenticated in the previous authorization flow. The AS will recognize the user's active
799 session and issue an access token to the second application without requiring the user to authenticate
800 again.

801 In the second case, again assume that the user has completed an OAuth flow, including authentication
802 to an IdP, while launching the first application. Later, the user launches a second application that
803 connects to an AS that is different from the first application. Again, the second application initiates an
804 authorization request using the same user-agent as the first application. The user has no active session
805 with the second AS, so the user-agent is redirected to the IdP to obtain an authentication assertion.
806 Provided that the user has not logged out at the IdP, the authentication request will include the
807 previously established session cookie, and the user will not be required to authenticate again at the IdP.
808 The IdP will return an assertion to the AS, which will then issue an access token to the second
809 application.

810 This architecture can also provide SSO across native and web applications. If the web application is an RP
811 to the same SAML or OIDC IdP used in the authentication flow described above, the application will
812 redirect the browser to the IdP and resume the user's existing session without the need to
813 reauthenticate, provided that the browser used to access the web application is the same one used in
814 the authorization flow described above. For example, if a Google Chrome Custom Tab is used in the
815 native-application OAuth flow, then accessing the web application in Chrome will provide a shared
816 cookie store and SSO. If the web application uses the OAuth 2.0 implicit grant, then SSO could follow
817 either of the above workflows, depending on whether the user is already authenticated at the AS used
818 by the application.

819 When applications use embedded web views instead of the system browser or in-application tabs for
820 the OAuth authorization flow, each individual application's web view has its own cookie store, so there
821 is no continuity of the session state as the user transitions from one application to another, and the user
822 must authenticate each time.

4.5 Application Developer Perspective of the Build

The following paragraphs provide takeaways from an application developer's perspective regarding the experience of the build team, inclusive of FIDO, the AppAuth library, PKCE, and Chrome Custom Tabs.

AppAuth was integrated as described in [Section C.1 of Appendix C](#). From an application developer perspective, the primary emphasis in the build was integrating AppAuth. The authentication technology was basically transparent to the developer. In fact, the native application developers for this project had no visibility to the FIDO U2F or UAF integration. This transparency was achieved through the AppAuth pattern of delegating the authentication process to the in-application browser tab capability of the OS.

Other application developer effects are listed below:

- Several pieces of information must be supplied by an application in the OAuth authorization request, such as the scope and the client ID, which an OAuth AS might use to apply appropriate authentication policy. These details are obtained during the OAuth client registration process with the AS.
- The ability to support multiple IdPs without requiring any hard-coding of IdP URLs in the application itself was achieved by using hypertext markup language (HTML) forms hosted by the IdP to collect information from end users (e.g., domain) during login, which was used to perform IdP discovery.

4.6 Identity Provider Perspective of the Build

The IdP is responsible for account and attribute creation and maintenance, as well as credential provisioning, management, and deprovisioning. Some IdP concerns for this architecture are listed below:

- Enrollment/registration of authenticators: IdPs should consider the enrollment process and life-cycle management for MFA. For this NCCoE project, FIDO UAF enrollment was launched by the user via tapping a native enrollment application (Nok Nok Labs' Passport application). During user authentication, the same application (Passport) was invoked programmatically (via AppLink) to perform FIDO authentication. In a production implementation, the IdP would need to put processes in place to enroll, retire, or replace authenticators when needed. A process for responding when authenticators are lost or stolen is particularly important to prevent unauthorized access.
- For UAF, a FIDO UAF client must be installed (e.g., we installed Nok Nok Labs' NNL Passport).
- For U2F, download and install Google Authenticator (or equivalent) because mobile browsers do not support FIDO U2F 1.1 natively (as do some desktop browsers). This situation is evolving with ratification of the World Wide Web Consortium Web Authentication (WebAuthn) standard [\[16\]](#) and mobile browser support for it. For implementations supporting U2F integration in the browser, such as the one described in this practice guide, Google Authenticator is still required

858 on Android devices. For implementations using WebAuthn, native browser support may
859 eliminate the need for Google Authenticator.

860 **4.7 Token and Session Management**

861 RP application owners have two separate areas of concern when it comes to token and session
862 management. They have authorization tokens to manage on the client side, and the identity
863 tokens/sessions to receive and manage from the IdP side. Each of these functions has its own separate
864 concerns and requirements.

865 When dealing with the native application's access to RP application data, RP operators need to make
866 sure that appropriate authorization is in place. The architecture in [Section 4.2](#) uses OAuth 2.0 and
867 authorization tokens for this purpose, following the guidance from IETF RFC 8252. Native-application
868 clients present a special challenge, as mentioned earlier, especially when it comes to protecting the
869 authorization code being returned to the client. To mitigate a code interception threat, RFC 8252
870 requires that both clients and servers use PKCE for public native-application clients. ASes should reject
871 authorization requests from native applications that do not use PKCE. The lifetime of the authorization
872 tokens depends on the use case, but the general recommendation from the OAuth working group is to
873 use short-lived access tokens and long-lived refresh tokens. The reauthentication requirements in NIST
874 SP 800-63B [\[10\]](#) can be used as guidance for maximum refresh token lifetimes at each authenticator
875 assurance level. All security considerations from RFC 8252 apply here as well, such as making sure that
876 attackers cannot easily guess any of the token values or credentials.

877 The RP may directly authenticate the user, in which case all of the current best practices for web session
878 security and protecting the channel with Transport Layer Security (TLS) apply. However, if there is
879 delegated or federated authentication via a third-party IdP, then the RP must also consider the
880 implications for managing the identity claims received from the IdP, whether it be an ID token from an
881 OIDC provider or a SAML assertion from a SAML IdP. This channel is used for authentication of the user,
882 which means that potential PII may be obtained. Care must be taken to obtain user consent prior to
883 authorization for release and use of this information in accordance with relevant regulations. If OIDC is
884 used for authentication to the RP, then all of the OAuth 2.0 security applies again here. In all cases, all
885 channels between parties must be protected with TLS encryption.

886 **5 Security Characteristic Analysis**

887 The purpose of the security characteristic analysis is to understand the extent to which the project
888 meets its objective of demonstrating MFA and mobile SSO for native and web applications. In addition, it
889 seeks to document the security benefits and drawbacks of the example solution.

5.1 Assumptions and Limitations

This security characteristics analysis is focused on the specific design elements of the build, consisting of MFA, SSO, and federation implementation. It discusses some elements of application development, but only the aspects that directly interact with the SSO implementation. It does not focus on potential underlying vulnerabilities in OSes, application run times, hardware, or general secure coding practices. It is assumed that risks to these foundational components are managed separately (e.g., through asset and patch management). As with any implementation, all layers of the architecture must be appropriately secured, and it is assumed that implementers will adopt standard security and maintenance practices to the elements not specifically addressed here.

This project did not include a comprehensive test of all security components or “red team” penetration testing or adversarial emulation. Cybersecurity is a rapidly evolving field where new threats and vulnerabilities are continually discovered. Therefore, this security guidance cannot be guaranteed to identify every potential weakness of the build architecture. It is assumed that implementers will follow risk management procedures as outlined in the NIST Risk Management Framework.

5.2 Threat Analysis

The following subsections describe how the build architecture addresses the threats discussed in [Section 3.5](#).

5.2.1 Mobile Ecosystem Threat Analysis

In [Section 3.5.2](#), we introduced the MTC, described the 32 categories of mobile threats that it covers, and highlighted the three categories that this practice guide addresses: [Vulnerable Applications](#), [Authentication: User or Device to Network](#), and [Authentication: User or Device to Remote Service](#).

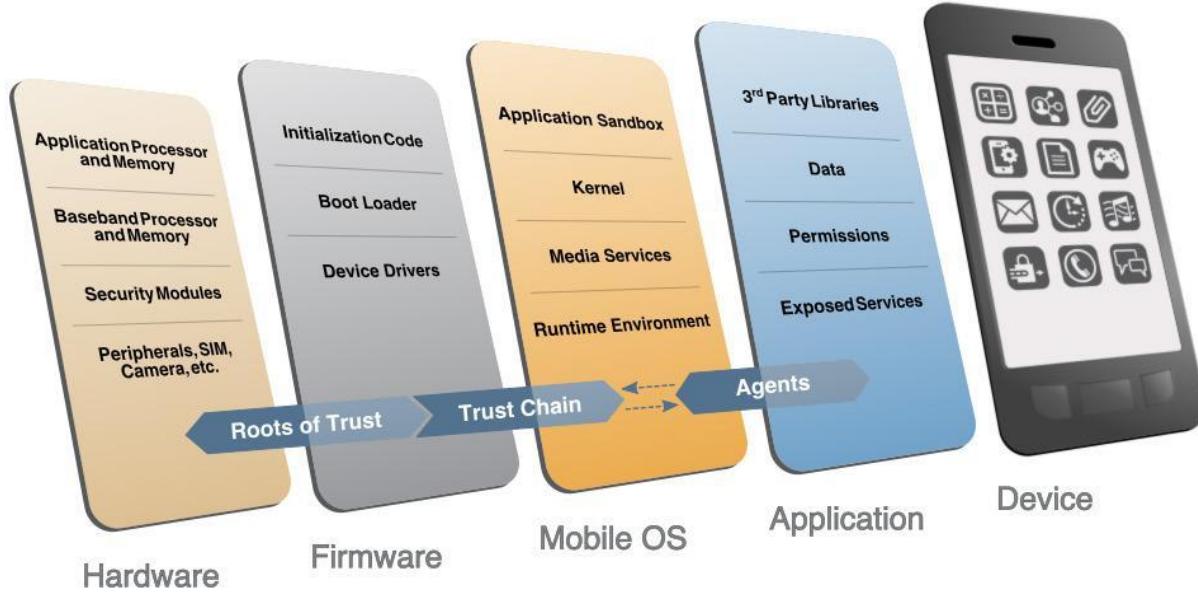
At the time of this writing, these categories encompass 18 entries in the MTC. However, the MTC is a living catalog, which is continually being updated. Instead of addressing each threat, we describe in general how these types of threats are mitigated by the architecture laid out in this practice guide:

- Use encryption for data in transit: The IdP and AS enforce https encryption by default, which the application is required to use during SSO authentication.
- Use newer mobile platforms: Volume C of this guide (NIST SP 1800-13C) calls for using at least Android 5.0 or iOS 8.0 or newer, which mitigates weaknesses of older versions (e.g., applications can access the system log in Android 4.0 and older).
- Use built-in browser features: The AppAuth for Android library utilizes the Chrome Custom Tabs feature, which activates the device’s native browser. This allows the application to leverage built-in browser features, such as identifying and avoiding known malicious web pages. AppAuth for iOS supports using the SFSafariViewController and SFAuthenticationSession APIs or the Safari browser.

- 924 ▪ Avoid hard-coded secrets: The AppAuth guidance recommends and supports the use of PKCE.
925 This allows developers to avoid using a hard-coded OAuth client secret.
- 926 ▪ Avoid logging sensitive data: The AppAuth library, which handles the OAuth 2 flow, does not log
927 any sensitive data.
- 928 ▪ Use sound authentication practices: By using SSO, the procedures outlined in this guide allow
929 application developers to rely on the IdP's implementation of authentication practices, such as
930 minimum length and complexity requirements for passwords, maximum authentication
931 attempts, and periodic reset requirements. In addition, the IdP can introduce new
932 authenticators without any downstream effect to applications.
- 933 ▪ Use sound token management practices: Again, this guide allows application developers to rely
934 on the IdP's implementation of authorization tokens and good management practices, such as
935 replay-resistance mechanisms and token expirations.
- 936 ▪ Use two-factor authentication: Both FIDO U2F and UAF, as deployed in this build architecture,
937 provide multifactor cryptographic user authentication. The U2F implementation requires the
938 user to authenticate with a password or PIN and with a single-factor cryptographic token.
939 However, the UAF implementation utilizes a key pair stored in the device's hardware-backed key
940 store that is unlocked through user verification consisting of a biometric (e.g., fingerprint or
941 voice match) or a password or PIN.
- 942 ▪ Protect cryptographic keys: FIDO U2F and UAF authentication leverage public key cryptography.
943 In this architecture, U2F private keys are stored external to the mobile device in a hardware-
944 secure element on a YubiKey Neo. UAF private keys are stored on the mobile device's hardware-
945 backed key store. These private keys are never sent to external servers.
- 946 ▪ Protect biometric templates: When using biometric authentication mechanisms, organizations
947 should consider storage and use of user biometric templates. This architecture relies on the
948 native biometric mechanisms implemented by modern mobile devices and OSes, which verify
949 biometric templates locally and store them in protected storage.

950 To fully address these threats and threats in other MTC categories, additional measures should be taken
951 by all parties involved in the mobile ecosystem: the mobile device user, the enterprise, the network
952 operator, the application developer, and the OEM. A figure depicting this ecosystem in total is shown in
953 [Section 3.5.2](#). In addition, the mobile platform stack should be understood in great detail to fully assess
954 the threats that may be applicable. An illustration of this stack, taken from NIST Interagency Report
955 8144 [9], is shown in Figure 5-1.

956 **Figure 5-1 Mobile Device Technology Stack**



957

958 Several tools, techniques, and best practices are available to mitigate these other threats. EMM
 959 software can allow enterprises to manage devices more fully and to gain a better understanding of
 960 device health; one example of this is detecting whether a device has been *rooted* or *jailbroken*, which
 961 compromises the security architecture of the entire platform. Application security-vetting software
 962 (commonly known as app-vetting software) can be utilized to detect vulnerabilities in first-party
 963 applications and to discover potentially malicious behavior in third-party applications. Using app-vetting
 964 software in conjunction with EMM software prevents the installation of unauthorized applications and
 965 reduces the attack surface of the platform. For more guidance on these threats and mitigations, refer to
 966 the [MTC](#) and NIST Interagency Report 8144 [9].

967 5.2.2 Authentication and Federation Threat Analysis

968 [Section 3.5.3](#) discussed threats specific to authentication and federation systems, which are cataloged in
 969 NIST SP 800-63-3 [17]. MFA, provided in the build architecture by FIDO U2F and UAF, is designed to
 970 mitigate several authentication risks:

- 971 ▪ Theft of physical authenticator: Possessing an authenticator, which could be a YubiKey (in the
 972 case of U2F) or the mobile device itself (in the case of UAF), does not in itself enable an attacker
 973 to impersonate the user to an RP or IdP. Additional knowledge or a biometric factor is needed to
 974 authenticate.
- 975 ▪ Eavesdropping: Some MFA solutions, including many onetime password (OTP) implementations,
 976 are vulnerable to eavesdropping attacks. FIDO implements cryptographic authentication, which
 977 does not involve transmission of secrets over the network.

- 978 ▪ Social engineering: A typical social engineering exploit involves impersonating a system
979 administrator or other authority figure under some pretext to convince users to disclose their
980 passwords over the phone, but this comprises only a single authentication factor.
981 ▪ Online guessing: Traditional password authentication schemes may be vulnerable to online
982 guessing attacks, though lockout and throttling policies can reduce the risk. Cryptographic
983 authentication schemes are not vulnerable to online guessing.

984 FIDO also incorporates protections against phishing and pharming attacks. When a FIDO authenticator is
985 registered with an RP, a new key pair is created and associated with the RP's application ID, which is
986 derived from the domain name in the URL where the registration transaction was initiated. During
987 authentication, the application ID is again derived from the URL of the page that is requesting
988 authentication, and the authenticator will sign the authentication challenge only if a key pair has been
989 registered with the matching application ID. The FIDO facets specification enables sites to define a list of
990 domain names that should be treated as a single application ID to accommodate service providers that
991 span multiple domain names, such as google.com and gmail.com.

992 The application ID verification effectively prevents the most common type of phishing attack, in which
993 the attacker creates a new domain and tricks users into visiting that domain instead of an intended RP
994 where the user has an account. For example, an attacker might register a domain called "google-
995 accts.com" and send emails with a pretext to get users to visit the site, such as a warning that the user's
996 account will be disabled unless some action is taken. The attacker's site would present a login screen
997 identical to Google's login screen to obtain the user's password (and OTP, if enabled) credentials and to
998 use them to impersonate the user to the real Google services. With FIDO, the authenticator would not
999 have an existing key pair registered under the attacker's domain, so the user would be unable to return
1000 a signed FIDO challenge to the attacker's site. If the attacker could convince the user to register the FIDO
1001 authenticator with the malicious site and then sign an authentication challenge, the signed FIDO
1002 assertion could not be used to authenticate to Google because the RP can also verify the application ID
1003 associated with the signed challenge, and it would not be the expected ID.

1004 A more advanced credential theft attack involves an active man in the middle that can intercept the
1005 user's requests to the legitimate RP and act as a proxy between the two. To avoid TLS server certificate
1006 validation errors, in this case, the attacker must obtain a TLS certificate for the legitimate RP site that is
1007 trusted by the user's device. This could be accomplished by exploiting a vulnerability in a commercial
1008 certificate authority; it presents a high bar for the attacker but is not unprecedented. Application ID
1009 validation is not sufficient to prevent this attacker from obtaining an authentication challenge from the
1010 RP, proxying it to the user, and using the signed assertion that it gets back from the user to authenticate
1011 to the RP. To prevent this type of attack, the FIDO specifications permit token binding to protect the
1012 signed assertion that is returned to the RP by including information in the assertion about the TLS
1013 channel over which it is being delivered. If there is a man in the middle (or a proxy of any kind) between
1014 the user and the RP, the RP can detect it by examining the token-binding message included in the
1015 assertion and comparing it with the TLS channel over which it was received. Token binding is not widely

1016 implemented today, but with finalization of the token-binding specification in RFC 8471 [18] and related
1017 RFCs, adoption is expected to increase.

1018 Many of the federation threats discussed in [Section 3.5.3](#) can be addressed by signing assertions,
1019 ensuring their integrity and authenticity. An encrypted assertion can also provide multiple protections,
1020 preventing disclosure of sensitive information contained in the assertion and providing a strong
1021 protection against assertion redirection because only the intended RP will have the key required to
1022 decrypt the assertion. Most mitigations to federation threats require application of protocol-specific
1023 guidance for SAML and OIDC. These considerations are not specific to the mobile SSO use case;
1024 application of a security-focused profile of these protocols can mitigate many potential issues.

1025 In addition to RFC 8252, application developers and RP service providers should consult the *OAuth 2.0*
1026 *Threat Model and Security Considerations* documented in RFC 6819 [19] for best practices for
1027 implementing OAuth 2.0. The AppAuth library supports a secure OAuth client implementation by
1028 automatically handling details like PKCE. Key protections for OAuth and OIDC include those listed below:

- 1029 ▪ Requiring https for protocol requests and responses protects access tokens and authorization
1030 codes and authenticates the server to the client.
- 1031 ▪ Using the mobile operating system browser or in-application browser tabs for the
1032 authentication flow, in conformance with RFC 8252, protects user credentials from exposure to
1033 the mobile client application or the application service provider.
- 1034 ▪ OAuth tokens are associated with access scopes, which can be used to limit the authorizations
1035 granted to any given client application, which somewhat mitigates the potential for misuse of
1036 compromised access tokens.
- 1037 ▪ PKCE, as explained previously, prevents interception of the authorization code by malicious
1038 applications on the mobile device.

1039 **5.3 Scenarios and Findings**

1040 The overall test scenario on Android devices involved launching the Motorola Solutions PSX Cockpit
1041 mobile application, authenticating, and then subsequently launching additional PSX applications and
1042 validating that the applications could access the back-end APIs and reflected the identity of the
1043 authenticated user. To enable testing of the two different authentication scenarios, two separate “user
1044 organization” infrastructures were created in the NCCoE lab, and both were registered as IdPs to the
1045 test PingFederate instance acting as the PSX AS. A “domain selector” was created in PingFederate to
1046 perform IdP discovery based on the domain of the user’s email address, enabling the user to trigger
1047 authentication at one of the IdPs.

1048 On iOS devices, two demonstration applications—a chat application and a mapping application, with
1049 corresponding back-end APIs—were developed to demonstrate SSO. The iOS demo used the same
1050 authentication infrastructure in the NCCoE lab as the Android demo. The demo consisted of launching

1051 either application and authenticating to the IdP that supported OpenID Connect and FIDO UAF, then
1052 launching the additional demo application to demonstrate SSO and access to the back-end APIs with the
1053 identity of the authenticated user.

1054 Prior to testing the authentication infrastructure, users had to register U2F and UAF authenticators at
1055 the respective IdPs. FIDO authenticator registration requires a process that provides high assurance that
1056 the authenticator is in possession of the claimed account holder. In practice, this typically requires a
1057 strongly authenticated session or an in-person registration process overseen by an administrator. In the
1058 lab, a notional enrollment process was implemented with the understanding that real-world processes
1059 would be different and subject to agency security policies. Organizations should refer to NIST SP 800-
1060 63B [\[10\]](#) for specific considerations regarding credential enrollment. From a FIDO perspective, however,
1061 the registration data used would be the same.

1062 Lab testing showed that the build architecture consistently provided SSO between applications. Two
1063 operational findings were uncovered during testing:

- 1064 ▪ Knowing the location of the NFC radio on the mobile device greatly improves the user
1065 experience when authenticating with an NFC token, such as the YubiKey Neo. The team found
1066 that NFC radios are in different locations on different devices; on the Nexus 6P, for example, the
1067 NFC radio is near the top of the device, near the camera, whereas on the Galaxy S6 Edge, the
1068 NFC radio is slightly below the vertical midpoint of the device. After initial experimentation to
1069 locate the radio, team members could quickly and reliably make a good NFC connection with the
1070 YubiKey by holding it in the correct location. Device manufacturers provide NFC radio location
1071 information via device technical specifications.
- 1072 ▪ Time synchronization between servers is critical. In lab testing, intermittent authentication
1073 errors were found to be caused by clock drift between the IdP and the AS. This manifested as
1074 the AS reporting JavaScript Object Notation Web Token validation errors when attempting to
1075 validate ID tokens received from the IdP. All participants in the federation scheme should
1076 synchronize their clocks to a reliable network time protocol (NTP) source, such as the NIST NTP
1077 pools [\[20\]](#). Implementations should allow for a small amount of clock skew—on the order of a
1078 few seconds—to account for the unpredictable latency of network traffic.

1079 6 Future Build Considerations

1080 6.1 Single Logout

1081 To ensure that only authorized personnel get access to application resources, users must be logged out
1082 from application sessions when access is no longer needed or when a session expires. In an SSO
1083 scenario, a user may need to be logged out from one or many applications at a given time. This scenario
1084 will demonstrate architectures for tearing down user sessions, clearly communicating to the user which
1085 application(s) has (have) active sessions, and ensuring that active sessions are not orphaned.

1086 **6.2 Shared Devices**

1087 This scenario will focus on a situation where two or more colleagues share a single mobile device to
1088 accomplish a mission. The credentials, such as the FIDO UAF and U2F used in this guide, will be included
1089 but may need to be registered to multiple devices. This scenario will explore situations in which multiple
1090 profiles or no profiles are installed on a device, potentially requiring the user to log out prior to giving
1091 the device to another user.

1092 **6.3 Step-Up Authentication**

1093 A user will access applications by using an acceptable but low assurance authenticator. Upon requesting
1094 access to an application that requires higher assurance, the user will be prompted for an additional
1095 authentication factor. Determinations on whether to step up may be based on risk-relevant data points
1096 collected by the IdP at the time of authentication, referred to as the authentication context.

1097 **Appendix A Mapping to Cybersecurity Framework Core**

1098 Table A-1 maps informative National Institute of Standards and Technology (NIST) and consensus
 1099 security references to the Cybersecurity Framework core Subcategories that are addressed by NIST
 1100 Special Publication (SP) 1800-13. The references do not include protocol specifications that are
 1101 implemented by the individual products that compose the demonstrated security platforms. While
 1102 some of the references provide general guidance that informs implementation of referenced
 1103 Cybersecurity Framework core functions, the NIST SP 1800-13 references provide specific
 1104 recommendations that should be considered when composing and configuring security platforms and
 1105 technologies described in this practice guide.

1106 **Table A-1 Cybersecurity Framework Categories**

Category	Subcategory	Informative References
Asset Management (ID.AM): The data, personnel, devices, systems, and facilities that enable the organization to achieve business purposes are identified and managed consistent with their relative importance to business objectives and the organization's risk strategy.	ID.AM-1: Physical devices and systems within the organization are inventoried.	CCS CSC 1 COBIT 5 BAI09.01, BAI09.02 ISA 62443-2-1:2009 4.2.3.4 ISA 62443-3-3:2013 SR 7.8 ISO/IEC 27001:2013 A.8.1.1, A.8.1.2 NIST SP 800-53 Rev. 4 CM-8
Access Control (PR.AC): Access to assets and associated facilities is limited to authorized users, processes, or devices, and to authorized activities and transactions.	PR.AC-1: Identities and credentials are managed for authorized devices and users.	CCS CSC 16 COBIT 5 DSS05.04, DSS06.03 ISA 62443-2-1:2009 4.3.3.5.1 ISA 62443-3-3:2013 SR 1.1, SR 1.2, SR 1.3, SR 1.4, SR 1.5, SR 1.7, SR 1.8, SR 1.9 ISO/IEC 27001:2013 A.9.2.1, A.9.2.2, A.9.2.4, A.9.3.1, A.9.4.2, A.9.4.3 NIST SP 800-53 Rev. 4 AC-2, Information Assurance Family

Category	Subcategory	Informative References
	PR.AC-3: Remote access is managed.	COBIT 5 APO13.01, DSS01.04, DSS05.03 ISA 62443-2-1:2009 4.3.3.6.6 ISA 62443-3-3:2013 SR 1.13, SR 2.6 ISO/IEC 27001:2013 A.6.2.2, A.13.1.1, A.13.2.1 NIST SP 800-53 Rev. 4 AC-17, AC-19, AC-20
	PR.AC-4: Access permissions are managed, incorporating the principles of least privilege and separation of duties.	CCS CSC 12, 15 ISA 62443-2-1:2009 4.3.3.7.3 ISA 62443-3-3:2013 SR 2.1 ISO/IEC 27001:2013 A.6.1.2, A.9.1.2, A.9.2.3, A.9.4.1, A.9.4.4 NIST SP 800-53 Rev. 4 AC-2, AC-3, AC-5, AC-6, AC-16
Data Security (PR.DS): Information and records (data) are managed consistent with the organization's risk strategy to protect the confidentiality, integrity, and availability of information.	PR.DS-5: Protections against data leaks are implemented.	CCS CSC 17 COBIT 5 APO01.06 ISA 62443-3-3:2013 SR 5.2 ISO/IEC 27001:2013 A.6.1.2, A.7.1.1, A.7.1.2, A.7.3.1, A.8.2.2, A.8.2.3, A.9.1.1, A.9.1.2, A.9.2.3, A.9.4.1, A.9.4.4, A.9.4.5, A.13.1.3, A.13.2.1, A.13.2.3, A.13.2.4, A.14.1.2, A.14.1.3 NIST SP 800-53 Rev. 4 AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4

Category	Subcategory	Informative References
<p>Protective Technology (PR.PT): Technical security solutions are managed to ensure the security and resilience of systems and assets, consistent with related policies, procedures, and agreements.</p>	<p>PR.PT-1: Audit/log records are determined, documented, implemented, and reviewed in accordance with policy.</p>	<p>CCS CSC 14 COBIT 5 APO11.04 ISA 62443-2-1:2009 4.3.3.3.9, 4.3.3.5.8, 4.3.4.4.7, 4.4.2.1, 4.4.2.2, 4.4.2.4 ISA 62443-3-3:2013 SR 2.8, SR 2.9, SR 2.10, SR 2.11, SR 2.12 ISO/IEC 27001:2013 A.12.4.1, A.12.4.2, A.12.4.3, A.12.4.4, A.12.7.1 NIST SP 800-53 Rev. 4 Audit and Accountability Family</p>
	<p>PR.PT-2: Removable media is protected and its use restricted according to policy.</p>	<p>COBIT 5 DSS05.02, APO13.01 ISA 62443-3-3:2013 SR 2.3 ISO/IEC 27001:2013 A.8.2.2, A.8.2.3, A.8.3.1, A.8.3.3, A.11.2.9 NIST SP 800-53 Rev. 4 MP-2, MP-4, MP-5, MP-7</p>
	<p>PR.PT-3: Access to systems and assets is controlled, incorporating the principle of least functionality.</p>	<p>COBIT 5 DSS05.02 ISA 62443-2-1:2009 4.3.3.5.1, 4.3.3.5.2, 4.3.3.5.3, 4.3.3.5.4, 4.3.3.5.5, 4.3.3.5.6, 4.3.3.5.7, 4.3.3.5.8, 4.3.3.6.1, 4.3.3.6.2, 4.3.3.6.3, 4.3.3.6.4, 4.3.3.6.5, 4.3.3.6.6, 4.3.3.6.7, 4.3.3.6.8, 4.3.3.6.9, 4.3.3.7.1, 4.3.3.7.2, 4.3.3.7.3, 4.3.3.7.4 ISA 62443-3-3:2013 SR 1.1, SR 1.2, SR 1.3, SR 1.4, SR 1.5, SR 1.6, SR 1.7, SR 1.8, SR 1.9, SR 1.10, SR 1.11, SR 1.12, SR 1.13, SR 2.1, SR 2.2, SR 2.3, SR 2.4, SR 2.5, SR 2.6, SR 2.7 ISO/IEC 27001:2013 A.9.1.2 NIST SP 800-53 Rev. 4 AC-3, CM-7</p>

Category	Subcategory	Informative References
	PR.PT-4: Communications and control networks are protected.	CCS CSC 7 COBIT 5 DSS05.02, APO13.01 ISA 62443-3-3:2013 SR 3.1, SR 3.5, SR 3.8, SR 4.1, SR 4.3, SR 5.1, SR 5.2, SR 5.3, SR 7.1, SR 7.6 ISO/IEC 27001:2013 A.13.1.1, A.13.2.1 NIST SP 800-53 Rev. 4 AC-4, AC-17, AC-18, CP-8, SC-7

Appendix B Assumptions Underlying the Build

1107 This project is guided by the following assumptions. Implementers are advised to consider whether the
1108 same assumptions can be made based on current policy, process, and information technology (IT)
1109 infrastructure. Where applicable, appropriate guidance is provided to assist this process as described in
1110 the following subsections.

1111 **B.1 Identity Proofing**

1112 National Institute of Standards and Technology (NIST) Special Publication (SP) 800-63A, *Enrollment and*
1113 *Identity Proofing* [21], addresses how applicants can prove their identities and become enrolled as valid
1114 subjects within an identity system. It provides requirements for processes by which applicants can both
1115 proof and enroll at one of three different levels of risk mitigation, in both remote and physically present
1116 scenarios. NIST SP 800-63A contains both normative and informative material. An organization should
1117 use NIST SP 800-63A to develop and implement an identity proofing plan within its enterprise.

1118 **B.2 Mobile Device Security**

1119 Mobile devices can add to an organization's productivity by providing employees with access to business
1120 resources at any time. Not only has this reshaped how traditional tasks are accomplished but
1121 organizations are also devising entirely new ways to work. However, mobile devices may be lost or
1122 stolen. A compromised mobile device may allow remote access to sensitive on-premises organizational
1123 data or any other data that the user has entrusted to the device. Several methods exist to address these
1124 concerns (e.g., using a device lock screen, setting shorter screen time-outs, forcing a device wipe in case
1125 of too many failed authentication attempts). It is up to the organization to implement these types of
1126 security controls, which can be enforced with Enterprise Mobility Management (EMM) software (see
1127 [Section B.4](#)).

1128 NIST SP 1800-4, *Mobile Device Security: Cloud and Hybrid Builds* [22], demonstrates how to secure
1129 sensitive enterprise data that is accessed by and/or stored on employees' mobile devices. The NIST
1130 *Mobile Threat Catalogue* [23] identifies threats to mobile devices and associated mobile infrastructure
1131 to support development and implementation of mobile security capabilities, best practices, and security
1132 solutions to better protect enterprise IT. We strongly encourage organizations implementing this
1133 practice guide in whole or in part to consult these resources when developing and implementing a
1134 mobile device security plan for their organizations.

1135 **B.3 Mobile Application Security**

1136 The security qualities of an entire platform can be compromised if an application exhibits vulnerable or
1137 malicious behavior. Application security is paramount in ensuring that the security controls
1138 implemented in other architecture components can effectively mitigate threats. The practice of making

1139 sure that an application is secure is known as software assurance (SwA). This is defined as “the level of
1140 confidence that software is free from vulnerabilities, either intentionally designed into the software or
1141 accidentally inserted at any time during its lifecycle, and that the software functions in the intended
1142 manner” [\[24\]](#).

1143 In an architecture that largely relies on third-party—usually closed-source—applications to handle daily
1144 user functions, good SwA hygiene can be difficult to implement. To address this problem, NIST has
1145 released guidance on how to structure and implement an application-vetting process (also known as
1146 “app vetting”) [\[25\]](#). This takes an organization through the following steps:

- 1147 1. understanding the process for vetting the security of mobile applications
- 1148 2. planning for implementation of an app-vetting process
- 1149 3. developing application security requirements
- 1150 4. understanding types of application vulnerabilities and testing methods used to detect those
1151 vulnerabilities
- 1152 5. determining whether an application is acceptable for deployment on the organization’s mobile
1153 devices

1154 Public safety organizations (PSOs) should carefully consider their application-vetting needs. Though
1155 major mobile-application stores, such as Apple’s iTunes Store and Google’s Play Store, have vetting
1156 mechanisms to find vulnerable and malicious applications, organizations may have needs beyond these
1157 proprietary tools. Per NIST SP 800-163, *Vetting the Security of Mobile Applications* [\[25\]](#):

1158 App stores may perform app vetting processes to verify compliance with their own
1159 requirements. However, because each app store has its own unique, and not always
1160 transparent, requirements and vetting processes, it is necessary to consult current agreements
1161 and documentation for a particular app store to assess its practices. Organizations should not
1162 assume that an app has been fully vetted and conforms to their security requirements simply
1163 because it is available through an official app store. Third party assessments that carry a
1164 moniker of “approved by” or “certified by” without providing details of which tests are
1165 performed, what the findings were, or how apps are scored or rated, do not provide a reliable
1166 indication of software assurance. These assessments are also unlikely to take organization
1167 specific requirements and recommendations into account, such as federal-specific cryptography
1168 requirements.

1169 The First Responder Network Authority (FirstNet) provides an application store specifically geared
1170 toward first responder applications. Through the FirstNet Developer Portal [\[26\]](#), application developers
1171 can submit mobile applications for evaluation against its published development guidelines. The
1172 guidelines include security, scalability, and availability. Compliant applications can be selected for

1173 inclusion in the FirstNet App Store. This provides first responder agencies with a repository of
1174 applications that have been tested to a known set of standards.

1175 PSOs should avoid the unauthorized “side loading” of mobile applications that are not subject to
1176 organizational vetting requirements.

1177 **B.4 Enterprise Mobility Management**

1178 The rapid evolution of mobile devices has introduced new paradigms for work environments, along with
1179 new challenges for enterprise IT to address. EMM solutions, as part of an EMM program, provide a
1180 variety of ways to view, organize, secure, and maintain a fleet of mobile devices. EMM solutions can
1181 vary greatly in form and function, but in general, they use platform-provided application programming
1182 interfaces. Sections 3 and 4 of NIST SP 800-124 [27] describe the two basic approaches of EMM, along
1183 with components, capabilities, and their uses. One approach, commonly known as “fully managed,”
1184 controls the entire device. Another approach, usually used for bring-your-own-device situations, wraps
1185 or “containerizes” applications inside a secure sandbox so that they can be managed without affecting
1186 the rest of the device.

1187 EMM capabilities can be grouped into four general categories:

- 1188 1. General policy—centralized technology to enforce security policies of particular interest for
1189 mobile device security, such as accessing hardware sensors like global positioning system (GPS),
1190 accessing native operating-system (OS) services like a web browser or email client, managing
1191 wireless networks, monitoring when policy violations occur, and limiting access to enterprise
1192 services if the device is vulnerable or compromised
- 1193 2. Data communication and storage—automatically encrypting data in transit between the device
1194 and the organization (e.g., through a virtual private network); strongly encrypting data at rest on
1195 internal and removable media storage; and wiping the device if it is being reissued to another
1196 user, has been lost, or has surpassed a certain number of incorrect unlock attempts
- 1197 3. User and device authentication—requiring a device password/passcode and parameters for
1198 password strength, remotely restoring access to a locked device, automatically locking the
1199 device after an idle period, and remotely locking the device if needed
- 1200 4. Applications—restricting which application stores may be used, restricting which applications can
1201 be installed, requiring specific application permissions (such as using the camera or GPS),
1202 restricting use of OS synchronization services, verifying digital signatures to ensure that
1203 applications are unmodified and sourced from trusted entities, and automatically
1204 installing/updating/removing applications according to administrative policies

1205 Public safety and first responder (PSFR) organizations will have different requirements for EMM; this
1206 document does not prescribe any specific processes or procedures but assumes that they have been

1207 established in accordance with agency requirements. However, sections of this document refer to the
1208 NIST Mobile Threat Catalogue [\[23\]](#), which does list the use of EMM solutions as mitigations for certain
1209 types of threats.

1210 **B.5 FIDO Enrollment Process**

1211 Fast Identity Online (FIDO) provides a framework for users to register a variety of different multifactor
1212 authenticators and use them to authenticate to applications and identity providers. Before an
1213 authenticator can be used in an online transaction, it must be associated with the user's identity. This
1214 process is described in NIST SP 800-63B [\[10\]](#) as *authenticator binding*. NIST SP 800-63B specifies
1215 requirements for binding authenticators to a user's account both during initial enrollment and after
1216 enrollment, and recommends that relying parties support binding multiple authenticators to each user's
1217 account to enable alternative strong authenticators in case the primary authenticator is lost, stolen, or
1218 damaged.

1219 Authenticator binding may be an in-person or remote process, but in both cases, the user's identity and
1220 control over the authenticator being bound to the account must be established. This is related to
1221 identity proofing, discussed in [Section B.1](#), but requires that credentials be issued in a manner that
1222 maintains a tight binding with the user identity that has been established through proofing. PSFR
1223 organizations will have different requirements for identity and credential management; this document
1224 does not prescribe any specific processes or procedures but assumes that they have been established in
1225 accordance with agency requirements.

1226 As an example, in-person authenticator binding could be implemented by having administrators
1227 authenticate with their own credentials and authorize the association of an authenticator with an
1228 enrolling user's account. Once a user has one enrolled authenticator, it can be used for online
1229 enrollment of other authenticators at the same assurance level or lower. Allowing users to enroll strong
1230 multifactor authenticators based on authentication with weaker credentials, such as username and
1231 password or knowledge-based questions, can undermine the security of the overall authentication
1232 scheme and should be avoided.

Appendix C Architectural Considerations for the Mobile Application Single Sign-On Build

This appendix details architectural considerations relating to single sign-on (SSO) with OAuth 2.0; Internet Engineering Task Force (IETF) Request for Comments (RFC) 8252; and AppAuth open-source libraries, federation, and types of multifactor authentication (MFA).

C.1 SSO with OAuth 2.0, IETF RFC 8252, and AppAuth Open-Source Libraries

As stated above, SSO streamlines the user experience by enabling a user to authenticate once and to subsequently access different applications without having to authenticate again. SSO on mobile devices is complicated by the sandboxed architecture, which makes it difficult to share the session state with back-end systems between individual applications. Enterprise Mobility Management (EMM) vendors have provided solutions through proprietary software development kits (SDKs), but this approach requires integrating the SDK with each individual application and does not scale to a large and diverse population, such as the public safety and first responder (PSFR) user community.

OAuth 2.0, when implemented in accordance with RFC 8252 (the *OAuth 2.0 for Native Apps Best Current Practice*), provides a standards-based SSO pattern for mobile applications. The OpenID Foundation's AppAuth libraries [14] can facilitate building mobile applications in full compliance with IETF RFC 8252, but any mobile application that follows RFC 8252's core recommendation of using a shared external user-agent for the OAuth authorization flow will have the benefit of SSO.

To implement SSO with OAuth 2.0, this practice guide recommends that application developers choose one of the following options:

- Implement IETF RFC 8252 themselves. This RFC specifies that OAuth 2.0 authorization requests from native applications should be made only through external user-agents, primarily the user's browser. This specification details the security and usability reasons for why this is the case and how native applications and authorization servers can implement this best practice. RFC 8252 also recommends the use of Proof Key for Code Exchange (PKCE), as detailed in RFC 7636 [28], which protects against authorization code interception attacks.
- Integrate the AppAuth open-source libraries (that implement RFC 8252 and RFC 7636) for mobile SSO. The AppAuth libraries make it easy for application developers to enable standards-based authentication, SSO, and authorization to application programming interfaces. This was the option chosen by the implementers of this build.

When OAuth is implemented in a native application, it operates as a *public client*; this presents security concerns with aspects like client secrets and redirected uniform resource identifiers (URIs). The AppAuth pattern mitigates these concerns and provides several security advantages for developers. The primary

1267 benefit of RFC 8252 is that native applications use an external user-agent (e.g., the Chrome for Android
1268 web browser) instead of an embedded user-agent (e.g., an Android WebView) for their OAuth
1269 authorization requests.

1270 An embedded user-agent is demonstrably less secure and user-friendly than an external user-agent.
1271 Embedded user-agents potentially allow the client to log keystrokes, capture user credentials, copy
1272 session cookies, and automatically submit forms to bypass user consent. In addition, session information
1273 for embedded user-agents is stored on a per-application basis. This does not allow for SSO functionality,
1274 which users generally prefer and which this practice guide sets out to implement. Recent versions of
1275 Android and iPhone operating system (iOS) both provide implementations of “in-application browser
1276 tabs” that retain the security benefits of using an external user-agent while avoiding visible context-
1277 switching between the application and the browser; RFC 8252 recommends their use where available.
1278 In-application browser tabs are supported in Android 4.1 and higher and in iOS 9 and higher.

1279 AppAuth also requires that public client applications eschew client secrets in favor of PKCE, which is a
1280 standard extension to the OAuth 2.0 framework. When using the AppAuth pattern, the following steps
1281 are performed:

- 1282 1. The user opens the client application and initiates a sign-in.
- 1283 2. The client uses a browser to initiate an authorization request to the authentication server (AS).
- 1284 3. The user authenticates to the identity provider (IdP).
- 1285 4. The OpenID Connect (OIDC)/security assertion markup language (SAML) flow takes place, and
1286 the user authenticates to the AS.
- 1287 5. The browser requests an authorization code from the AS.
- 1288 6. The browser returns the authorization code to the client.
- 1289 7. The client uses its authorization code to request and obtain an access token.

1290 There is a possible attack vector at the end user’s device in this workflow if PKCE is not enabled. During
1291 step 6, so that the client application can receive the authorization code, the AS redirects the browser to
1292 a URI on which the client application is listening. However, a malicious application could register for this
1293 URI and attempt to intercept the code so that it may obtain an access token. PKCE-enabled clients use a
1294 dynamically generated random *code verifier* to ensure proof of possession for the authorization code. If
1295 the grant is intercepted by a malicious application before being returned to the client, the malicious
1296 application will be unable to use the grant without the client’s secret verifier.

1297 AppAuth also outlines several other actions to consider, such as three types of redirect URIs, native-
1298 application client registration guidance, and reverse domain-name-based schemes. These are supported
1299 and/or enforced with secure defaults in the AppAuth libraries. The libraries are open-source and include

1300 sample code for implementation. In addition, if Universal Second Factor (U2F) or Universal
1301 Authentication Framework (UAF) is desired, that flow is handled entirely by the external user-agent, so
1302 client applications do not need to implement any of that functionality.

1303 The AppAuth library takes care of several boilerplate tasks for developers, such as caching access tokens
1304 and refresh tokens, checking access-token expiration, and automatically refreshing access tokens. To
1305 implement the AppAuth pattern in an Android application by using the provided library, a developer
1306 needs to perform the following actions:

- 1307 ▪ Add the Android AppAuth library as a Gradle dependency.
- 1308 ▪ Add a redirect URI to the Android manifest.
- 1309 ▪ Add the Java code to initiate the AppAuth flow and to use the access token afterward.
- 1310 ▪ Register the application's redirect URI with the AS.

1311 Using the AppAuth library in an iOS application is a similar process:

- 1312 ▪ Add the AppAuth library by using either Pods or Carthage.
- 1313 ▪ Configure a custom uniform resource locator (URL) scheme in the info.plist file.
- 1314 ▪ Update the view controllers and application delegate to initiate the AppAuth flow and to use the
1315 access token afterward.
- 1316 ▪ Register the application's redirect URI with the AS.

1317 To implement the AppAuth pattern *without* using a library, the user will need to follow the general
1318 guidance laid out in RFC 8252, review and follow the operating system-specific guidance in the AppAuth
1319 documentation [\[14\]](#), and adhere to the requirements of both the OAuth 2.0 framework documented in
1320 RFC 6749 [\[29\]](#) and the PKCE.

1321 [C.1.1 Attributes and Authorization](#)

1322 Authorization, in the sense of applying a policy to determine the rights and privileges that apply to
1323 application requests, is beyond the scope of this practice guide. OAuth 2.0 provides delegation of user
1324 authorizations to mobile applications acting on their behalf, but this is distinct from the authorization
1325 policy enforced by the application. This guide is agnostic to the specific authorization model (e.g., role-
1326 based access control [RBAC], attribute-based access control [ABAC], capability lists) that applications will
1327 use, and the SSO mechanism documented here is compatible with virtually any back-end authorization
1328 policy.

1329 While applications could potentially manage user roles and privileges internally, federated
1330 authentication provides the capability for the IdP to provide user attributes to relying parties (RPs).
1331 These attributes might be used to map users to defined application roles or used directly in an ABAC

1332 policy (e.g., to restrict access to sworn law enforcement officers). Apart from authorization, attributes
1333 may provide identifying information useful for audit functions, contact information, or other user data.

1334 In the build architecture, the AS is an RP to the user's IdP, which is either a SAML IdP or an OIDC
1335 provider. SAML IdPs can return attribute elements in the SAML response. OIDC providers can return
1336 attributes as claims in the identification (ID) token, or the AS can request them from the user
1337 information end point. In both cases, the AS can validate the IdP's signature of the asserted attributes to
1338 ensure their validity and integrity. Assertions can also optionally be encrypted, which both protects their
1339 confidentiality in transit and enforces audience restrictions because only the intended RP will be able to
1340 decrypt them.

1341 Once the AS has received and validated the asserted user attributes, it could use them as issuance
1342 criteria to determine whether an access token should be issued for the client to access the requested
1343 scopes. In the OAuth 2.0 framework, *scopes* are individual access entitlements that can be granted to a
1344 client application. In addition, the attributes could be provided to the protected resource server to
1345 enable the application to enforce its own authorization policies. Communications between the AS and
1346 protected resource are internal design concerns for the software as a service (SaaS) provider. One
1347 method of providing attributes to the protected resource is for the AS to issue the access token as a
1348 JavaScript Object Notation (JSON) Web Token (JWT) containing the user's attributes. The protected
1349 resource could also obtain attributes by querying the AS's token introspection end point, where they
1350 could be provided as part of the token metadata in the introspection response.

1351 **C.2 Federation**

1352 The preceding section discussed the communication of attributes from the IdP to the AS for use in
1353 authorization decisions. In the build architecture, it is assumed that the SaaS provider may be an RP of
1354 many IdPs supporting different user organizations. Several first responder organizations have their own
1355 IdPs, each managing its own users' attributes. This presents a challenge if the RP needs to use those
1356 attributes for authorization. Local variations in attribute names, values, and encodings would make it
1357 difficult to apply a uniform authorization policy across the user base. If the SaaS platform enables
1358 sharing of sensitive data between organizations, participants would need some assurance that their
1359 partners were establishing and managing user accounts and attributes appropriately—promptly
1360 removing access for terminated employees and performing appropriate validation before assigning
1361 attributes that enable privileged access. Federations attempt to address this issue by creating common
1362 profiles and policies governing use and management of attributes and authentication mechanisms,
1363 which members are expected to follow. This facilitates interoperability, and members are also typically
1364 audited for compliance with the federation's policies and practices, enabling mutual trust in attributes
1365 and authentication.

1366 As an example, the National Identity Exchange Federation (NIEF) is a federation serving law enforcement
1367 organizations and networks, including the Federal Bureau of Investigation, the Department of Homeland

1368 Security, the Regional Information Sharing System, and the Texas Department of Public Safety. NIEF has
 1369 established SAML profiles for both web-browser and system-to-system use cases, and a registry of
 1370 common attributes for users, resources, and other entities. NIEF attributes are grouped into attribute
 1371 bundles, with some designated as mandatory, meaning that all participating IdPs must provide those
 1372 attributes, and participating RPs can depend on their presence in the SAML response.

1373 The architecture documented in this build guide is fully compatible with NIEF and other federations,
 1374 though this would require configuring IdPs and RPs in compliance with the federation's policies. The use
 1375 of SAML IdPs is fully supported by this architecture, as is the coexistence of SAML IdPs and OIDC
 1376 providers.

1377 NIST SP 800-63-3 [\[17\]](#) defines Federation Assurance Levels (FALs) and their implementation
 1378 requirements. FALs are a measure of the assurance that assertions presented to an RP are genuine and
 1379 unaltered, pertain to the individual presenting them, are not subject to replay at other RPs, and are
 1380 protected from many additional potential attacks on federated authentication schemes. A high-level
 1381 summary of the requirements for FALs 1–3 is provided in Table C-1.

1382 **Table C-1 FAL Requirements**

FAL	Requirement
1	Bearer assertion, signed by IdP
2	Bearer assertion, signed by IdP, and encrypted to RP
3	Holder of key assertion, signed by IdP, and encrypted to RP

1383 IdPs typically sign assertions, and this functionality is broadly supported in available software. For SAML,
 1384 the IdP's public key is provided in the SAML metadata. For OIDC, the public key can be provided through
 1385 the discovery end point, if supported; otherwise, the key would be provided to the RP out of band.
 1386 Encrypting assertions is also relatively trivial and requires providing the RP's public key to the IdP. The
 1387 build architecture in this guide can support FAL-1 and FAL-2 with relative ease.

1388 The requirement for holder of key assertions makes FAL-3 more difficult to implement. A SAML holder
 1389 of key profile exists but has never been widely implemented in a web-browser SSO context. The OIDC
 1390 core specification does not include a mechanism for a holder of key assertions; however, the
 1391 forthcoming token binding over the hypertext transfer protocol (http) specification [\[30\]](#) and related
 1392 RFCs may provide a pathway to supporting FAL-3 in an OIDC implementation.

1393 **C.3 Authenticator Types**

1394 When considering MFA implementations, PSFR organizations should carefully consider organizationally
 1395 defined authenticator requirements. These requirements may include:

- 1396 ■ the sensitivity of data being accessed and the commensurate level of authentication assurance
1397 needed
- 1398 ■ environmental constraints, such as gloves or masks, that may limit the usability and
1399 effectiveness of certain authentication modalities
- 1400 ■ costs throughout the authenticator life cycle, such as authenticator binding, loss, theft,
1401 unauthorized duplication, expiration, and revocation
- 1402 ■ policy and compliance requirements, such as the Health Insurance Portability and Accountability
1403 Act (HIPAA) [\[31\]](#), the Criminal Justice Information System Security Policy [\[32\]](#), or other
1404 organizationally defined requirements
- 1405 ■ support of current information technology infrastructure, including mobile devices, for various
1406 authenticator types

1407 The new, third revision of NIST SP 800-63, *Digital Identity Guidelines* [\[17\]](#), is a suite of documents that
1408 provide technical requirements and guidance for federal agencies implementing digital identity services,
1409 and it may assist PSFR organizations when selecting authenticators. The most significant difference from
1410 previous versions of NIST SP 800-63 is the retirement of the previous assurance rating system, known as
1411 the Levels of Assurance (LOA), established by Office of Management and Budget Memorandum M-04-
1412 04, *E-Authentication Guidance for Federal Agencies*. In the new NIST SP 800-63-3 guidance, digital
1413 identity assurance is split into three ordinals as opposed to the single ordinal in LOA. The three ordinals
1414 are listed below:

- 1415 ■ identity assurance level (IAL)
1416 ■ authenticator assurance level (AAL)
1417 ■ FAL

1418 This practice guide is primarily concerned with AALs and how they apply to the reference architecture
1419 outlined in Table 3-2.

1420 The strength of an authentication transaction is measured by the AAL. A higher AAL means stronger
1421 authentication and requires more resources and capabilities by attackers to subvert the authentication
1422 process. We discuss a variety of multifactor implementations in this practice guide. NIST SP 800-63-3
1423 gives us a reference to map the risk reduction of the various implementations recommended in this
1424 practice guide.

1425 The AAL is determined by authenticator type and combination, verifier requirements, reauthentication
1426 policies, and security control baselines, as defined in NIST SP 800-53, *Security and Privacy Controls for*
1427 *Federal Information Systems and Organizations* [\[33\]](#). A summary of requirements at each of the levels is
1428 provided in Table C-2.

1429 A memorized secret (most commonly implemented as a password) satisfies AAL1, but this alone is not
1430 enough to reach the higher levels shown in Table C-2. For AAL2 and AAL3, some form of MFA is

1431 required. MFA comes in many forms. The architecture in this practice guide describes two examples.
 1432 One example is a multifactor software cryptographic authenticator, where a biometric authenticator
 1433 application is installed on the mobile device—the two factors being possession of the private key and
 1434 the biometric. The other example is a combination of a memorized secret and a single-factor
 1435 cryptographic device, which performs cryptographic operations via a direct connection to the user end
 1436 point.

1437 Reauthentication requirements also become more stringent for higher levels. AAL1 requires
 1438 reauthentication only every 30 days, but AAL2 and AAL3 require reauthentication every 12 hours. At
 1439 AAL2, users may reauthenticate by using a single authentication factor, but at AAL3, users must
 1440 reauthenticate by using both of their authentication factors. At AAL2, 30 minutes of idle time is allowed,
 1441 but only 15 minutes is allowed at AAL3.

1442 For a full description of the different types of multifactor authenticators and AAL requirements, please
 1443 refer to NIST SP 800-63B [\[10\]](#).

1444 **Table C-2 AAL Summary of Requirements**

Requirement	AAL1	AAL2	AAL3
Permitted authenticator types	Memorized Secret; Lookup Secret; Out of Band; Single Factor (SF) Onetime Password (OTP) Device; Multifactor (MF) OTP Device; SF Crypto Software; SF Crypto Device; MF Crypto Software; MF Crypto Device	MF OTP Device; MF Crypto Software; MF Crypto Device; or Memorized Secret plus: <ul style="list-style-type: none">▪ Lookup Secret▪ Out of Band▪ SF OTP Device▪ SF Crypto Software▪ SF Crypto Device	MF Crypto Device; SF Crypto Device plus Memorized Secret; SF OTP Device plus MF Crypto Device or Software; SF OTP Device plus SF Crypto Software plus Memorized Secret
Federal Information Processing Standard (FIPS) 140-2 verification	Level 1 (government agency verifiers)	Level 1 (government agency authenticators and verifiers)	Level 2 overall (MF authenticators) Level 1 overall (verifiers and SF Crypto Devices) Level 3 physical security (all authenticators)

Requirement	AAL1	AAL2	AAL3
Reauthentication	30 days	12 hours, or after 30 minutes of inactivity; MAY use one authentication factor	12 hours, or after 15 minutes of inactivity; SHALL use both authentication factors
Security controls	NIST SP 800-53 Low Baseline (or equivalent)	NIST SP 800-53 Moderate Baseline (or equivalent)	NIST SP 800-53 High Baseline (or equivalent)
Man-in-the-middle resistance	Required	Required	Required
Verifier-impersonation resistance	Not required	Not required	Required
Verifier-compromise resistance	Not required	Not required	Required
Replay resistance	Not required	Required	Required
Authentication intent	Not required	Recommended	Required
Records retention policy	Required	Required	Required
Privacy controls	Required	Required	Required

1445 The Fast Identity Online (FIDO) Alliance has published specifications for two types of authenticators
 1446 based on UAF and U2F. These protocols operate agnostic of the FIDO authenticator, allowing PSOs to
 1447 choose any FIDO-certified authenticator that meets operational requirements and to implement it with
 1448 this solution. As new FIDO-certified authenticators become available in the marketplace, PSOs may
 1449 choose to migrate to these new authenticators if they better meet PSFR needs in their variety of duties.

1450 C.3.1 UAF Protocol

1451 The UAF protocol [2] allows users to register their device to the online service by selecting a local
 1452 authentication mechanism, such as swiping a finger, looking at the camera, speaking into the
 1453 microphone, or entering a personal identification number (PIN). The UAF protocol allows the service to
 1454 select which mechanisms are presented to the user. Once registered, the user simply repeats the local
 1455 authentication action whenever they need to authenticate to the service. The user no longer needs to
 1456 enter their password when authenticating from that device. UAF also allows experiences that combine
 1457 multiple authentication mechanisms, such as fingerprint plus PIN. Data used for local user verification,

1458 such as biometric templates, passwords, or PINs, is validated locally on the device and is not transmitted
1459 to the server. Authentication to the server is performed with a cryptographic key pair, which is unlocked
1460 after local user verification.

1461 C.3.2 U2F Protocol

1462 The U2F protocol [3] allows online services to augment the security of their existing password
1463 infrastructure by adding a strong second factor to user login, typically an external hardware-backed
1464 cryptographic device. The user logs in with a username and password as before and is then prompted to
1465 present the external second factor. The service can prompt the user to present a second-factor device at
1466 any time that it chooses. The strong second factor allows the service to simplify its passwords (e.g., four-
1467 digit PIN) without compromising security. During registration and authentication, the user presents the
1468 second factor by simply pressing a button on a universal serial bus device or tapping over near field
1469 communication.

1470 The user can use their FIDO U2F device across all online services that support the protocol. On desktop
1471 operating systems, the Google Chrome and Opera browsers currently support U2F. U2F is also
1472 supported on Android through the Google Authenticator application, which must be installed from the
1473 Play Store.

1474 C.3.3 FIDO 2

1475 The FIDO 2 project comprises a set of related standardization efforts undertaken by the FIDO Alliance
1476 and the World Wide Web Consortium (W3C). The second iteration of the FIDO standards will support
1477 the W3C's Web Authentication standard [16]. As a W3C recommendation, Web Authentication is
1478 expected to be widely adopted by web browser developers and to provide out-of-the-box FIDO support
1479 without the need to install additional client applications or extensions.

1480 In addition, the proposed FIDO Client-to-Authenticator Protocol (CTAP) standard will support new
1481 authenticator functions, including the ability to set a PIN on authenticators such as YubiKeys. By
1482 requiring a PIN at authentication time, a CTAP-compliant authenticator can provide MFA in a manner
1483 similar to a smart card. This would eliminate the need to pair an external authenticator with an existing
1484 knowledge factor such as username/password authentication against an LDAP database, as was used in
1485 the U2F implementation of this build.

1486 C.3.4 FIDO Key Registration

1487 From the perspective of an IdP, enabling users to authenticate themselves with FIDO-based credentials
1488 requires that users register a cryptographic key with the IdP and associate the registered key with the
1489 username or distinguished name known to the IdP. FIDO registration must be repeated for each
1490 authenticator that the user chooses to associate with their account. FIDO protocols are different from
1491 most authentication protocols in that they permit registering multiple cryptographic keys (from different

1492 authenticators) to use with a single account. This is convenient for end users as it provides a natural
1493 backup solution to lost, misplaced, or forgotten authenticators—users may use any one of their
1494 registered authenticators to access their applications.

1495 The process of a first-time FIDO key registration is fairly simple:

1496 1. A user creates an account for themselves at an application site, or one is created for them as
1497 part of a business process.

1498 2. The user registers a FIDO key with the application through one of the following processes:

1499 a. as part of the account self-creation process

1500 b. upon receiving an email with an invitation to register

1501 c. as part of a registration process, after an authentication process within an organization
1502 application

1503 d. A FIDO authenticator with a temporary, preregistered key is provided so that the user
1504 can strongly authenticate to register a new key with the application, at which point the
1505 temporary key is deleted permanently. Authenticators with preregistered keys may be
1506 combined with shared secrets given/sent to the user out of band to verify their identity
1507 before enabling them to register a new FIDO key with the organization's application.

1508 e. as part of a custom process local to the IdP

1509 Policy at the organization dictates what might be considered most appropriate for a registration process.

1510 C.3.5 FIDO Authenticator Attestation

1511 To meet AAL requirements, RPs may need to restrict the types of FIDO authenticators that can be
1512 registered and used to authenticate. They may also require assurances that the authenticators in use are
1513 not counterfeit or vulnerable to known attacks. The FIDO specifications include mechanisms that enable
1514 the RP to validate the identity and security properties of authenticators, which are provided in a
1515 standard metadata format.

1516 Each FIDO authenticator has an attestation key pair and certificate. To maintain FIDO's privacy
1517 guarantees, these attestation keys are not unique for each device but are typically assigned on a
1518 manufacturing batch basis. During authenticator registration, the RP can check the validity of the
1519 attestation certificate and validate the signed registration data to verify that the authenticator
1520 possesses the private attestation key.

1521 For software authenticators, which cannot provide protection of a private attestation key, the UAF
1522 protocol allows for surrogate basic attestation. In this mode, the key pair generated to authenticate the
1523 user to the RP is used to sign the registration data object, including the attestation data. This is

1524 analogous to the use of self-signed certificates for https in that it does not actually provide
1525 cryptographic proof of the security properties of the authenticator. A potential concern is that the RP
1526 could not distinguish between a genuine software authenticator and a malicious look-alike
1527 authenticator that could provide registered credentials to an attacker. In an enterprise setting, this
1528 concern could be mitigated by delivering the valid authenticator application using EMM or another
1529 controlled distribution mechanism.

1530 Authenticator metadata would be most important in scenarios where an RP accepts multiple
1531 authenticators with different assurance levels and applies authorization policies based on the security
1532 properties of the authenticators (e.g., whether they provide FIPS 140-2-validated key storage [\[34\]](#)). In
1533 practice, most existing enterprise implementations use a single type of authenticator.

1534 [C.3.6 FIDO Deployment Considerations](#)

1535 To support any of the FIDO standards for authentication, some integration needs to happen on the
1536 server side. Depending on how the federated architecture is set up—whether with OIDC or SAML—this
1537 integration may look different. In general, there are two servers where a FIDO server can be integrated:
1538 the AS (also known as the RP) and the IdP.

1539 **FIDO Integration at the IdP**

1540 Primary authentication already happens at the IdP, so logic follows that FIDO authentication (e.g., U2F,
1541 UAF) would as well. This is the most common and well-understood model for using a FIDO
1542 authentication server and, consequently, there is solid guidance for setting up such an architecture. The
1543 IdP already has detailed knowledge of the user and directly interacts with the user (e.g., during
1544 registration), so it is not difficult to insert the FIDO server into the registration and authentication flows.
1545 In addition, this gives PSOs the most control over the security controls that are used to authenticate
1546 their users. However, there are a few downsides to this approach:

- 1547 ■ The PSO must now budget, host, manage, and/or pay for the cost of the FIDO server.
- 1548 ■ The only authentication of the user at the AS is the bearer assertion from the IdP, so an
1549 assertion intercepted by an attacker could be used to impersonate the legitimate user at the AS.

1550 **FIDO Integration at the AS**

1551 Another option is to integrate FIDO authentication at the AS. One benefit of this is that PSOs will not be
1552 responsible for the expenses of maintaining a FIDO server. In addition, an attacker who intercepted a
1553 valid user's SAML assertion or ID token could not easily impersonate the user because of the
1554 requirement to authenticate to the AS as well. This approach assumes that some mechanism is in place
1555 for tightly binding the FIDO authenticator with the user's identity, which is a nontrivial task. In addition,
1556 this approach has several downsides:

- 1557 ▪ Splitting authentication into a two-stage process that spans the IdP and AS is a less well
1558 understood model for authentication, which may lead to subtle issues.
- 1559 ▪ The AS does not have detailed knowledge of—or direct action with—users, so enrollment is
1560 more difficult.
- 1561 ▪ Users would have to register their FIDO authenticators at every AS that is federated to their IdP,
1562 which adds complexity and frustration to the process.
- 1563 ▪ PSOs would lose the ability to enforce which kinds of FIDO token(s) their users utilize.

1564 **Appendix D Acronyms**

AAL	Authenticator Assurance Level
ABAC	Attribute-Based Access Control
API	Application Programming Interface
AS	Authorization Server
BCP	Best Current Practice
CRADA	Cooperative Research and Development Agreement
CTAP	Client-to-Authenticator Protocol
EMM	Enterprise Mobility Management
FAL	Federation Assurance Level
FIDO	Fast Identity Online
FIPS	Federal Information Processing Standard
FirstNet	First Responder Network Authority
GPS	Global Positioning System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identification
IdP	Identity Provider
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
iOS	iPhone Operating System
ISO	International Organization for Standardization
IT	Information Technology
LOA	Level of Assurance
MF	Multifactor
MFA	Multifactor Authentication
MSSO	Mobile Single Sign-On
MTC	Mobile Threat Catalogue
NCCoE	National Cybersecurity Center of Excellence
NFC	Near Field Communication
NIEF	National Identity Exchange Federation
NIST	National Institute of Standards and Technology
NTP	Network Time Protocol
OEM	Original Equipment Manufacturer
OIDC	OpenID Connect
OOB	Out of Band
OS	Operating System
OTP	Onetime Password
PII	Personally Identifiable Information
PIN	Personal Identification Number

PKCE	Proof Key for Code Exchange
PSFR	Public Safety and First Responder
PSO	Public Safety Organization
PSX	Public Safety Experience
RFC	Request for Comments
RP	Relying Party
SaaS	Software as a Service
SAML	Security Assertion Markup Language
SDK	Software Development Kit
SF	Single Factor
SKCE	StrongKey Crypto Engine
SP	Special Publication
SSO	Single Sign-On
SwA	Software Assurance
TLS	Transport Layer Security
U2F	Universal Second Factor
UAF	Universal Authentication Framework
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium

1565 **Appendix E References**

- [1] W. Denniss and J. Bradley, *OAuth 2.0 for Native Apps*, Best Current Practice 212, Internet Engineering Task Force (IETF) Network Working Group Request for Comments (RFC) 8252, Oct. 2017. Available: <https://www.rfc-editor.org/info/rfc8252>.
- [2] S. Machani et al., *FIDO UAF Architectural Overview: FIDO Alliance Implementation Draft*, FIDO Alliance, Wakefield, Mass., 2017. Available: <https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-uaf-overview-v1.1-id-20170202.html>.
- [3] S. Srinivas et al., *Universal 2nd Factor (U2F) Overview: FIDO Alliance Proposed Standard*, FIDO Alliance, Wakefield, Mass., 2017. Available: <https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-overview-v1.2-ps-20170411.html>.
- [4] S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*, OASIS Standard, Mar. 2005. Available: <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>.
- [5] N. Sakimura et al., *OpenID Connect Core 1.0 incorporating errata set 1*, Nov. 2014. Available: http://openid.net/specs/openid-connect-core-1_0.html.
- [6] Joint Task Force Transformation Initiative, *Guide for Conducting Risk Assessments*, National Institute of Standards and Technology (NIST) Special Publication (SP) 800-30 Revision 1, Gaithersburg, Md., Sept. 2012. Available: <https://doi.org/10.6028/NIST.SP.800-30r1>.
- [7] Joint Task Force Transformation Initiative, *Guide for Applying the Risk Management Framework to Federal Information Systems: A Security Life Cycle Approach*, NIST SP 800-37 Revision 1, Gaithersburg, Md., Feb. 2010. Available: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r1.pdf>.
- [8] C. Johnson et al., *Guide to Cyber Threat Information Sharing*, NIST SP 800-150, Gaithersburg, Md., Oct. 2016. Available: <https://doi.org/10.6028/NIST.SP.800-150>.
- [9] C. Brown et al., *Assessing Threats to Mobile Devices & Infrastructure: The Mobile Threat Catalogue*, Draft NIST Interagency Report 8144, Gaithersburg, Md., Sept. 2016. Available: <https://nccoe.nist.gov/sites/default/files/library/mtc-nistir-8144-draft.pdf>.
- [10] P. Grassi et al., *Digital Identity Guidelines: Authentication and Lifecycle Management*, NIST SP 800-63B, Gaithersburg, Md., June 2017. Available: <https://doi.org/10.6028/NIST.SP.800-63b>.

- [11] P. Grassi et al., *Digital Identity Guidelines: Federation and Assertions*, NIST SP 800-63C, Gaithersburg, Md., June 2017. Available: <https://doi.org/10.6028/NIST.SP.800-63c>.
- [12] International Organization for Standardization/International Electrotechnical Commission/Institute of Electrical and Electronics Engineers, *Systems and software engineering—System life cycle processes*, ISO/IEC/IEEE 15288:2015, 2015. Available: <https://www.iso.org/standard/63711.html>.
- [13] R. Ross et al., *Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems*, NIST SP 800-160, Gaithersburg, Md., Nov. 2016. Available: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160v1.pdf>
- [14] AppAuth. *AppAuth*. Available: <https://appauth.io/>.
- [15] M. Jones and D. Hardt, *The OAuth 2.0 Authorization Framework: Bearer Token Usage*, IETF Network Working Group RFC 6750, Oct. 2012. Available: <https://www.rfc-editor.org/info/rfc6750>.
- [16] D. Balfanz et al., *Web Authentication: An API for accessing Public Key Credentials Level 1*, W3C Recommendation, Mar. 2019. Available: <https://www.w3.org/TR/webauthn/>.
- [17] P. Grassi et al., *Digital Identity Guidelines*, NIST SP 800-63-3, Gaithersburg, Md., June 2017. Available: <https://pages.nist.gov/800-63-3/>.
- [18] A. Popov et al., *The Token Binding Protocol Version 1.0*, IETF Network Working Group RFC 8471, Oct. 2018. Available: <https://www.rfc-editor.org/info/rfc8471>.
- [19] T. Lodderstedt, Ed., et al., *OAuth 2.0 Threat Model and Security Considerations*, IETF Network Working Group RFC 6819, Jan. 2013. Available: <https://www.rfc-editor.org/info/rfc6819>.
- [20] NIST. *NIST Internet Time Servers*. Available: <https://tf.nist.gov/tf-cgi/servers.cgi>.
- [21] P. Grassi et al., *Digital Identity Guidelines: Enrollment and Identity Proofing*, NIST SP 800-63A, Gaithersburg, Md., June 2017. Available: <https://doi.org/10.6028/NIST.SP.800-63a>.
- [22] J. Franklin et al., *Mobile Device Security: Cloud and Hybrid Builds*, NIST SP 1800-4, Gaithersburg, Md., Nov. 2015. Available: <https://www.nccoe.nist.gov/sites/default/files/library/sp1800/mds-nist-sp1800-4-draft.pdf>.
- [23] C. Brown et al., *Mobile Threat Catalogue*, NIST, 2016. Available: <https://pages.nist.gov/mobile-threat-catalogue/>.

- [24] Committee on National Security Systems (CNSS), *National Information Assurance (IA) Glossary*, CNSS Instruction Number 4009, Apr. 2015. Available: <https://rmf.org/wp-content/uploads/2017/10/CNSSI-4009.pdf>.
- [25] S. Quirolgico et al., *Vetting the Security of Mobile Applications*, NIST SP 800-163, Gaithersburg, Md., Jan. 2015. <https://doi.org/10.6028/NIST.SP.800-163>.
- [26] First Responder Network Authority. *FirstNet Developer Portal*. Available: <https://developer.firstnet.com/firstnet>.
- [27] M. Souppaya and K. Scarfone, *Guidelines for Managing the Security of Mobile Devices in the Enterprise*, NIST SP 800-124 Revision 1, Gaithersburg, Md., June 2013. Available: <https://doi.org/10.6028/NIST.SP.800-124r1>.
- [28] N. Sakimura et al., *Proof Key for Code Exchange by OAuth Public Clients*, IETF Network Working Group RFC 7636, Sept. 2015. Available: <https://www.rfc-editor.org/info/rfc7636>.
- [29] D. Hardt, Ed., *The OAuth 2.0 Authorization Framework*, IETF Network Working Group RFC 6749, Oct. 2012. Available: <https://www.rfc-editor.org/info/rfc6749>.
- [30] A. Popov et al., *Token Binding over HTTP*, IETF Network Working Group RFC 8473, Oct. 2018. Available: <https://www.rfc-editor.org/info/rfc8473>.
- [31] U.S. Department of Labor, Employee Benefits Security Administration. *Fact Sheet: The Health Insurance Portability and Accountability Act (HIPAA)*. Available: <https://permanent.access.gpo.gov/gpo10291/fshipaa.html>.
- [32] *Criminal Justice Information Services (CJIS) Security Policy*, Version 5.6, U.S. Department of Justice, Federal Bureau of Investigation, Criminal Justice Information Services Division, June 2017. Available: <https://www.fbi.gov/services/cjis/cjis-security-policy-resource-center>.
- [33] Joint Task Force Transformation Initiative, *Security and Privacy Controls for Federal Information Systems and Organizations*, NIST SP 800-53 Revision 4, Gaithersburg, Md., Jan. 2015. Available: <https://dx.doi.org/10.6028/NIST.SP.800-53r4>.
- [34] U.S. Department of Commerce. *Security Requirements for Cryptographic Modules*, Federal Information Processing Standards (FIPS) Publication 140-2, May 2001. Available: <https://doi.org/10.6028/NIST.FIPS.140-2>.

NIST SPECIAL PUBLICATION 1800-13C

Mobile Application Single Sign-On

Improving Authentication for Public Safety First Responders

Volume C: How-To Guides

Bill Fisher

Paul Grassi*

Applied Cybersecurity Division

Information Technology Laboratory

Spike E. Dog

Santos Jha

William Kim*

Taylor McCorkill

Joseph Portner*

Mark Russell

Sudhi Umarji

The MITRE Corporation

McLean, Virginia

William C. Barker

Dakota Consulting

Silver Spring, Maryland

**Former employee; all work for this publication was done while at employer.*

May 2019

SECOND 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 by name or company logo or other insignia in order to acknowledge their participation in this collaboration or to describe an experimental procedure or concept adequately. Such identification is not intended to imply special status or relationship with NIST or recommendation or endorsement by NIST or NCCoE; neither 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, 174 pages (May 2019), 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: May 29, 2019, through June 28, 2019

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

National Cybersecurity Center of Excellence
National Institute of Standards and Technology
100 Bureau Drive
Mailstop 2002
Gaithersburg, Maryland 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 information technology 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 Cybersecurity 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, Maryland.

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 1800 series) 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. This practice guide describes a reference design for multifactor authentication (MFA) and mobile single sign-on (MSSO) for native and web applications, while improving interoperability between mobile platforms, applications, and identity providers, irrespective of the application 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; relying party; single sign-on

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	StrongKey
Pushkar Marathe	StrongKey
Max Smyth	StrongKey
Scott Wong	StrongKey
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 Applications
Yubico	External Authenticators
Nok Nok Labs	Fast Identity Online (FIDO) Universal Authentication Framework (UAF) Server
StrongKey	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 on Android Devices.....	7
12	2.1.2 Supporting SSO on iOS Devices	8
13	2.1.3 Supporting FIDO U2F on Android Devices.....	10
14	2.1.4 Supporting FIDO U2F on iOS Devices	10
15	2.1.5 Supporting FIDO UAF.....	10
16	2.2 How to Install and Configure the Mobile Applications.....	11
17	2.2.1 How to Install and Configure SSO-Enabled Applications	11
18	2.2.2 How to Install and Configure a FIDO U2F Authenticator	23
19	2.2.3 How to Install and Configure a FIDO UAF Client	25
20	2.3 How Application Developers Must Integrate AppAuth for SSO.....	34
21	2.3.1 AppAuth Integration for Android	35
22	2.3.2 AppAuth Integration for iOS.....	43
23	3 How to Install and Configure the OAuth 2 AS.....	51
24	3.1 Platform and System Requirements	51
25	3.1.1 Software Requirements.....	51
26	3.1.2 Hardware Requirements	51
27	3.1.3 Network Requirements	52
28	3.2 How to Install the OAuth 2 AS.....	53
29	3.2.1 Java Installation	53

30	3.2.2 Java Post Installation	53
31	3.2.3 PingFederate Installation.....	55
32	3.2.4 Certificate Installation	55
33	3.3 How to Configure the OAuth 2 AS.....	55
34	3.4 How to Configure the OAuth 2 AS for Authentication.....	69
35	3.4.1 How to Configure Direct Authentication.....	70
36	3.4.2 How to Configure SAML Authentication	79
37	3.4.3 How to Configure OIDC Authentication	86
38	3.4.4 How to Configure the Authentication Policy.....	93
39	4 How to Install and Configure the Identity Providers	99
40	4.1 How to Configure the User Store.....	99
41	4.2 How to Install and Configure the SAML Identity Provider	103
42	4.2.1 Configuring Authentication to the IdP	105
43	4.2.2 Configure the SP Connection.....	115
44	4.3 How to Install and Configure the OIDC Identity Provider	122
45	4.3.1 Configuring Authentication to the OIDC IdP	123
46	4.3.2 Configuring the OIDC Client Connection	135
47	5 How to Install and Configure the FIDO UAF Authentication Server ...	137
48	5.1 Platform and System Requirements	138
49	5.1.1 Hardware Requirements	138
50	5.1.2 Software Requirements.....	138
51	5.2 How to Install and Configure the FIDO UAF Authentication Server	139
52	5.3 How to Install and Configure the FIDO UAF Gateway Server	140
53	5.4 How to Install and Configure the FIDO UAF Adapter for the OAuth 2 AS.....	140
54	6 How to Install and Configure the FIDO U2F Authentication Server ...	141
55	6.1 Platform and System Requirements	141
56	6.1.1 Software Requirements.....	141
57	6.1.2 Hardware Requirements	142
58	6.1.3 Network Requirements	142

59	6.2 How to Install and Configure the FIDO U2F Authentication Server.....	143
60	6.3 How to Install and Configure the FIDO U2F Adapter for the IdP	147
61	6.3.1 FIDO U2F Registration in Production.....	148
62	7 Functional Tests	148
63	7.1 Testing FIDO Authenticators	148
64	7.2 Testing FIDO Servers.....	149
65	7.3 Testing IdPs.....	149
66	7.4 Testing the AS.....	155
67	7.5 Testing the Application.....	157
68	Appendix A Abbreviations and Acronyms.....	158
69	Appendix B References	160

70 **List of Figures**

71	Figure 1-1 Lab Build Architecture.....	3
72	Figure 2-1 Comparison of UAF and U2F Standards.....	7
73	Figure 2-2 SFAuthenticationSession Consent Prompt	8
74	Figure 2-3 Safari Transition Prompt	9
75	Figure 2-4 FIDO UAF Architectural Overview	11
76	Figure 2-5 PSX Cockpit Setup	13
77	Figure 2-6 PSX Cockpit Setup, Continued.....	14
78	Figure 2-7 PSX Cockpit Group List Selection.....	15
79	Figure 2-8 PSX Cockpit Groups.....	16
80	Figure 2-9 PSX Cockpit Group List Setup Complete	17
81	Figure 2-10 PSX Cockpit User Interface	18
82	Figure 2-11 PSX Mapping User Interface	19
83	Figure 2-12 PSX Mapping Group Member Information	20
84	Figure 2-13 PSX Messenger User Interface	21

85	Figure 2-14 PSX Messenger Messages.....	22
86	Figure 2-15 FIDO U2F Registration	24
87	Figure 2-16 FIDO U2F Authentication.....	25
88	Figure 2-17 Nok Nok Labs Tutorial Application Authentication	27
89	Figure 2-18 Nok Nok Labs Tutorial Application Login	28
90	Figure 2-19 FIDO UAF Registration Interface	29
91	Figure 2-20 FIDO UAF Registration QR Code.....	30
92	Figure 2-21 FIDO UAF Registration Device Flow, Android Device.....	31
93	Figure 2-22 FIDO UAF Registration Device Flow, iPhone X.....	32
94	Figure 2-23 FIDO UAF Fingerprint Authenticator, Android Device	33
95	Figure 2-24 FIDO UAF Registration Success	34
96	Figure 2-25 Linked Frameworks and Libraries.....	44
97	Figure 2-26 Creating a New Run Script Phase	45
98	Figure 2-27 Carthage Run Script.....	45
99	Figure 2-28 Custom URL Scheme.....	47
100	Figure 3-1 Access Token Attribute Mapping Framework.....	56
101	Figure 3-2 Server Roles for AS.....	59
102	Figure 3-3 Federation Info	60
103	Figure 3-4 AS Settings.....	61
104	Figure 3-5 Scopes	63
105	Figure 3-6 Access Token Management Instance	64
106	Figure 3-7 Access Token Manager Instance Configuration	65
107	Figure 3-8 Access Token Manager Attribute Contract	66
108	Figure 3-9 OAuth Client Registration, Part 1	67
109	Figure 3-10 OAuth Client Registration, Part 2	68
110	Figure 3-11 Create Adapter Instance	71
111	Figure 3-12 FIDO Adapter Settings	72
112	Figure 3-13 FIDO Adapter Contract	73

113	Figure 3-14 FIDO Adapter Instance Summary	74
114	Figure 3-15 Policy Contract Information	75
115	Figure 3-16 Policy Contract Attributes	75
116	Figure 3-17 Create Authentication Policy Contract Mapping	76
117	Figure 3-18 Authentication Policy Contract Fulfillment	77
118	Figure 3-19 Create Access Token Attribute Mapping	78
119	Figure 3-20 Access Token Mapping Contract Fulfillment	78
120	Figure 3-21 Create IdP Connection	80
121	Figure 3-22 IdP Connection Options	80
122	Figure 3-23 IdP Connection General Info	81
123	Figure 3-24 IdP Connection—User-Session Creation	82
124	Figure 3-25 IdP Connection OAuth Attribute Mapping	83
125	Figure 3-26 IdP Connection—Protocol Settings	84
126	Figure 3-27 Policy Contract for SAML RP	85
127	Figure 3-28 Contract Mapping for SAML RP	86
128	Figure 3-29 IdP Connection Type	87
129	Figure 3-30 IdP Connection Options	87
130	Figure 3-31 IdP Connection General Info	88
131	Figure 3-32 IdP Connection Authentication Policy Contract	89
132	Figure 3-33 IdP Connection Policy Contract Mapping	90
133	Figure 3-34 IdP Connection OAuth Attribute Mapping	91
134	Figure 3-35 IdP Connection Protocol Settings	92
135	Figure 3-36 IdP Connection Activation and Summary	93
136	Figure 3-37 Authentication Selector Instance	94
137	Figure 3-38 Authentication Selector Details	95
138	Figure 3-39 Selector Result Values	96
139	Figure 3-40 Policy Settings	96
140	Figure 3-41 Authentication Policy	97

141	Figure 3-42 Policy Contract Mapping for IdP Connections	98
142	Figure 3-43 Policy Contract Mapping for Local Authentication	99
143	Figure 4-1 Active Directory Users and Computers.....	100
144	Figure 4-2 Server Configuration	101
145	Figure 4-3 Data Store Type	102
146	Figure 4-4 LDAP Data Store Configuration.....	103
147	Figure 4-5 Server Roles for SAML IdP	104
148	Figure 4-6 SAML IdP Federation Info	105
149	Figure 4-7 Create Password Credential Validator.....	106
150	Figure 4-8 Credential Validator Configuration	107
151	Figure 4-9 Password Credential Validator Extended Contract	108
152	Figure 4-10 Password Validator Summary.....	109
153	Figure 4-11 HTML Form Adapter Instance	110
154	Figure 4-12 Form Adapter Settings.....	111
155	Figure 4-13 Form Adapter Extended Contract.....	112
156	Figure 4-14 Create U2F Adapter Instance	113
157	Figure 4-15 U2F Adapter Settings.....	114
158	Figure 4-16 IdP Authentication Policy	115
159	Figure 4-17 SP Connection Type.....	116
160	Figure 4-18 SP Connection General Info	117
161	Figure 4-19 SP Browser SSO Profiles	118
162	Figure 4-20 Assertion Identity Mapping	119
163	Figure 4-21 Assertion Attribute Contract.....	119
164	Figure 4-22 Assertion Attribute Contract Fulfillment	120
165	Figure 4-23 Browser SSO Protocol Settings.....	121
166	Figure 4-24 OIDC IdP Roles	122
167	Figure 4-25 Create Access Token Manager	124
168	Figure 4-26 Access Token Manager Configuration	125

169	Figure 4-27 Access Token Attribute Contract	126
170	Figure 4-28 Access Token Contract Fulfillment	127
171	Figure 4-29 Data Store for User Lookup	128
172	Figure 4-30 Attribute Directory Search	129
173	Figure 4-31 Access Token Contract Fulfillment	130
174	Figure 4-32 Access Token Issuance Criteria	131
175	Figure 4-33 OIDC Policy Creation	132
176	Figure 4-34 OIDC Policy Attribute Contract	133
177	Figure 4-35 OIDC Policy Contract Fulfillment	134
178	Figure 4-36 OIDC Client Configuration	136
179	Figure 6-1 Glassfish SSL Settings	146
180	Figure 7-1 Using Postman to Obtain the ID Token	154
181	Figure 7-2 Authorization Prompt	156
182	Figure 7-3 Token Introspection Request and Response	157

183 1 Introduction

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

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

194 1.1 Practice Guide Structure

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

199 This guide contains three volumes:

- 200 ▪ NIST SP 1800-13A: *Executive Summary*
- 201 ▪ NIST SP 1800-13B: *Approach, Architecture, and Security Characteristics*—what we built and why
- 202 ▪ NIST SP 1800-13C: *How-To Guides*—instructions for building the example solution (**you are here**)

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

205 1.2 Build Overview

206 The National Cybersecurity Center of Excellence (NCCoE) worked with its build team partners to create a
207 lab demonstration environment that includes all of the architectural components and functionality
208 described in Section 4 of Volume B of this build guide. This includes mobile devices with sample
209 applications, hardware and software-based authenticators to demonstrate the Fast Identity Online
210 (FIDO) standards for MFA, and the authentication server and authorization server (AS) components
211 required to demonstrate the AppAuth authorization flows (detailed in Internet Engineering Task Force
212 [IETF] Request for Comments [RFC] 8252 [\[1\]](#)) with federated authentication to a Security Assertion
213 Markup Language (SAML) Identity Provider (IdP) and an OpenID Connect (OIDC) provider. The complete

214 build includes several systems deployed in the NCCoE lab by StrongKey, Yubico, and Ping Identity as well
215 as cloud-hosted resources made available by Motorola Solutions and by Nok Nok Labs.

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

218 **1.2.1 Usage Scenarios**

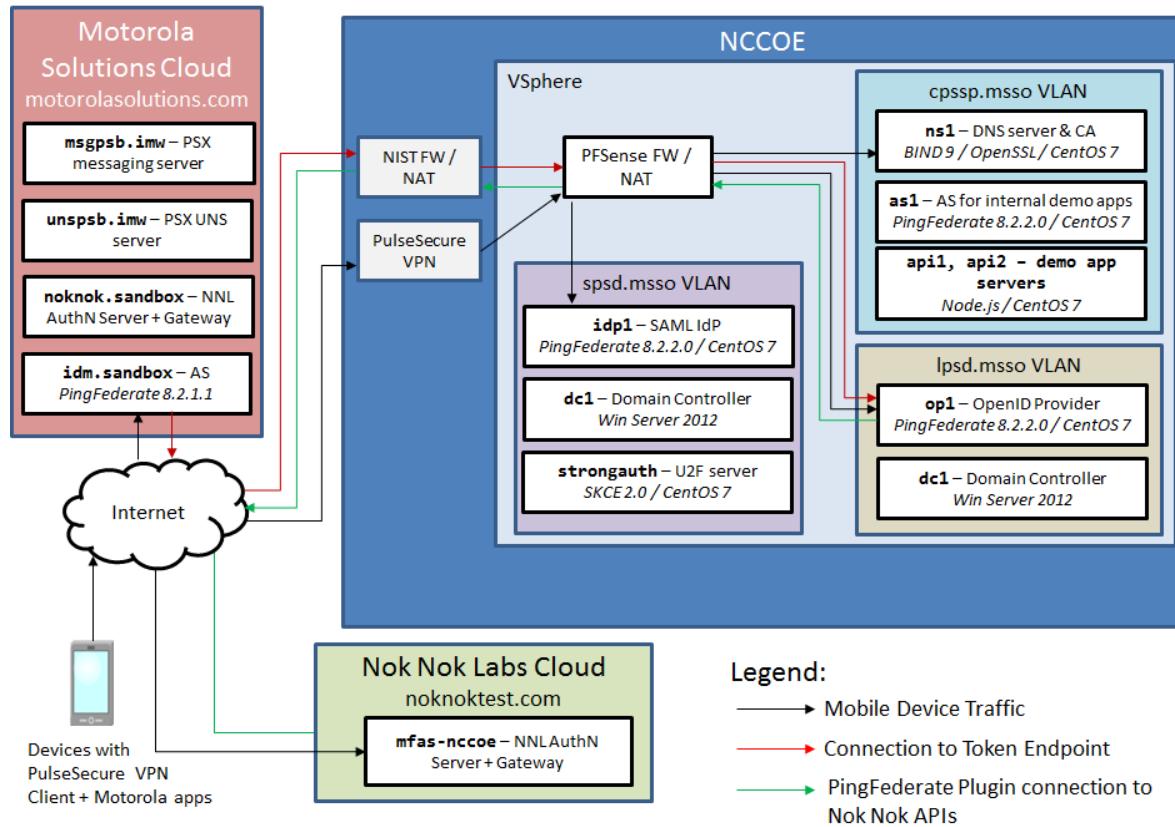
219 The build architecture supports three usage scenarios. The scenarios all demonstrate single sign-on
220 (SSO) among Motorola Solutions Public Safety Experience (PSX) applications and custom-built Apple
221 iPhone operating system (iOS) demo applications using the AppAuth pattern but differ in the details of
222 the authentication process. The three authentication mechanisms are as follows:

- 223 ■ The OAuth AS directly authenticates the user with FIDO Universal Authentication Framework
224 (UAF); user accounts are managed directly by the service provider.
- 225 ■ The OAuth AS redirects the user to a SAML IdP, which authenticates the user with a password
226 and FIDO Universal Second Factor (U2F).
- 227 ■ The OAuth AS redirects the user to an OIDC IdP, which authenticates the user with FIDO UAF.

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

238 **1.2.2 Architectural Overview**

239 Figure 1-1 shows the lab build architecture.

240 **Figure 1-1 Lab Build Architecture**

241

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

- Motorola Solutions cloud—a cloud-hosted environment providing the back-end application servers for the Motorola Solutions PSX Mapping and Messaging applications, as well as an OAuth AS that the application servers use to authorize requests from mobile devices
- Nok Nok Labs cloud—a cloud-hosted server running both the Nok Nok Authentication Server (NNAS) and the Nok Nok Labs Gateway
- 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 application back ends are also hosted in the NCCoE lab for internal testing.

- 253 ■ mobile devices connected to public cellular networks with the required client software to
254 authenticate to, and access, Motorola Solutions back-end applications and the NCCoE lab
255 systems

256 The names of the virtual local area networks (VLANs) in the NCCoE lab are meant to depict different
257 organizations participating in an MSSO scheme:

- 258 ■ PSD—State Public Safety Department, a PSFR organization with a SAML IdP
259 ■ LPSP—Local Public Safety Department, a PSFR organization with an OIDC IdP
260 ■ CPSSP—Central Public Safety Service Provider, a software as a service (SaaS) provider serving the
261 PSFR organizations, analogous to Motorola Solutions

262 The fictitious *.mssso* top-level domain is simply a reference to the MSSO project. The demonstration
263 applications hosted in the CPSSP VLAN were used to initially test and validate the federation setups in
264 the user organization and were later expanded to support the iOS demonstration build.

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

268 The depicted traffic flows are described below:

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

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

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

297 **1.2.3 General Infrastructure Requirements**

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

- 301 ■ DNS—All server names are expected to be resolvable in a DNS. This is especially important for
 302 FIDO functionality, as the application identification (App ID) associated with cryptographic keys
 303 is derived from the host name used in application uniform resource locators (URLs).
- 304 ■ Network Time Protocol (NTP)—Time synchronization among servers is important. A clock
 305 difference of five minutes or more is sufficient to cause JavaScript Object Notation (JSON) Web
 306 Token (JWT) validation, for example, to fail. All servers should be configured to synchronize time
 307 with a reliable NTP source.
- 308 ■ Certificate Authority (CA)—Hypertext Transfer Protocol Secure ([https](https://)) connections should be
 309 used throughout the architecture. Transport Layer Security (TLS) certificates are required for all
 310 servers in the build. If an in-house CA is used to issue certificates, the root and any intermediate
 311 certificates must be provisioned to the trust stores in client mobile devices and servers.

312 **1.3 Typographic Conventions**

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

Typeface/ Symbol	Meaning	Example
<i>Italics</i>	file names and path names, references to documents that are not hyperlinks, new terms, and placeholders	For detailed definitions of terms, see the <i>NCCoE Glossary</i> .

Typeface/ Symbol	Meaning	Example
Bold	names of menus, options, command buttons, and fields	Choose File > Edit .
Monospace	command-line input, onscreen computer output, sample code examples, and status codes	<code>mkdir</code>
Monospace Bold	command-line user input contrasted with computer output	service sshd start
blue text	link to other parts of the document, a web URL, or an email address	All publications from NIST's NCCoE are available at https://www.nccoe.nist.gov .

314 2 How to Install and Configure the Mobile Device

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

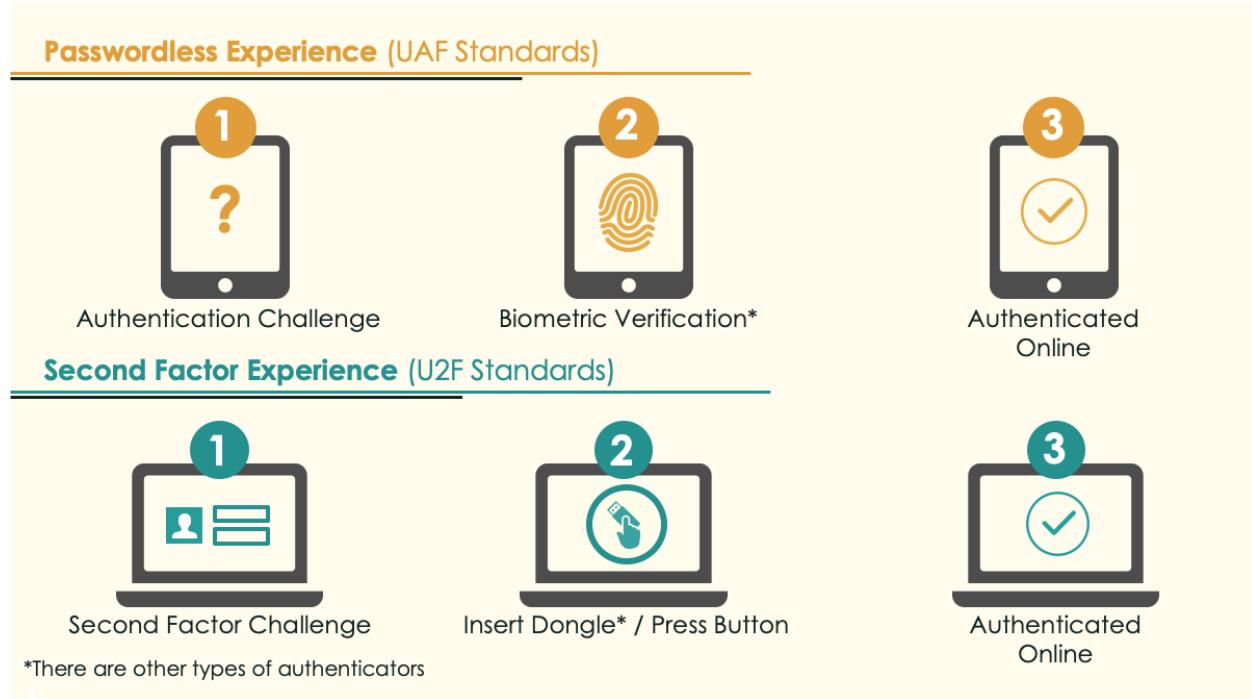
318 2.1 Platform and System Requirements

319 This section covers requirements for mobile devices—both hardware and software—for the SSO and
 320 FIDO authentication components of the architecture to work properly. The two dominant mobile
 321 platforms are Google’s Android and Apple’s iOS. The NCCoE reference architecture incorporates both
 322 iOS and Android devices and applications.

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

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

331 Figure 2-1 Comparison of UAF and U2F Standards



332

333 The following subsections address mobile device requirements to support SSO and FIDO authentication.

334 2.1.1 Supporting SSO on Android Devices

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

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

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

345 Any device manufacturer can support Custom Tabs in its Android browser. However, Google
346 implemented this in its Chrome for Android web browser in September 2015 [3]. Because Chrome is not
347 part of the operating system (OS) itself but is downloaded from the Google Play Store, recent versions of
348 Chrome can be used on older versions of Android. In fact, the Chrome Developer website’s page on

349 Chrome Custom Tabs [4] states that it can be used on Android Jelly Bean (4.1), which was released in
350 2012, and up.

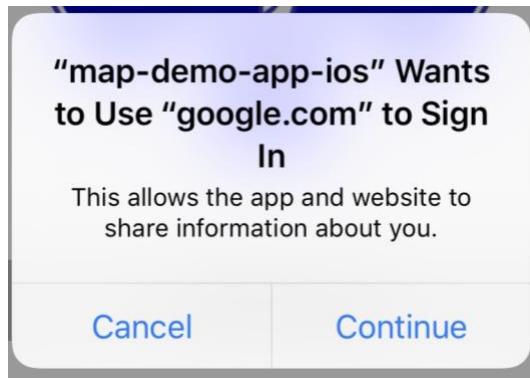
351 To demonstrate SSO, the NCCoE reference architecture utilizes the Motorola Solutions PSX App Suite,
352 which requires Android Lollipop (5.0) or newer.

353 **2.1.2 Supporting SSO on iOS Devices**

354 Apple's Safari browser is the default external user-agent provided on iOS devices, and iOS has also
355 supported in-application browser tabs with the SFSafariViewController API [5] since iOS 9. Like Chrome
356 Custom Tabs, SFSafariViewController provides the functionality of the OS browser without exiting from
357 the mobile application.

358 Apple made changes to its in-application browser tab implementation in iOS 11 [6] that impacted SSO
359 functionality. SFSafariViewController instances created by different applications are now effectively
360 sandboxed from each other, with no shared cookie store between them. As described in Section 4.4 of
361 Volume B of this practice guide, the AppAuth pattern depends on shared cookie storage to provide SSO
362 between applications. Apple introduced a new API called SFAuthenticationSession to provide an in-
363 application browser tab implementation specifically for authentication with SSO capabilities with access
364 to the shared Safari cookie store. iOS also prompts for the user's consent when SFAuthenticationSession
365 is used. An example of the consent prompt is shown in Figure 2-2.

366 **Figure 2-2 SFAuthenticationSession Consent Prompt**



367

368 In iOS 12, Apple replaced the SFAuthenticationSession API with ASWebAuthenticationSession [7], which
369 performs the same functions as SFAuthenticationSession and presents an identical consent prompt. In
370 lab testing, the build team frequently encountered issues with SFAuthenticationSession where cookies
371 created in an SFAuthenticationSession spawned by one application were not available in an
372 SFAuthenticationSession spawned by another application. When this issue occurred, users would be
373 prompted to authenticate in each application that was launched and SSO did not function properly. The
374 team has not encountered these issues with ASWebAuthenticationSession, and the SSO capabilities of
375 in-application browser tabs are much improved in iOS 12.

376 By default, the AppAuth library for iOS [8] automatically selects an appropriate user-agent based on the
 377 version of iOS installed on the mobile device as shown in Table 2-1.

378 **Table 2-1 AppAuth User-Agent by iOS Version**

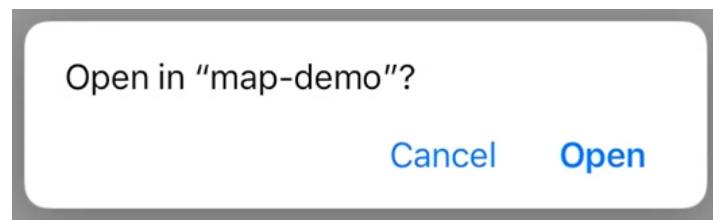
iOS Version	User-Agent
12 and higher	ASWebAuthenticationSession
11	SFAuthenticationSession
9 or 10	SFSafariViewController
8 and lower	Safari

379

380 The build team encountered issues with the FIDO UAF login flow demonstrated in this practice guide
 381 and the iOS in-application browser tab APIs (SFAuthenticationSession and
 382 ASWebAuthenticationSession). In the demo scenario, the login flow begins in the browser, which then
 383 launches the Passport application for user verification and FIDO authentication, and then control is
 384 returned to the browser to complete the authentication flow and return the user to the application.
 385 With ASWebAuthenticationSession, the authentication flow begins successfully in an in-application
 386 browser tab, and the user is redirected to the Passport application to authenticate, but control is not
 387 properly returned to the in-application browser tab when the Passport application closes. See Section
 388 4.3.2 of Volume B of this practice guide for additional details about this issue. The build team speculates
 389 that this issue would generally apply to any login flow that entails launching an external application and
 390 then returning control to an in-application browser tab.

391 This issue was resolved by overriding the default user-agent selection in the AppAuth library. AppAuth
 392 provides the OIDEExternalUserAgentIOSCustomBrowser interface to enable an application to specify the
 393 user-agent that should be used for the login flow. The iOS demo applications were configured to use the
 394 Safari browser instead of an in-application browser tab, which enabled the UAF login flow to succeed.
 395 The user experience with Safari is very similar to that with ASWebAuthenticationSession. The animation
 396 shown when transitioning to the web session is slightly different, and the consent dialogue shown in
 397 Figure 2-2 is not shown. After authentication is completed, however, a different dialogue is displayed,
 398 prompting the user to open the mobile application as shown in Figure 2-3.

399 **Figure 2-3 Safari Transition Prompt**



400

401 **2.1.3 Supporting FIDO U2F on Android Devices**

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

- 403 □ a web browser compatible with FIDO U2F
404 □ a FIDO U2F client application capable of handling the challenge
405 □ Near Field Communication (NFC) hardware support

406 Chrome for Android [9] is a U2F-compatible browser. Google has added U2F functionality to the Google
407 Play Services component of Android [10], so devices running Android 5 and later can natively support
408 U2F authentication over NFC, Universal Serial Bus (USB), and Bluetooth Low Energy (BLE) with an over-
409 the-air update to Play Services. To support U2F in the browser, the Google Authenticator application
410 [11] (available on Android Gingerbread [2.3.3] and up) must also be installed.

411 **2.1.4 Supporting FIDO U2F on iOS Devices**

412 At the time of writing, the U2F login flow demonstrated in this practice guide cannot be implemented on
413 iOS devices. Apple's Core NFC APIs do not expose required functionality to implement U2F over NFC.
414 Yubico has published an API enabling the YubiKey Neo to be used for authentication over NFC with an
415 iOS device, but this implementation uses the onetime password authentication mechanism of the
416 YubiKey, not the U2F protocol [12]. BLE U2F authenticators can be paired and used with iOS devices, but
417 their use has been limited. The Google Smart Lock application, which protects Google accounts with U2F
418 authentication on iOS devices, is the only notable U2F implementation on iOS of which the build team is
419 aware.

420 Yubico has announced development of an authenticator with a Lightning adapter, specifically targeting
421 iOS and Mac devices; and a corresponding mobile software development kit (SDK) for iOS that could
422 enable U2F authentication in native iOS applications [13]. To enable the AppAuth login flow used in this
423 practice guide, a U2F-capable browser is also needed. If Apple adds W3C Web Authentication support to
424 the Safari browser, it may support U2F authentication over Lightning and BLE in the future. Apple has
425 already added experimental support to the Safari Technology Preview release for Mac OS [14].

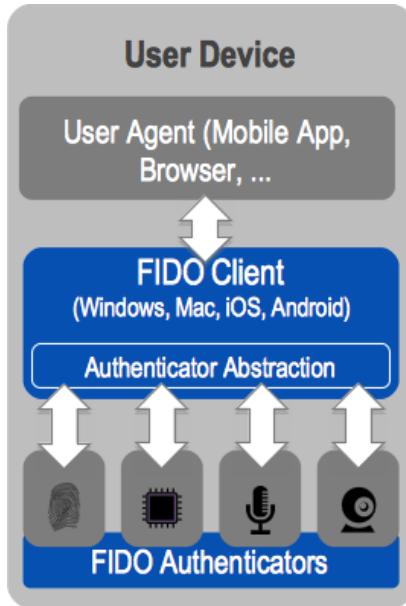
426 **2.1.5 Supporting FIDO UAF**

427 Supporting FIDO UAF is fairly similar on Android and iOS devices. The device will need the following
428 components for FIDO UAF:

- 429 □ a web browser
430 □ a FIDO UAF client application capable of handling the challenge
431 □ a FIDO UAF authenticator

432 These components are pictured in Figure 2-4, which is from the *FIDO UAF Architectural Overview* [15].

433 Figure 2-4 FIDO UAF Architectural Overview



434

435 While the overview refers to the last two components (client and authenticator) as separate
 436 components, these components can—and often do—come packaged in a single application. The NCCoE
 437 reference architecture utilizes the Nok Nok Passport application for Android [16] and iOS [17] to provide
 438 these two components. In addition to the applications, the device will need to provide some hardware
 439 component to support the FIDO UAF authenticator. For example, for biometric-based FIDO UAF
 440 authenticators, a camera would be needed to support face or iris scanning, a microphone would be
 441 needed to support voice prints, and a fingerprint sensor would be needed to support fingerprint
 442 biometrics. Of course, if a personal identification number (PIN) authenticator is used, a specific
 443 hardware sensor is not required. Beyond the actual input method of the FIDO UAF factor, additional
 444 (optional) hardware considerations for a UAF authenticator include secure key storage for registered
 445 FIDO key pairs, storage of biometric templates, and execution of matching functions (e.g., within
 446 dedicated hardware or on processor trusted execution environments).

447 2.2 How to Install and Configure the Mobile Applications

448 This section covers the installation and configuration of the mobile applications needed for various
 449 components of the reference architecture: SSO, FIDO U2F, and FIDO UAF.

450 2.2.1 How to Install and Configure SSO-Enabled Applications

451 For SSO-enabled applications, there is no universal set of installation and configuration procedures;
 452 these will vary depending on the design choices of the application manufacturer. For the Android demo,
 453 the NCCoE reference architecture uses the *Motorola Solutions PSX App Suite* [18] Version 5.4. This set of

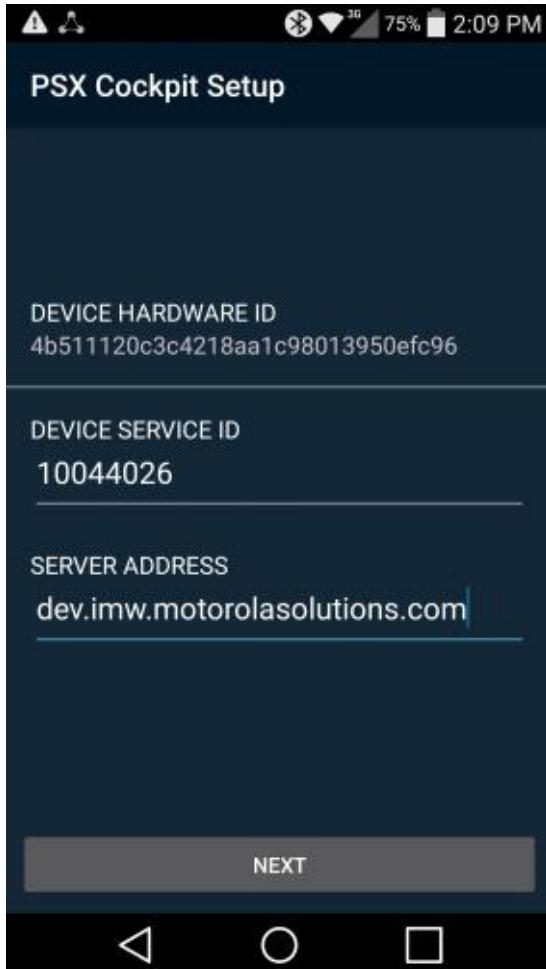
454 mobile applications provides several capabilities for the public safety community. Our setup consisted of
455 three applications: *PSX Messenger* for text, photo, and video communication; *PSX Mapping* for shared
456 location awareness; and *PSX Cockpit* to centralize authentication and identity information across the
457 other applications. These applications cannot be obtained from a public venue (e.g., the Google Play
458 Store); rather, the binaries must be obtained from Motorola Solutions and installed via other means,
459 such as a Mobile Device Management (MDM) solution or private application store.

460 For the iOS demo, the team built two iOS demonstration applications—a mapping application called
461 map-demo and a chat application called chat-demo. These applications were built by using Apple's
462 XCode integrated development environment and installed on lab devices using developer certificates.

463 ***2.2.1.1 Configuring the PSX Cockpit Application***

464 1. Open the Cockpit application. Your screen should look like Figure 2-5.

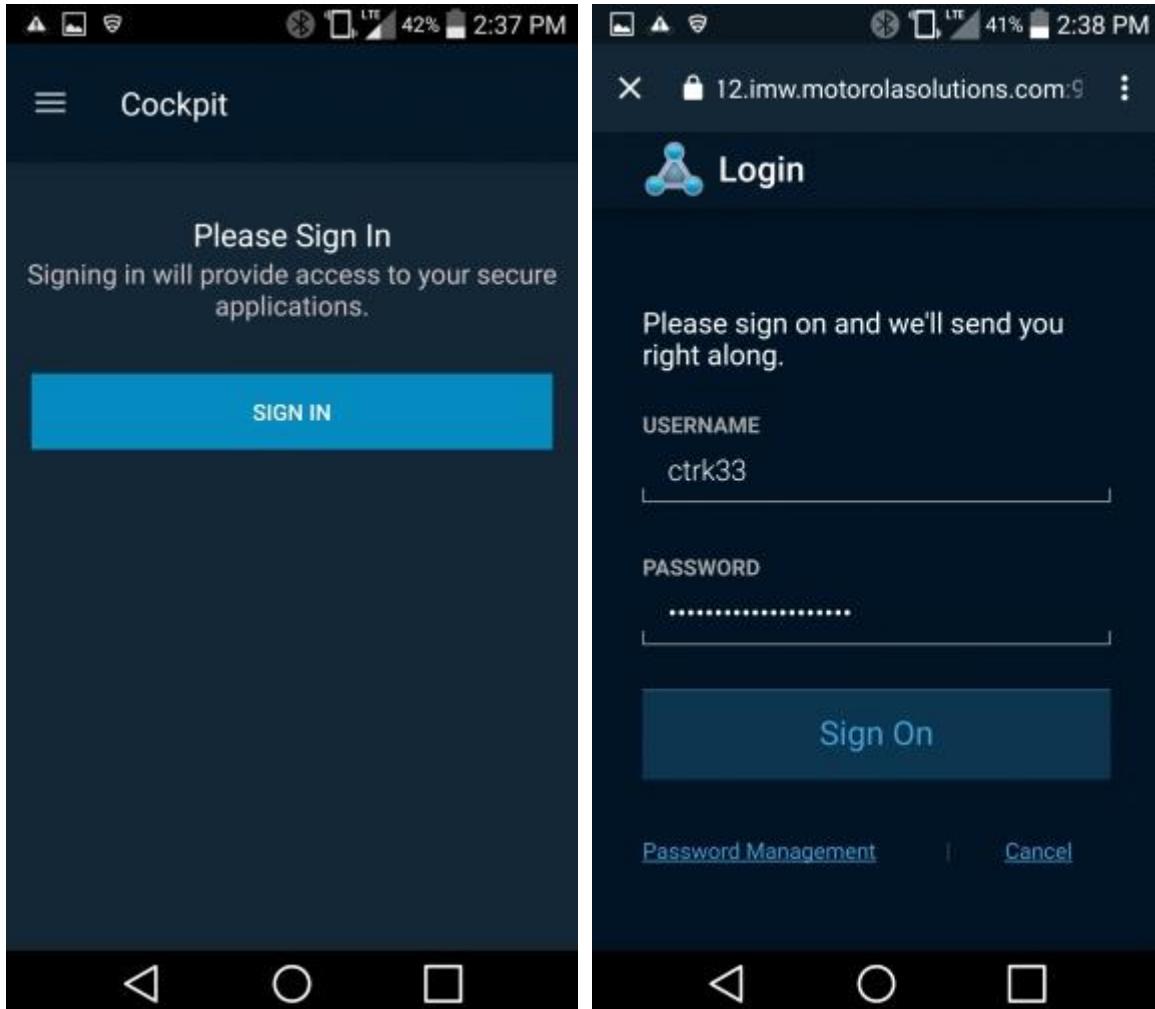
465 Figure 2-5 PSX Cockpit Setup



466

- 467 2. For **DEVICE SERVICE ID**, select a Device Service ID in the range given to you by your
468 administrator. Note that these details will be provided by Motorola Solutions if you are using
469 their service offering, or by your administrator if you are hosting the PSX application servers in
470 your own environment. Each device should be configured with a unique Device Service ID
471 corresponding to the username from the username range. For example, the NCCoE lab used a
472 Device Service ID of 22400 to correspond to a username of 2400.
- 473 3. For **SERVER ADDRESS**, use the Server Address given to you by your administrator. For example,
474 the NCCoE lab used a Server Address of uns5455.imw.motorolasolutions.com.
- 475 4. If a **Use SUPL APN** checkbox appears, leave it unchecked.
- 476 5. Tap **NEXT**. Your screen should look like Figure 2-6.

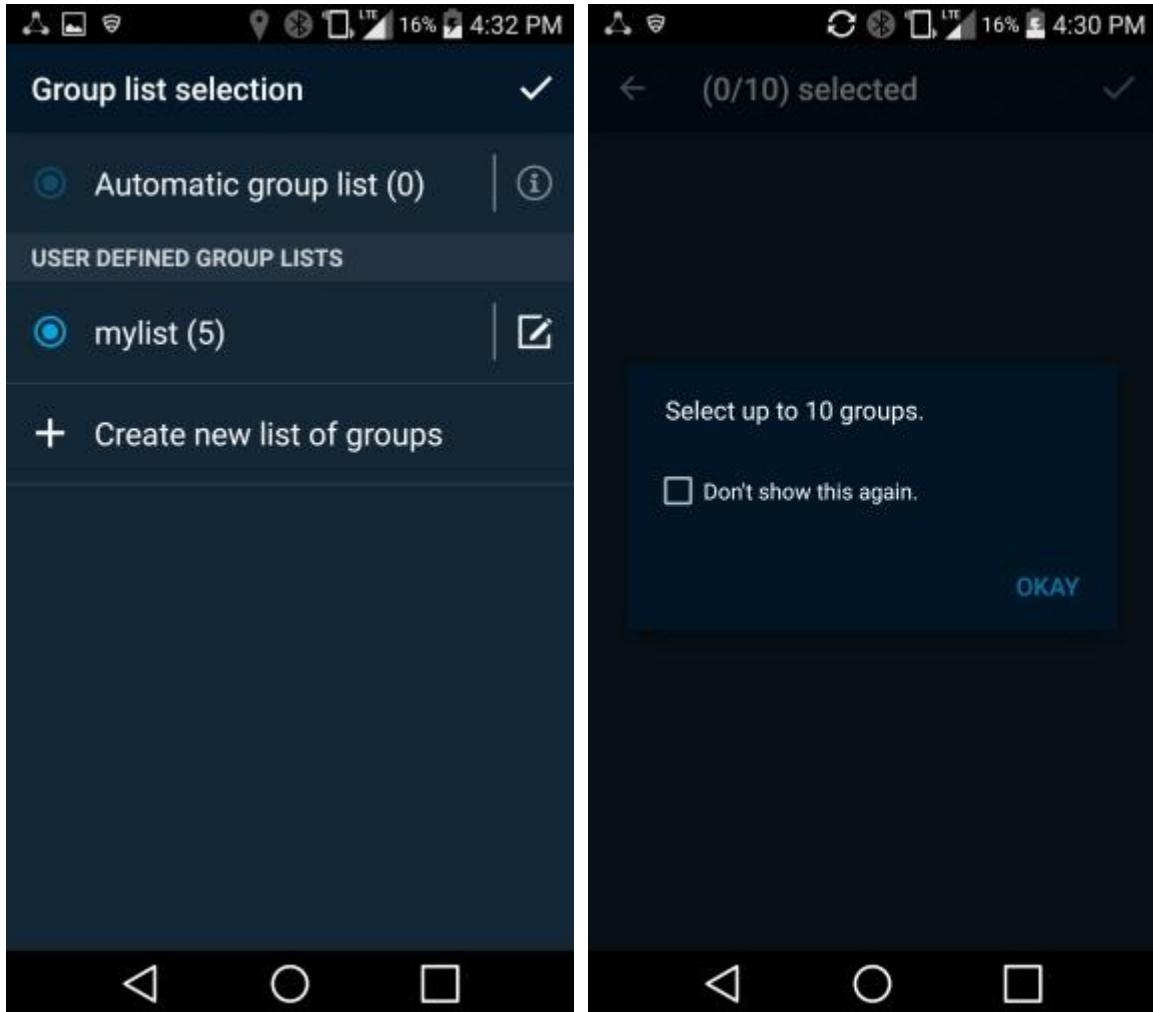
477 Figure 2-6 PSX Cockpit Setup, Continued



478

- 479 6. Tap **SIGN IN**.
- 480 7. Log in with the authentication procedure determined by the AS and IdP policies. Note that if
481 UAF is used, a FIDO UAF authenticator must be enrolled before this step can be completed. See
482 [Section 2.2.3](#) for details on FIDO UAF enrollment. After you log in, your screen should look like
483 Figure 2-7.

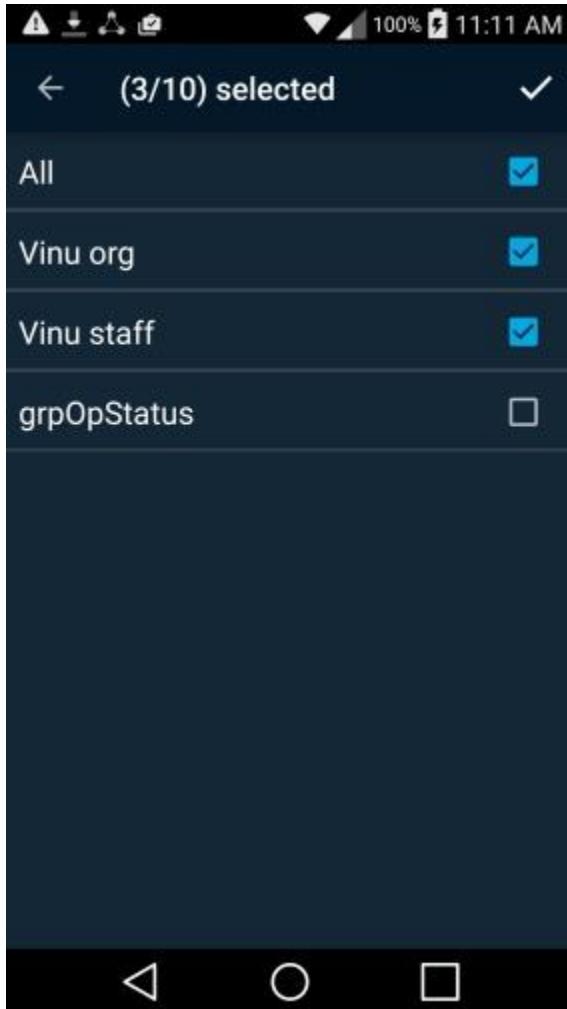
484 Figure 2-7 PSX Cockpit Group List Selection



485

- 486 8. Tap **Create new list of groups**. This is used to select which organizationally defined groups of
487 users you can receive data updates for in the other PSX applications.
- 488 9. Tap **OKAY**. Your screen should look like Figure 2-8.

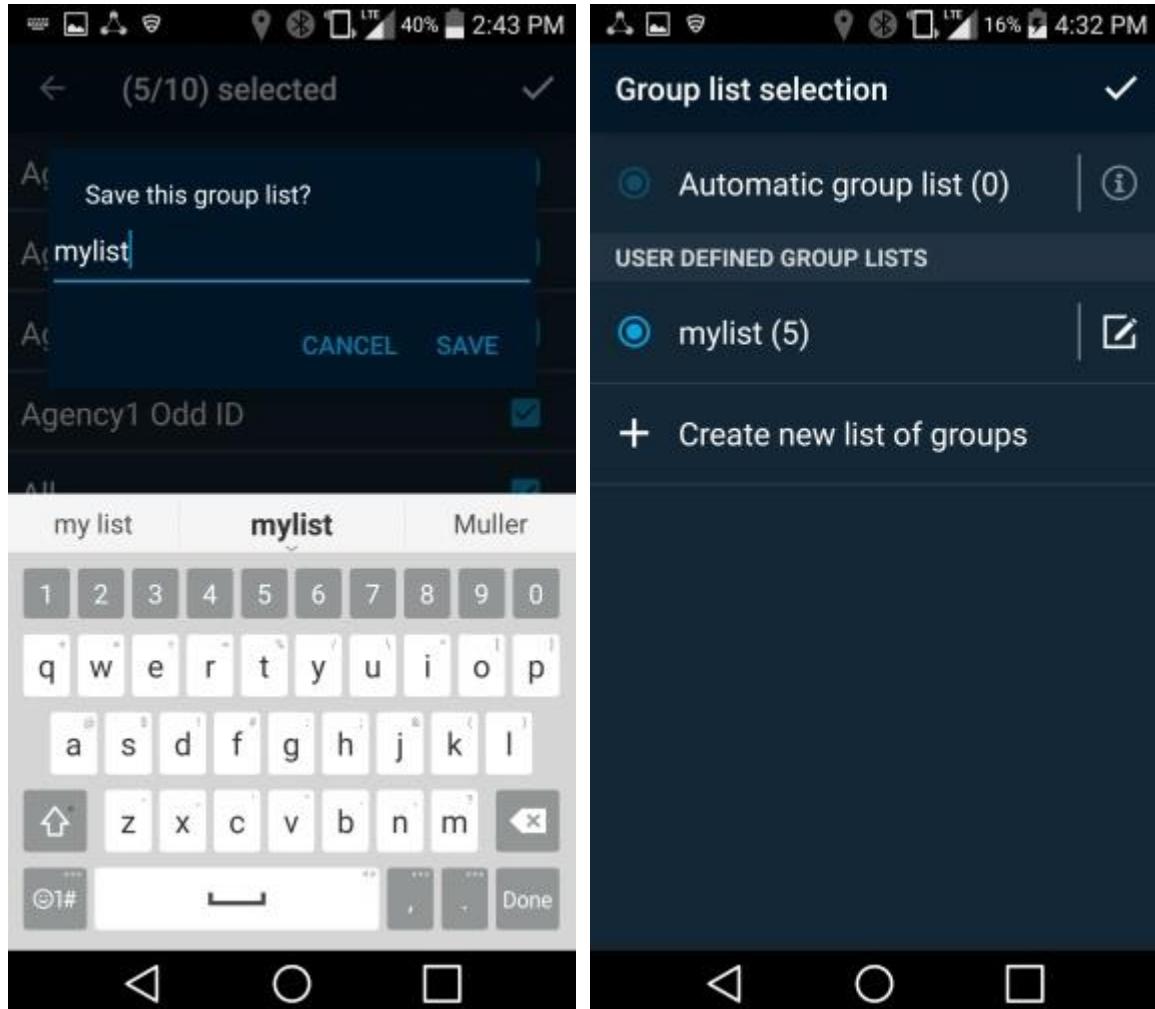
489 Figure 2-8 PSX Cockpit Groups



490

- 491 10. Check the checkboxes for the groups that you wish to use. Note that it may take a short time for
492 the groups to appear.
- 493 11. Tap on the upper-right check mark. Your screen should look like Figure 2-9.

494 Figure 2-9 PSX Cockpit Group List Setup Complete

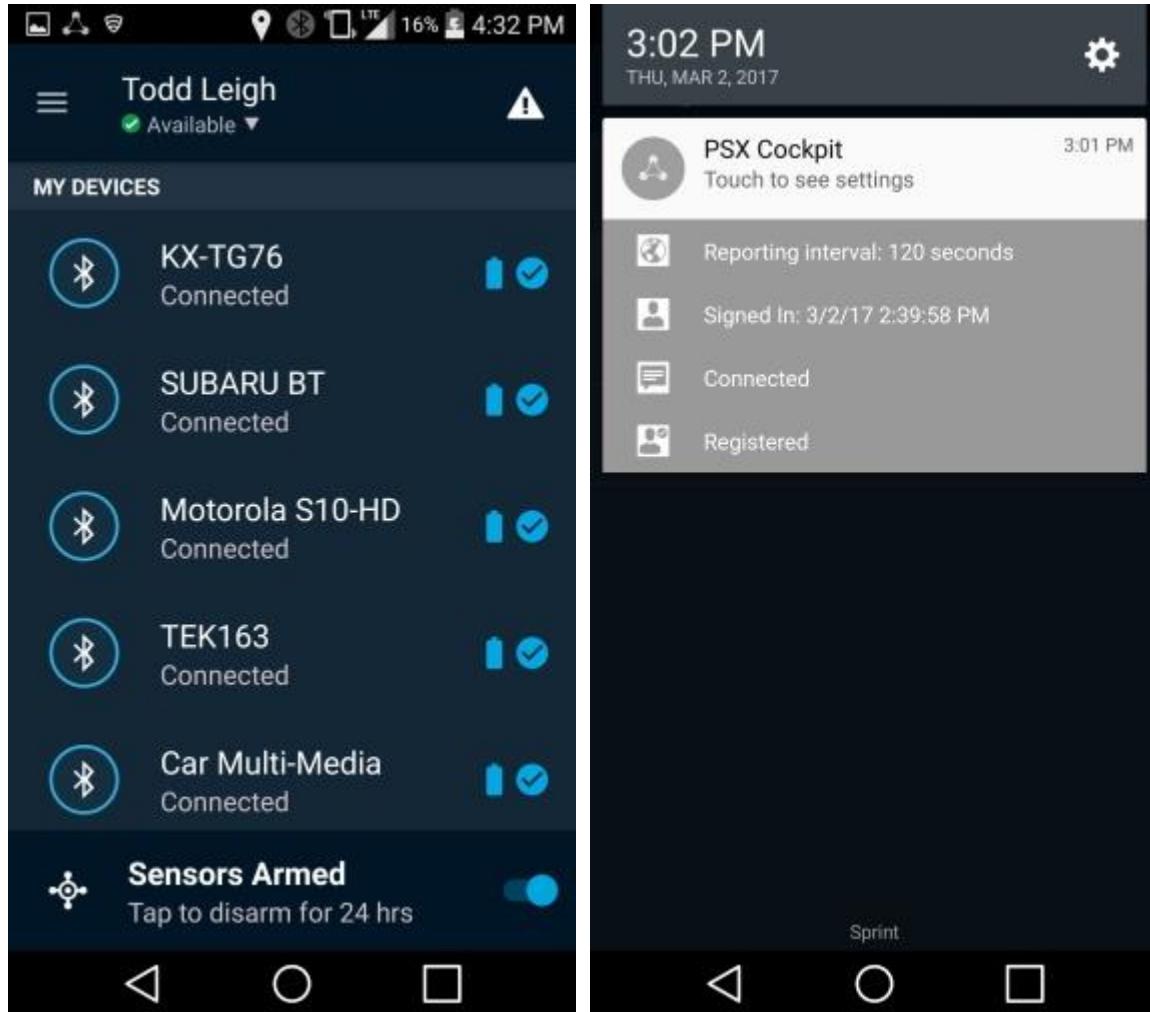


495

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

497 13. Tap the upper-right check mark to select the list. Your screen should look like Figure 2-10.

498 Figure 2-10 PSX Cockpit User Interface



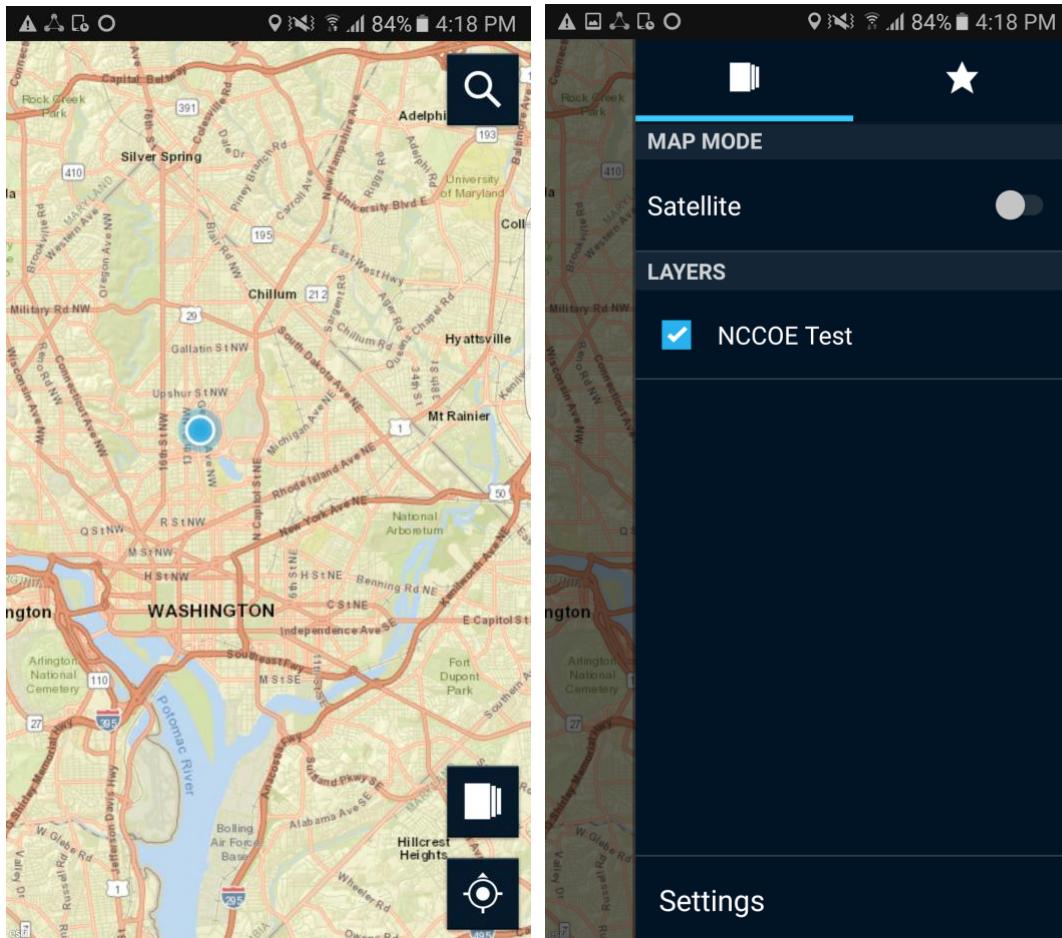
499

- 500 14. On the Cockpit screen, you can trigger an emergency (triangle icon in the upper right). Set your
501 status (drop-down menu under your name); or reselect roles and groups, see configuration, and
502 sign off (hamburger menu to the left of your name, and then tap **username**).
503 15. If you pull down your notifications, you should see icons and text indicating Reporting interval:
504 120 seconds, Signed In: <date> <time>, Connected, and Registered.

505 *2.2.1.2 Configuring the PSX Mapping Application*

506 1. Open the Mapping application. You should see the screen shown in Figure 2-11.

507 **Figure 2-11 PSX Mapping User Interface**

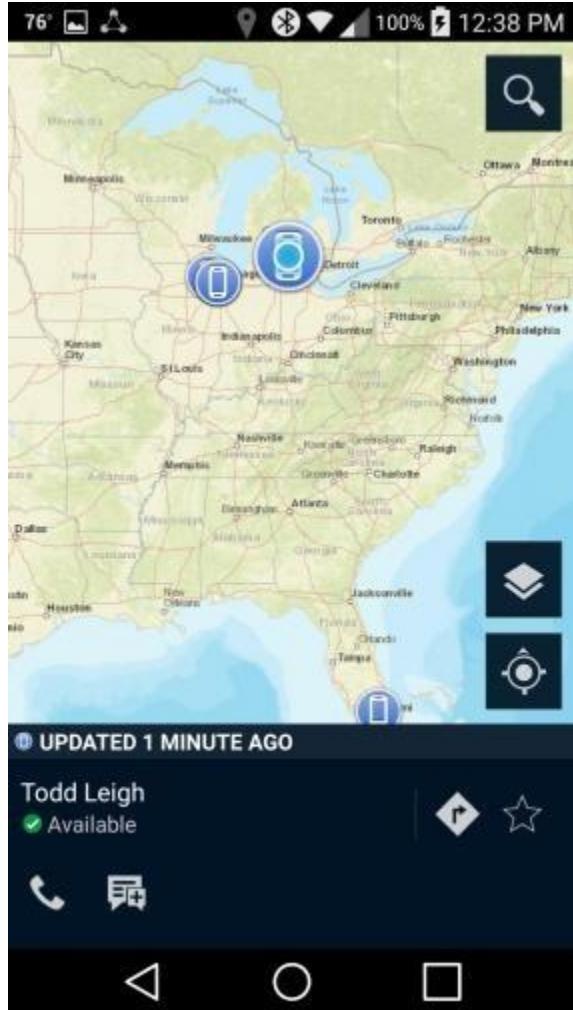


508

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

510 3. Select a group. Your screen should look like Figure 2-12.

511 Figure 2-12 PSX Mapping Group Member Information



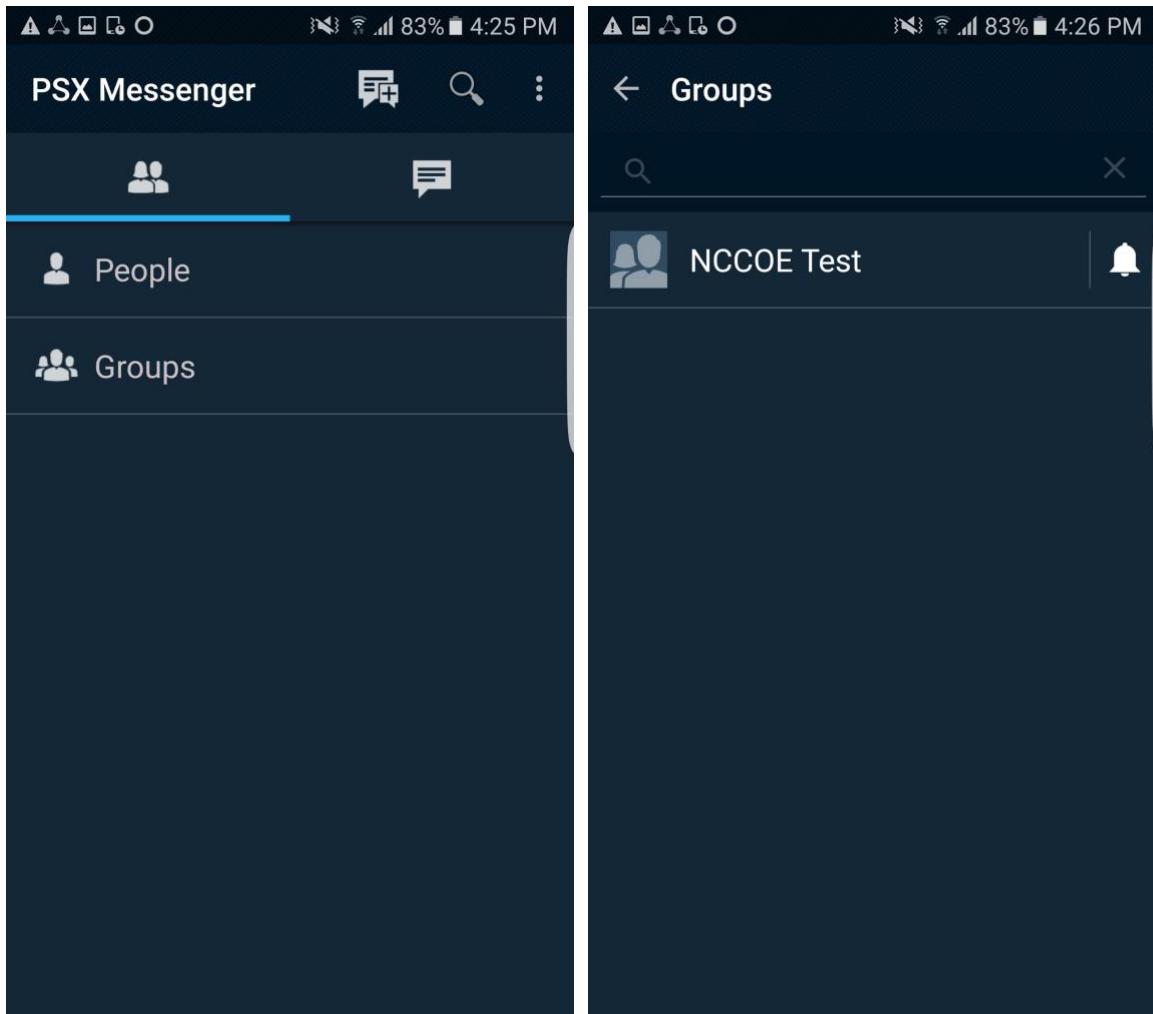
512

- 513 4. The locations of the devices that are members of that group should appear as dots on the map.
- 514 5. Select a device. A pop-up will show the user of the device, and icons for phoning and messaging that user.
- 515
- 516 6. Selecting the Messenger icon for the selected user will take you to the Messenger application, where you can send a message to the user.
- 517

518 *2.2.1.3 Configuring the PSX Messenger Application*

519 1. Open the Messenger application. Your screen should look like Figure 2-13.

520 **Figure 2-13 PSX Messenger User Interface**



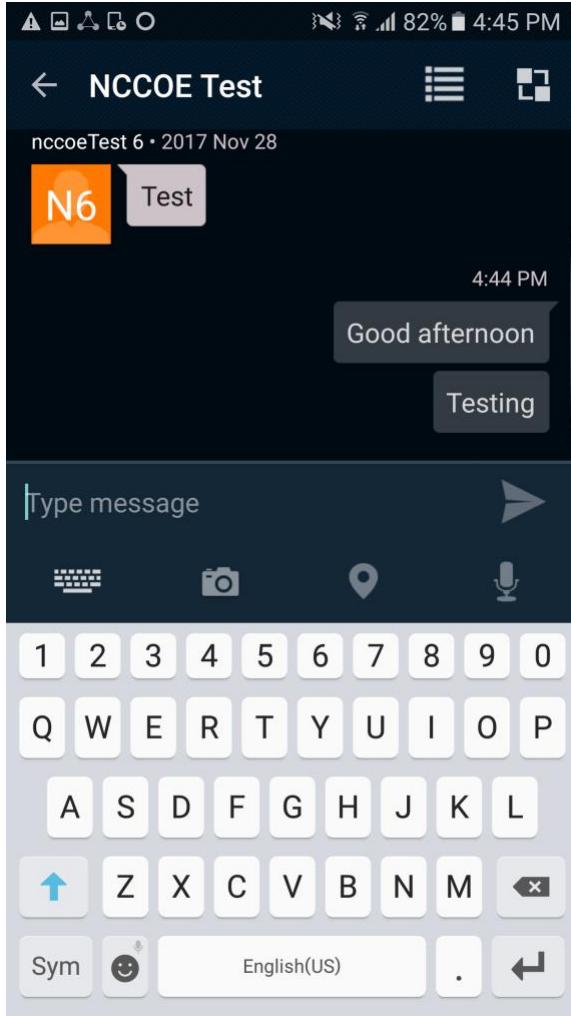
521

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

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

524 Your screen should look like Figure 2-14.

525 Figure 2-14 PSX Messenger Messages



526

- 527 4. You are now viewing the messaging window. You can type text for a message and attach a
528 picture, video, voice recording, or map.
- 529 5. Tap the Send icon. The message should appear on your screen.
- 530 6. Tap the Pivot icon in the upper-right corner of the message window. Select Locate, and you will
531 be taken to the Mapping application with the location of the people or group you selected.

532 **2.2.2 How to Install and Configure a FIDO U2F Authenticator**

533 This section covers the installation and usage of a FIDO U2F authenticator on an Android mobile device.
534 As explained in Section 2.1.4, the U2F login flow is not supported on iOS devices. The NCCoE reference
535 architecture utilizes the Google Authenticator application on the mobile device, and a Yubico YubiKey
536 NEO as a hardware token. The application provides an interface between the Chrome browser and the
537 U2F capabilities built into Play Services and is available on Google's Play Store [\[11\]](#).

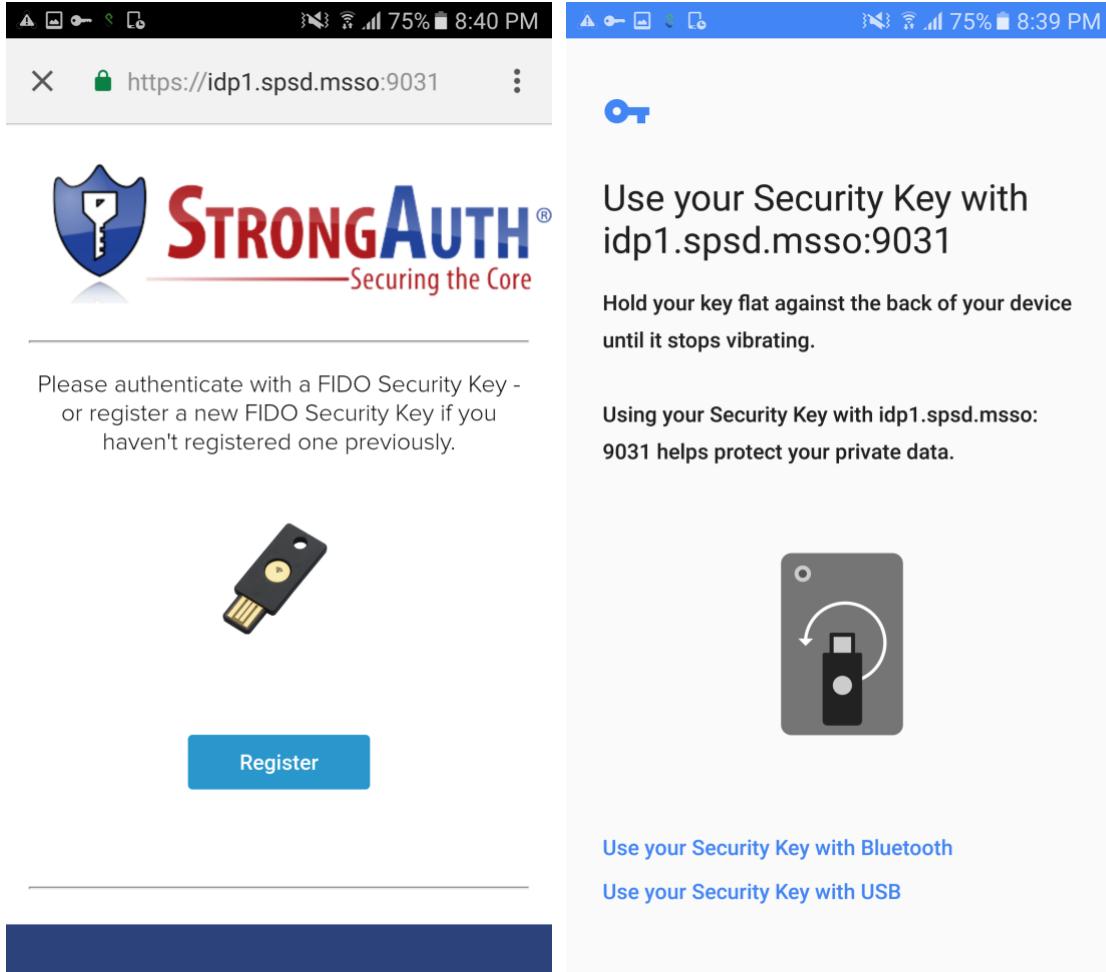
538 **2.2.2.1 Installing Google Authenticator**

- 539 1. On your Android device, open the Play Store application.
- 540 2. Search for Google Authenticator, and install the application. There is no configuration needed
541 until you are ready to register a FIDO U2F token with a StrongKey server.

542 **2.2.2.2 Registering the Token**

543 In the architecture that is laid out in this practice guide, there is no out-of-band process to register the
544 user's U2F token. This takes place the first time the user tries to log in with whatever SSO-enabled
545 application they are using. For instance, when using the PSX Cockpit application, once the user tries to
546 sign into an IdP that has U2F enabled and has successfully authenticated with a username and
547 password, they will be presented with the screen shown in Figure 2-15.

548 Figure 2-15 FIDO U2F Registration



549

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

- 551 1. Click **Register**, and the web page will activate the Google Authenticator application, which asks
552 you to use a U2F token to continue (Figure 2-15 above).
- 553 2. Hold the U2F token to your device, and then the token will be registered to your account and
554 you will be redirected to the U2F login screen again.

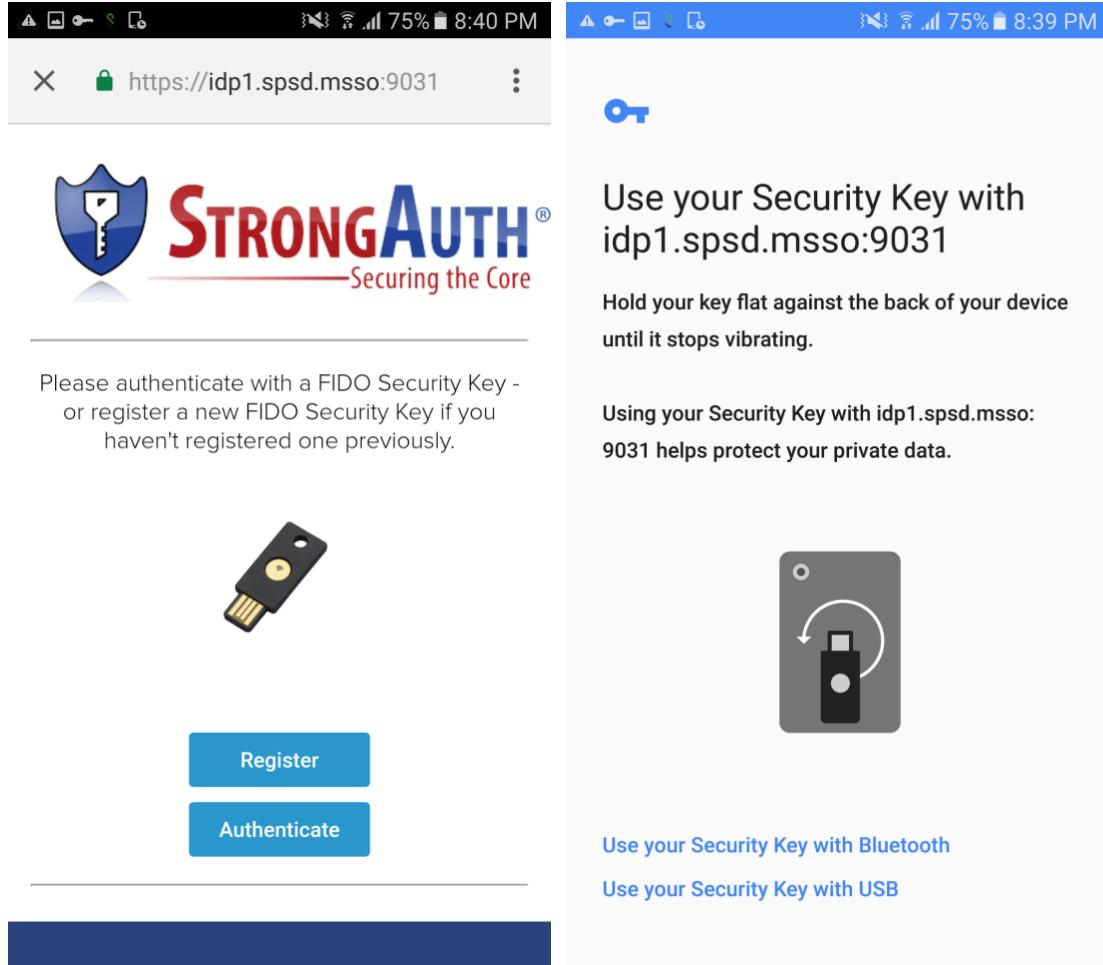
555 *2.2.2.3 Authenticating with the Token*

556 Now, because the system has a U2F token on file for the user, the user has the option to authenticate.

- 557 1. Click **Authenticate** (Figure 2-16), and the Google Authenticator application will be activated
558 once more.

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

561 **Figure 2-16 FIDO U2F Authentication**



562

563 **2.2.3 How to Install and Configure a FIDO UAF Client**

564 This section covers the installation and usage of a FIDO UAF client on the mobile device. Any FIDO UAF
565 client can be used, but the NCCoE reference architecture utilizes the Nok Nok Passport application
566 (hereafter referred to as “Passport”). The Passport application functions as the client-side UAF
567 application and is available on Google’s Play Store [\[16\]](#) and Apple’s App Store [\[17\]](#). The following excerpt
568 is from the Play Store page:

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

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

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

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

582 *2.2.3.1 Installing Passport on Android*

- 583 1. On your Android device, open the Play Store application.
- 584 2. Search for Nok Nok Passport, and install the application. There is no configuration needed until
585 you are ready to enroll the device with a Nok Nok Labs server.

586 Normally, the user will never need to open the Passport application during authentication; it will
587 automatically be invoked by the SSO-enabled application (e.g., PSX Cockpit). Instead of entering a
588 username and password into a Chrome Custom Tab, the user will be presented with the Passport screen
589 to use the user's UAF credential.

590 *2.2.3.2 Installing Passport on iOS*

- 591 1. On your iOS device, open the App Store application.
- 592 2. Search for Nok Nok Passport, and install the application. There is no configuration needed until
593 you are ready to enroll the device with a Nok Nok Labs server.

594 As with the Android application, the Passport application for iOS is invoked automatically during login
595 with a UAF-enabled server.

596 *2.2.3.3 Enrolling the Device*

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

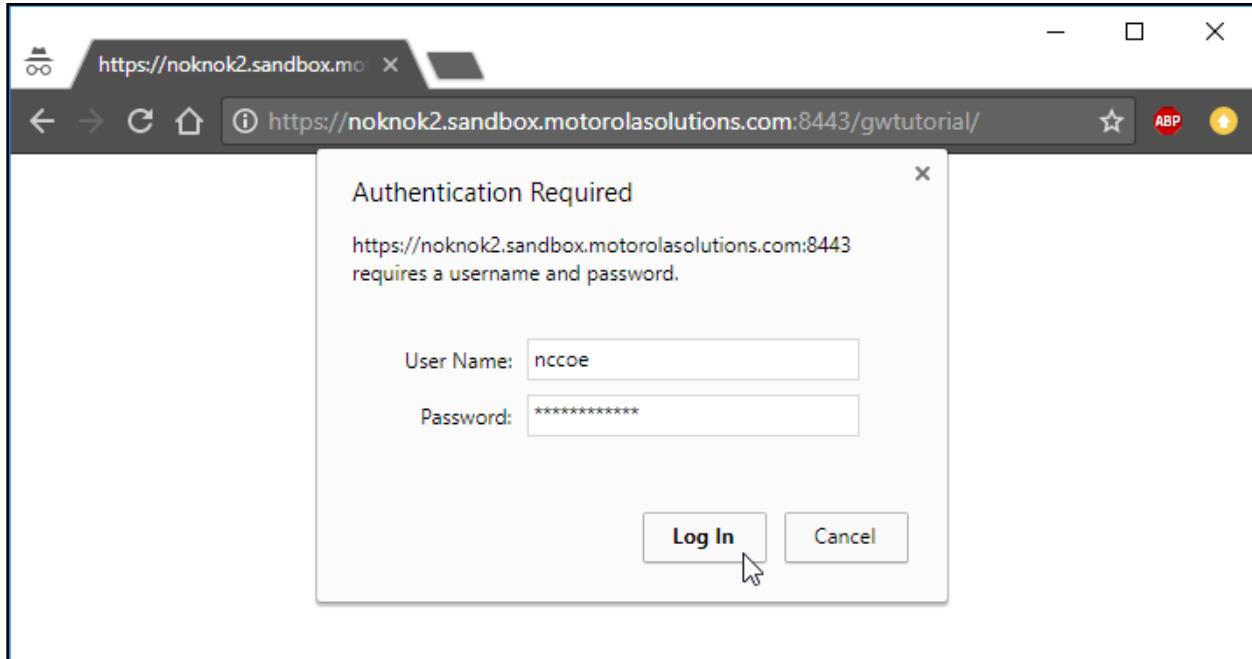
600 *Note: Users are not authenticated during registration. We are using the “tutorial” application provided
601 with the NNAS. This sample implementation does not meet the FIDO requirement of authentication prior*

602 to registration. The production version of the NNAS may require additional steps and may have a
603 different interface.

604 Screenshots that demonstrate the enrollment process are shown in Figure 2-17 through Figure 2-24.

605 1. First, use your computer to navigate to the NNAS web interface. You will be prompted for a
606 username and password; enter your administrator credentials and click **Log In** (Figure 2-17).

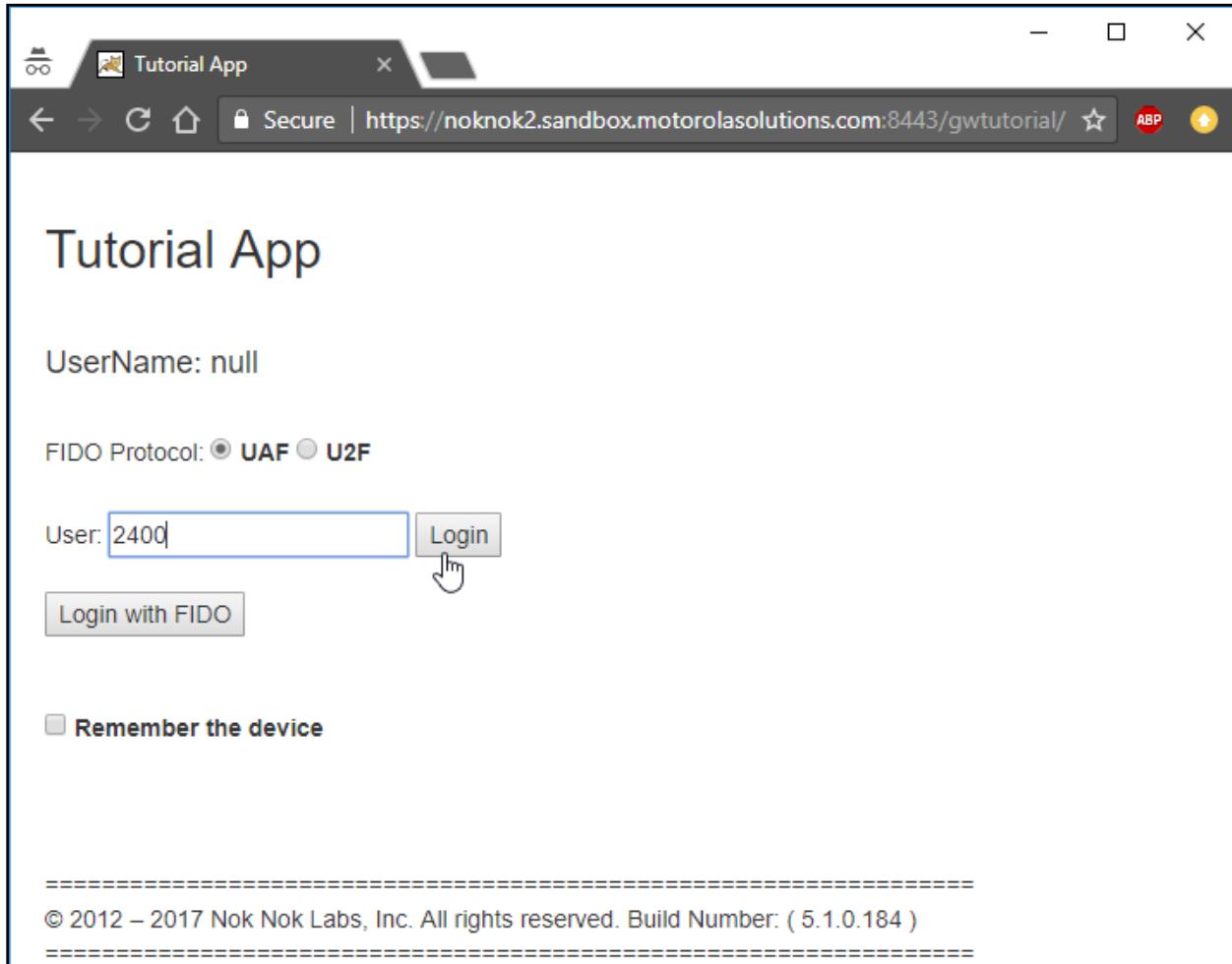
607 **Figure 2-17 Nok Nok Labs Tutorial Application Authentication**



608
609 2. Once you have logged in to the NNAS as an administrator, you need to identify which user you
610 want to manage. Enter the username and click **Login** (Figure 2-18).

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

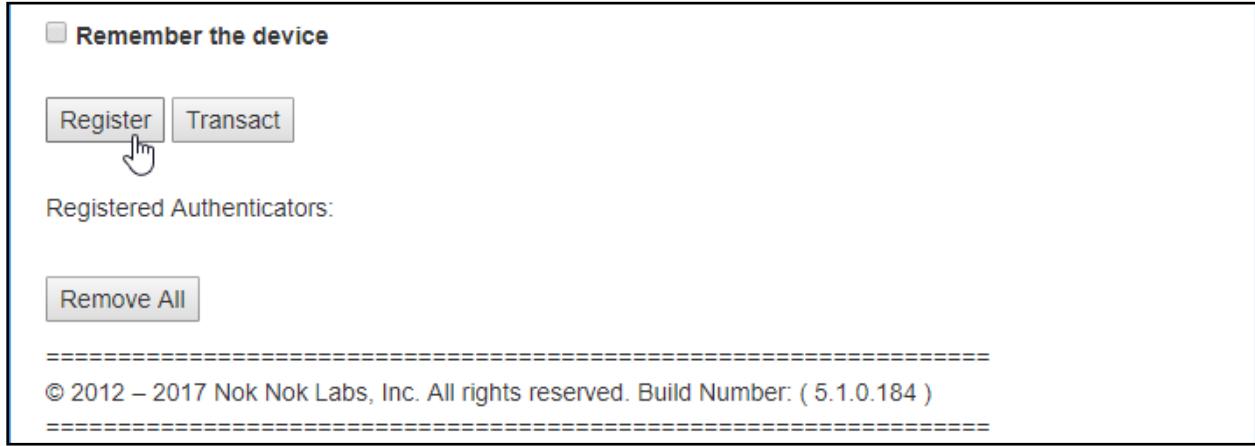
613 Figure 2-18 Nok Nok Labs Tutorial Application Login



614

- 615 3. Once you have selected the user, you will need to start the FIDO UAF registration process. To
616 begin, click **Register** (Figure 2-19).

617 **Figure 2-19 FIDO UAF Registration Interface**

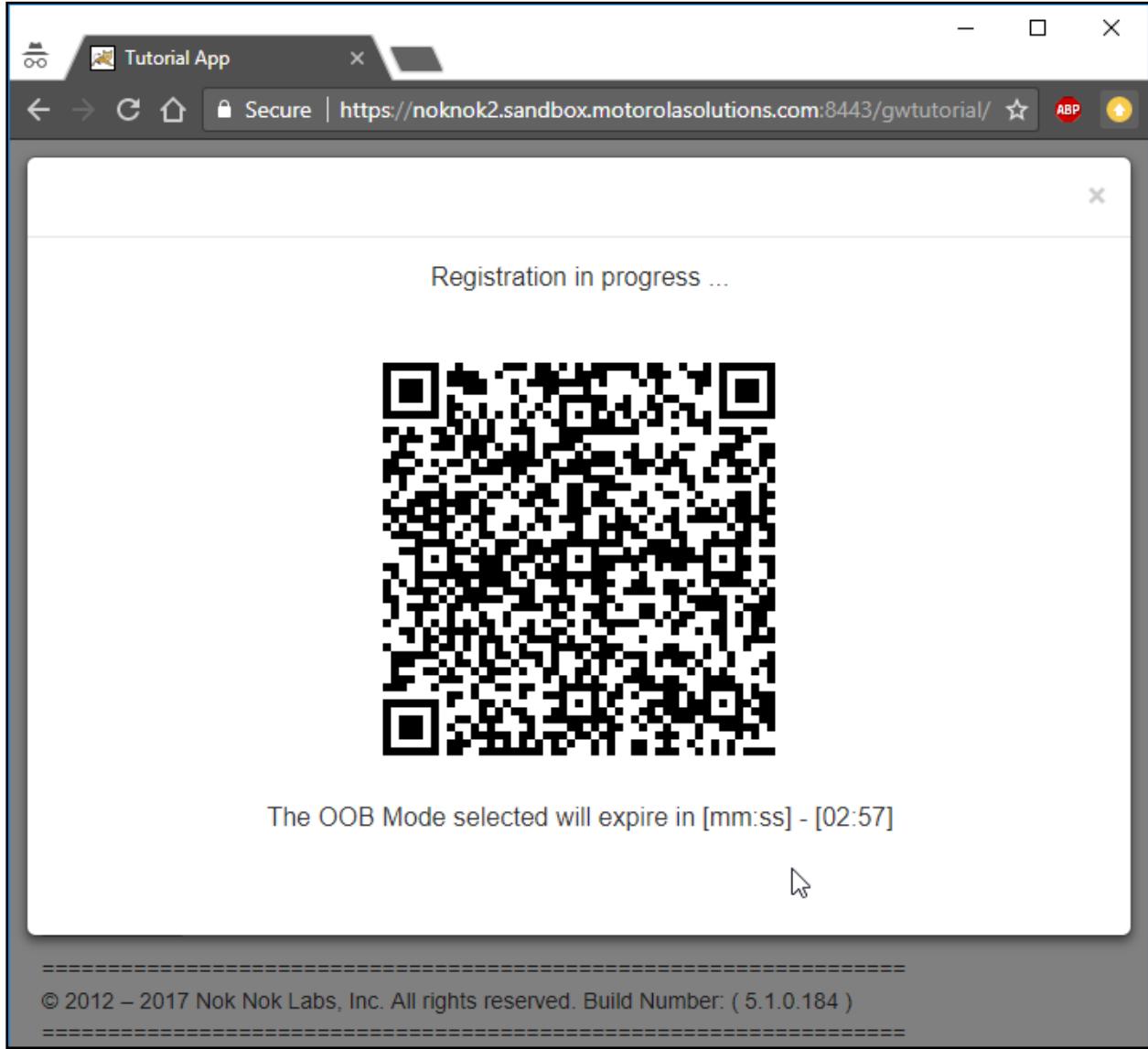


618

619 4. You will see a window with a QR code and a countdown (Figure 2-20). You have three minutes
620 to finish the registration process with your device.

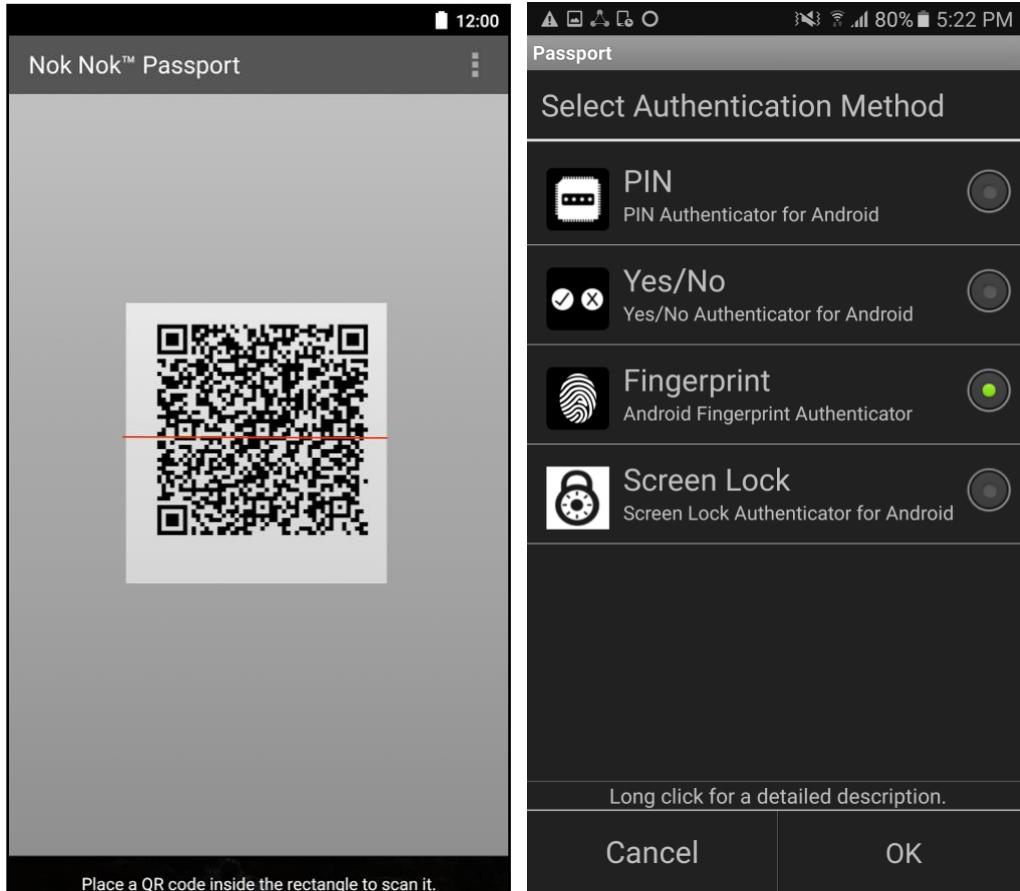
- 621 a. Once the QR image appears, launch the Passport application on the phone. The Passport
622 application activates the device camera to enable capturing the QR code by centering
623 the code in the square frame in the middle of the screen (Figure 2-21Figure 2-21).
- 624 b. Once the QR code is scanned, the application prompts the user to select the type of
625 verification (fingerprint, PIN, etc.) to use (Figure 2-21). The selections may vary based on
626 the authenticator modules installed on the device. Figure 2-21 shows the Passport
627 application on an Android device. Figure 2-22 shows the same flow on an iOS device. On
628 iOS devices that support Face ID, such as the iPhone X, Face ID is available as a user
629 verification option.

630 Figure 2-20 FIDO UAF Registration QR Code



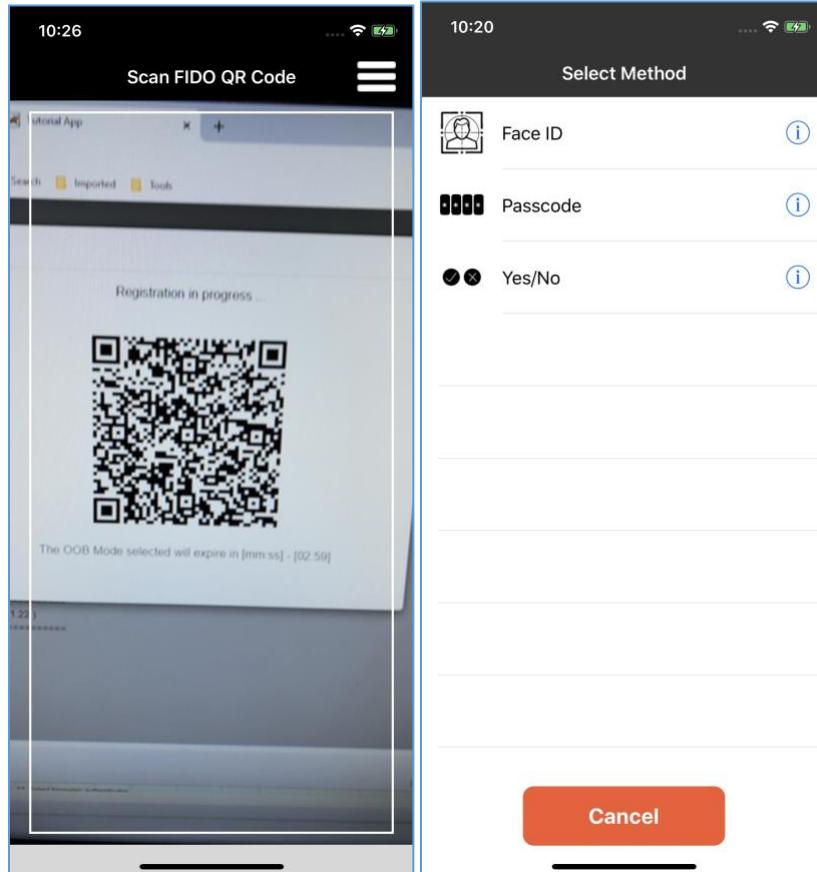
631

632 Figure 2-21 FIDO UAF Registration Device Flow, Android Device



633

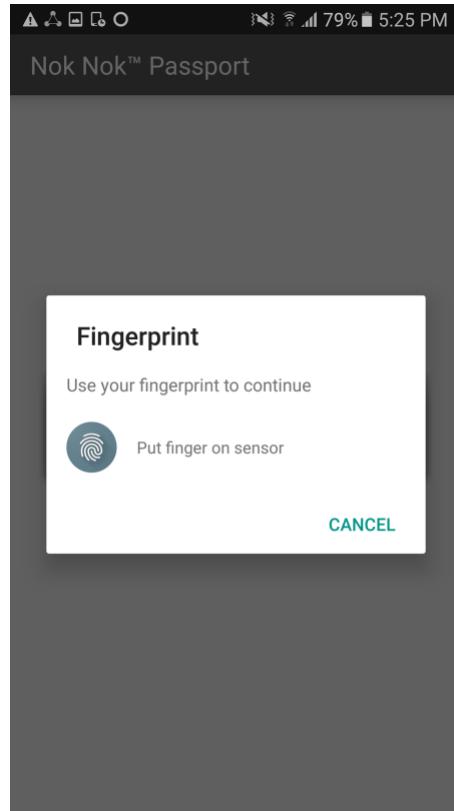
634 Figure 2-22 FIDO UAF Registration Device Flow, iPhone X



635

- 636 5. The user is then prompted to perform user verification with the selected method. In the
637 example shown in Figure 2-23, a fingerprint authenticator is registered. The user is prompted for
638 a fingerprint scan to complete registration. The fingerprint authenticator uses a fingerprint
639 previously registered in the Android screen-lock settings. If a PIN authenticator were registered,
640 the user would be prompted to set a PIN instead.

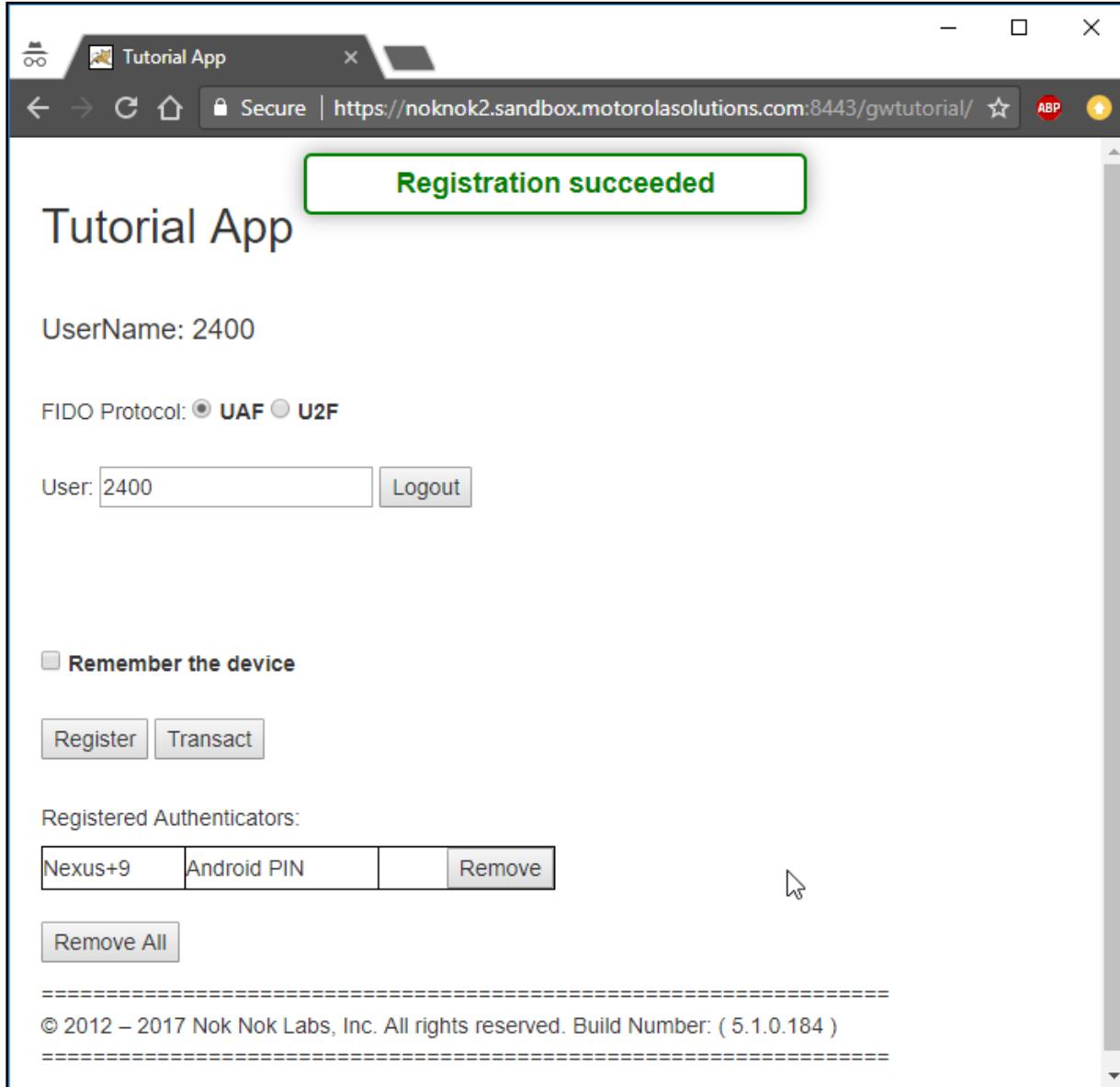
641 **Figure 2-23 FIDO UAF Fingerprint Authenticator, Android Device**



642

- 643 6. If user verification is successful, then a new UAF key pair is generated, the public key is sent to
644 the server, and registration is completed (Figure 2-24).

645 Figure 2-24 FIDO UAF Registration Success



646

647

2.3 How Application Developers Must Integrate AppAuth for SSO

648 Application developers can easily integrate AppAuth to add SSO capabilities to their applications. The
649 first step to doing this is reading through the documentation on GitHub for AppAuth for Android [19] or
650 iOS [8]. After doing so, an application developer can begin the integration of AppAuth. The degree of
651 this integration can vary—for instance, you may choose to utilize user attributes to personalize the

652 user's application experience. The following sections describe AppAuth integration for Android and iOS
653 applications.

654 For either platform, the mobile application must be registered with the OAuth AS and given a client ID as
655 described in Section 3.3. The client ID will be needed when building the mobile application.

656 **2.3.1 AppAuth Integration for Android**

657 In this example, we use Android Studio 3.0, Android Software Development Kit 25, and Gradle 2.14.1.

658 **2.3.1.1 Adding the Library Dependency**

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

```
662 =====  
663 dependencies {  
664     ...  
665     compile 'net.openid:appauth:0.7.0'  
666 }  
667 =====
```

668 **2.3.1.2 Adding Activities to the Manifest**

669 1. First, you need to identify your AS's host name, OAuth redirect path, and what scheme was set
670 when you registered your application. The scheme here is contrived, but it is common practice
671 to use reverse DNS style names; you should choose whatever aligns with your organization's
672 common practices. Another alternative to custom schemes is to use App Links.

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

```
674 =====  
675 <manifest xmlns:android="http://schemas.android.com/apk/res/android"  
676     xmlns:tools="http://schemas.android.com/tools"  
677     package="com.example.app">  
678     ...  
679         <activity  
680             android:name="net.openid.appauth.RedirectUriReceiverActivity"  
681             tools:node="replace">  
682             <intent-filter>
```

```

683         <action android:name="android.intent.action.VIEW" />
684         <category android:name="android.intent.category.DEFAULT" />
685         <category android:name="android.intent.category.BROWSABLE" />
686     <data
687         android:host="as.example.com"
688         android:path="/oauth2redirect"
689         android:scheme="myappsscheme" />
690     </intent-filter>
691   </activity>
692   <activity android:name=".activity.AuthResultHandlerActivity" />
693   <activity android:name=".activity.AuthCanceledHandlerActivity" />
694 </application>
695 </manifest>
696 =====

```

2.3.1.3 *Creating 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**):

```

701 =====
702 public class Utility {
703     public static AuthorizationService getAuthorizationService(Context context) {
704
705         AppAuthConfiguration appAuthConfig = new AppAuthConfiguration.Builder()
706             .setBrowserMatcher(new BrowserWhitelist(
707                 VersionedBrowserMatcher.CHROME_CUSTOM_TAB,
708                 VersionedBrowserMatcher.SAMSUNG_CUSTOM_TAB))
709
710         // the browser matcher above allows you to choose which in-app
711         browser
712
713         // tab providers will be supported by your app in its OAuth2 flow
714         .setConnectionBuilder(new ConnectionBuilder() {
715
716             @NonNull

```

```
714         public HttpURLConnection openConnection(@NonNull Uri uri)
715             throws IOException {
716
717             URL url = new URL(uri.toString());
718             HttpURLConnection connection =
719                 (HttpURLConnection) url.openConnection();
720
721             if (connection instanceofHttpsURLConnection) {
722
723                 // optional: use your own trust manager to set a custom
724                 // SSLSocketFactory on the HttpsURLConnection
725
726             }
727
728             return connection;
729         }
730
731         return new AuthorizationService(context, appAuthConfig);
732     }
733
734     public static AuthState restoreAuthState(Context context) {
735
736         // we use SharedPreferences to store a String version of the JSON
737         // Auth State, and here we retrieve it to convert it back to a POJO
738
739         SharedPreferences sharedPreferences =
740             PreferenceManager.getDefaultSharedPreferences(context);
741
742         String jsonString = sharedPreferences.getString("AUTHSTATE", null);
743
744         if (!TextUtils.isEmpty(jsonString)) {
745
746             try {
747
748                 return AuthState.jsonDeserialize(jsonString);
749             } catch (JSONException jsonException) {
750
751                 // handle this appropriately
752             }
753
754         }
755
756         return null;
757     }
758 }
```

```
744         }
745     }
746 =====
747     public class AuthResultHandlerActivity extends Activity {
748
749         private static final String TAG = AuthResultHandlerActivity.class.getName();
750
751         private AuthState mAuthState;
752
753         private AuthorizationService mAuthService;
754
755         @Override
756         protected void onCreate(Bundle savedInstanceState) {
757             super.onCreate(savedInstanceState);
758
759             AuthorizationResponse res =
760             AuthorizationResponse.fromIntent(getIntent());
761
762             AuthorizationException ex =
763             AuthorizationException.fromIntent(getIntent());
764
765             mAuthState = new AuthState(res, ex);
766
767             mAuthService = Utility.getAuthorizationService(this);
768
769             if (res != null) {
770                 Log.d(TAG, "Received AuthorizationResponse");
771                 performTokenRequest(res.createTokenExchangeRequest());
772             } else {
773                 Log.d(TAG, "Authorization failed: " + ex);
774             }
775         }
776
777         @Override
778         protected void onDestroy() {
```

```
775         super.onDestroy();
776
777         mAuthService.dispose();
778     }
779
780     private void performTokenRequest(TokenRequest request) {
781
782         TokenResponseCallback callback = new TokenResponseCallback() {
783
784             @Override
785
786             public void onTokenRequestCompleted(
787                 TokenResponse tokenResponse,
788                 AuthorizationException authException) {
789
790                 receivedTokenResponse(tokenResponse, authException);
791
792             }
793
794             mAuthService.performTokenRequest(request, callback);
795
796         }
797
798         private void receivedTokenResponse(TokenResponse tokenResponse,
799
800                 AuthorizationException authException) {
801
802             Log.d(TAG, "Token request complete");
803
804             if (tokenResponse != null) {
805
806                 mAuthState.update(tokenResponse, authException);
807
808                 // persist auth state to SharedPreferences
809                 PreferenceManager.getDefaultSharedPreferences(this)
810
811                     .edit()
812
813                     .putString("AUTHSTATE", mAuthState.jsonSerializeString())
814
815                     .commit();
816
817
818             String accessToken = mAuthState.getAccessToken();
819
820             if (accessToken != null) {
```

```
805             // optional: pull claims out of JWT (name, etc.)
806         }
807     } else {
808         Log.d(TAG, " ", authException);
809     }
810 }
811 =====
812 public class AuthCanceledHandlerActivity extends Activity {
813
814
815     private static final String TAG =
816     AuthCanceledHandlerActivity.class.getName();
817
818     @Override
819     protected void onCreate(Bundle savedInstanceState) {
820         super.onCreate(savedInstanceState);
821
822         Log.d(TAG, "OpenID Connect authorization flow canceled");
823
824         // go back to MainActivity
825         finish();
826     }
827 }
828 =====
829 2.3.1.4 Executing the OAuth 2 Authorization Flow
830 1. In whatever activity you are using to initiate authentication, add the necessary code to use the
831     AppAuth SDK to execute the OAuth 2 authorization flow:
832 =====
833 ...
834
835 // some method, usually a "login" button, activates the OAuth2 flow
836
```

```

837     String OAUTH_AUTH_ENDPOINT =
838         "https://as.example.com:9031/as/authorization.oauth2";
839     String OAUTH_TOKEN_ENDPOINT = "https://as.example.com:9031/as/token.oauth2";
840     String OAUTH_REDIRECT_URI = "myappscheme://app.example.com/oauth2redirect";
841     String OAUTH_CLIENT_ID = "myapp";
842     String OAUTH_PKCE_CHALLENGE_METHOD = "S256"; // options are "S256" and "plain"
843
844     // CREATE THE SERVICE CONFIGURATION
845     AuthorizationServiceConfiguration config = new
846     AuthorizationServiceConfiguration(
847         Uri.parse(OAUTH_AUTH_ENDPOINT), // auth endpoint
848         Uri.parse(OAUTH_TOKEN_ENDPOINT), // token endpoint
849         null // registration endpoint
850     );
851
852     // OPTIONAL: Add any additional parameters to the authorization request
853     HashMap<String, String> additionalParams = new HashMap<>();
854     additionalParams.put("acr_values", "urn:acr:form");
855
856     // BUILD THE AUTHORIZATION REQUEST
857     AuthorizationRequest.Builder builder = new AuthorizationRequest.Builder(
858         config,
859         OAUTH_CLIENT_ID,
860         ResponseTypeValues.CODE,
861         Uri.parse(OAUTH_REDIRECT_URI))
862         .setScopes("profile") // scope is optional, set whatever is needed by
863         your app
864         .setAdditionalParameters(additionalParams);
865
866     // SET UP PKCE CODE VERIFIER
867     String codeVerifier = CodeVerifierUtil.generateRandomCodeVerifier();
868     String codeVerifierChallenge =
869     CodeVerifierUtil.deriveCodeVerifierChallenge(codeVerifier);
870     builder.setCodeVerifier(codeVerifier, codeVerifierChallenge,
871
872         OAUTH_PKCE_CHALLENGE_METHOD);
873
874     AuthorizationRequest request = builder.build();
875
876     // PERFORM THE AUTHORIZATION REQUEST
877     // this pauses and leaves the current activity
878     Intent postAuthIntent = new Intent(this, AuthResultHandlerActivity.class);
879     Intent authCanceledIntent = new Intent(this,
880     AuthCanceledHandlerActivity.class);
881     mAuthService.performAuthorizationRequest(
882         request,
883         PendingIntent.getActivity(this, request.hashCode(), postAuthIntent, 0),
884         PendingIntent.getActivity(this, request.hashCode(), authCanceledIntent,
885         0));
886
887     ...

```

```

888     // when the activity resumes, check if the OAuth2 flow was successful
889
890     @Override
891     protected void onResume() {
892         super.onResume();
893
894         AuthState authState = Utility.restoreAuthState(this);
895         if (authState != null) {
896
897             // we are authorized!
898             // proceed to the next activity that requires an access token
899         }
900
901         ...
902 =====

```

[2.3.1.5 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:

```

908 =====
909 ...
910
911 // assuming we have an instance of a Context as mContext...
912
913 // ensure we have a fresh access token to perform any future actions
914 final AuthorizationService authService =
915     Utility.getAuthorizationService(mContext);
916 AuthState authState = Utility.restoreAuthState(mContext);
917 authState.performActionWithFreshTokens(authService, new
918     AuthState.AuthStateAction() {
919         @Override
920         public void execute(String accessToken, String idToken,
921                             AuthorizationException ex) {
922             JWT jwt = null;
923             if (ex != null) {
924                 // negotiation for fresh tokens failed, check ex for more details
925             } else {
926                 // we can now use accessToken to access remote services
927
928                 // this is typically done by including the token in an HTTP header,
929                 // or in a handshake transaction if another transport protocol is
930                 used

```

```
929         }
930         authService.dispose();
931     }
932 });
933
934 ...
935 =====
```

936 2.3.2 AppAuth Integration for iOS

937 The iOS demo applications were built with XCode 10.1 for iOS deployment target 11.0. using the Swift
938 programming language.

939 2.3.2.1 Adding the Library Dependency

940 The AppAuth library can be added to an XCode project by using either the CocoaPods or Carthage
941 dependency manager. The CocoaPods method automatically uses the official released version of the
942 library. To use a particular code branch or to get recent updates not available in the release version,
943 Carthage must be used. The official release should be suitable for the majority of applications.

944 To add the AppAuth library by using CocoaPods:

- 945 1. Create a Podfile in the root directory of the project. The following is a sample Podfile from the
946 maps-demo application that adds AppAuth and two other libraries.

```
947 =====
948 source 'https://github.com/CocoaPods/Specs.git'
949 target 'map-demo-app-ios' do
950   pod 'GoogleMaps'
951   pod 'GooglePlaces'
952   pod 'AppAuth'
953 end
954 =====
```

- 955 2. Open a terminal and navigate to the root directory of the project and run the command:

```
956 pod install
```

- 957 3. In XCode, close any open projects. Click File–Open, navigate to the root of the project, and open
958 the file <project-name>.xcworkspace.

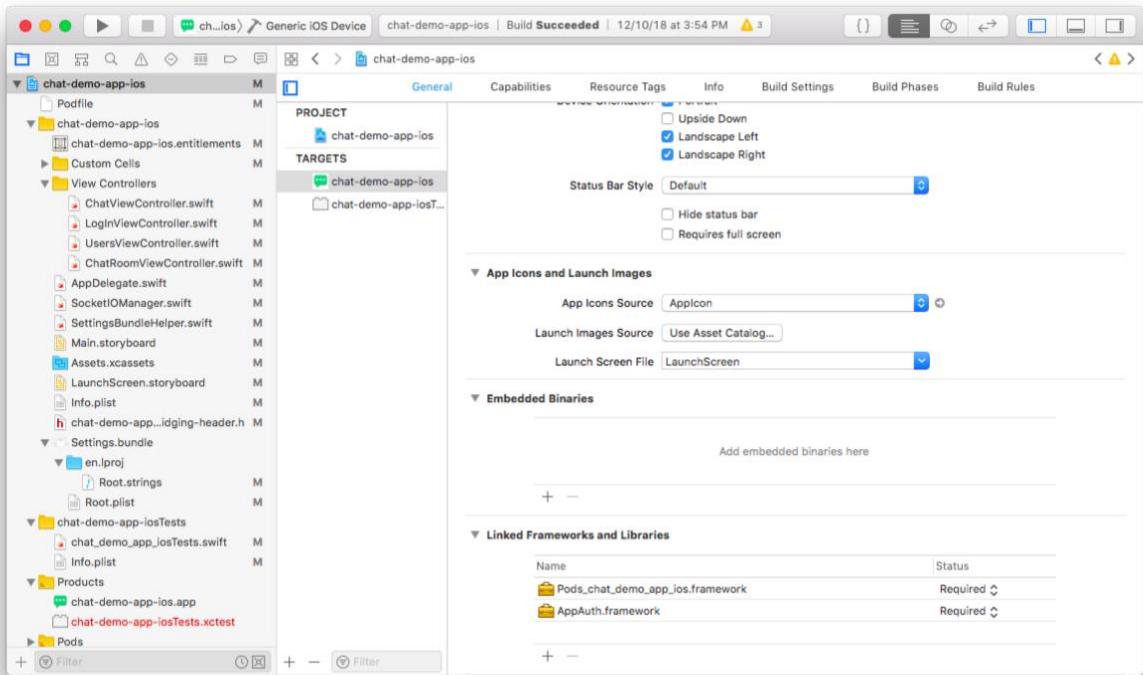
959 To add the AppAuth library by using Carthage:

- 960 1. Create a Cartfile with the following contents in the root directory of the project:

```
961 =====
962 github "openid/AppAuth-iOS" "master"
963 =====
```

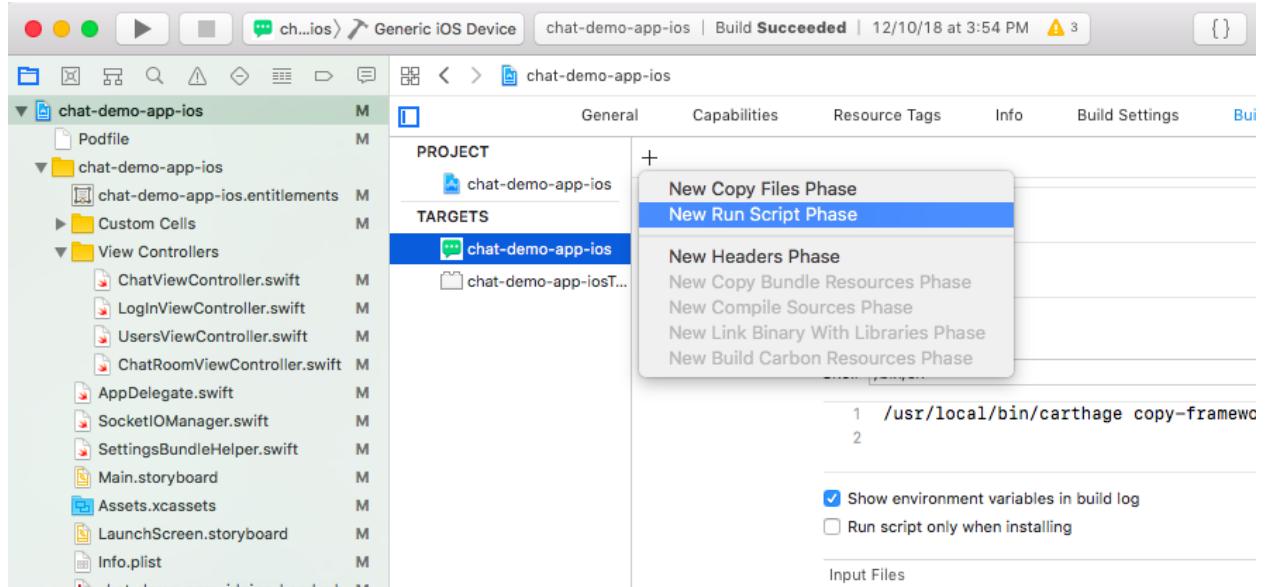
- 964 2. Open a terminal and navigate to the root directory of the project and run the command:
- 965 **carthage bootstrap**
- 966 3. In XCode, click on the project in the project navigator and select the General tab. Under Linked Frameworks and Libraries, click the plus icon to add a framework.
- 967
- 968 4. Click Add Other.... A file selection dialogue should open and display the root folder of the project. Navigate to the Carthage/Build/iOS subfolder, select AppAuth.framework, and click Open. The Frameworks and Libraries interface is shown in Figure 2-25.
- 969
- 970

971 **Figure 2-25 Linked Frameworks and Libraries**



- 972
- 973 5. On the Build Phases tab, click the plus icon in the top left corner of the editor and select New Run Script Phase as shown in Figure 2-26.
- 974

975 Figure 2-26 Creating a New Run Script Phase



976

- 977 6. Add the following command to the Run Script:

```
/usr/local/bin/carthage copy-frameworks
```

- 979 7. Click the plus icon under Input Files and add the following entry:

```
$(SRCROOT)/Carthage/Build/iOS/AppAuth.framework
```

981 Figure 2-27 shows a completed Run Script.

982 Figure 2-27 Carthage Run Script



983

984 Once either of the above procedures is completed, you should be able to import AppAuth into your
985 project without compiler errors.

986 *2.3.2.2 Registering a Custom URL Scheme*

987 To enable the AS to send a redirect through the browser back to your mobile application, you must
988 either register a custom URL scheme or use Universal Links. This example shows the use of a custom URL
989 scheme. This scheme must be included in the redirect_uri registered with the AS; see Section 3.3 for
990 details on OAuth client registration. To configure the custom URL scheme:

- 991 1. In the XCode Project Navigator, select the Info.plist file.
992 2. Select “URL Types” and click the Plus icon to add a type.
993 3. Under the created item, click on the selector icon and choose “URL Schemes.”
994 4. Edit the item value to match the URL scheme. Figure 2-28 shows a custom URL scheme of
995 “org.mitre.chatdemo.”

996 Figure 2-28 Custom URL Scheme

Key	Type	Value
▼ Information Property List	Dictionary	(18 items)
Localization native development re...	String	\$(DEVELOPMENT_LANGUAGE)
Bundle display name	String	chat-demo
Executable file	String	\$(EXECUTABLE_NAME)
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
▼ URL types	Array	(1 item)
▼ Item 0 (Editor)	Dictionary	(2 items)
Document Role	String	Editor
▼ URL Schemes	Array	(1 item)
Item 0	String	org.mitre.chatdemo
Bundle version	String	1
► LSApplicationQueriesSchemes	Array	(3 items)
Application requires iPhone environ...	Boolean	YES
► App Transport Security Settings	Dictionary	(1 item)
Launch screen interface file base...	String	LaunchScreen
Main storyboard file base name	String	Main
► Required device capabilities	Array	(1 item)
► Supported interface orientations	Array	(3 items)
► Supported interface orientations (i...)	Array	(4 items)

997

998

2.3.2.3 Handling Authorization Responses

999 Add the following lines to AppDelegate.swift to handle authorization responses submitted to your
1000 application's redirect_uri:

```

1001 =====
1002 var currentAuthorizationFlow:OIDAuthorizationFlowSession?
1003 func application(_ app: UIApplication, open url: URL, options:
1004 [UIApplicationOpenURLOptionsKey : Any] = [:]) -> Bool {
1005     if let authorizationFlow = self.currentAuthorizationFlow,
1006     authorizationFlow.resumeAuthorizationFlow(with: url) {
1007         self.currentAuthorizationFlow = nil
1008         return true
1009     }
1010     return false
1011 }
```

1012 =====

1013 ***2.3.2.4 Executing the OAuth 2 Authorization Flow***

1014 In the View Controller that handles authentication events, add the necessary code to use AppAuth to
 1015 submit authorization requests to the AS. The configuration parameters for the AS, such as the URLs for
 1016 the authorization and token endpoints, can be automatically discovered if the AS supports OpenID
 1017 Connect Discovery; otherwise these parameters must be provided either in settings or in the code. In
 1018 this example, they are specified in the code. This example also demonstrates how to specify the user-
 1019 agent for the authorization flow; in this case, Safari will be used.

1020 =====

```

1021 class LogInViewController: UIViewController, OIDAuthStateChangeDelegate,
1022 OIDAuthStateErrorDelegate {
1023     let kAppAuthExampleAuthStateKey = authState";
1024
1025     ...
1026
1027     func authenticateUsingLab() {
1028         var configuration: OIDServiceConfiguration =
1029             OIDServiceConfiguration(authorizationEndpoint: URL(string:
1030             "https://as1.cpssp.msso:9031/as/authorization.oauth2")!, tokenEndpoint: URL(string:
1031             "https://as1.cpssp.msso:9031/as/token.oauth2")!)
1032
1033         guard let redirectURI = URL(string:
1034             "org.mitre.chatdemo:/msso.nccoe.nist/oauth2redirect") else {
1035             print("Error creating URL for :
1036             org.mitre.chatdemo:/msso.nccoe.nist/oauth2redirect")
1037             return
1038         }
1039
1040         guard let appDelegate = UIApplication.shared.delegate as? AppDelegate else {
1041             print("Error accessing AppDelegate")
1042             return
1043         }
1044
1045         // builds authentication request
1046         let request = OIDAuthorizationRequest(configuration: configuration,
1047             clientId: "chatdemo",
1048             clientSecret: nil,
1049             scopes: ["testScope"],
1050             redirectURL: redirectURI,
1051             responseType: OIDResponseTypeCode,
1052             additionalParameters: nil)
1053
1054         print("Initiating authorization request with scope: \(request.scope ??
1055             "DEFAULT_SCOPE")")
1056
1057         doAuthWithAutoCodeExchange(configuration: configuration, request: request,
1058             appDelegate: appDelegate)

```

```
1059     }
1060
1061     func doAuthWithAutoCodeExchange(configuration: OIDServiceConfiguration, request:
1062 OIDAuthorizationRequest, appDelegate: AppDelegate) {
1063
1064         let coordinator: OIDAuthorizationUICoordinatorCustomBrowser =
1065 OIDAuthorizationUICoordinatorCustomBrowser.customBrowserSafari()
1066
1067         appDelegate.currentAuthorizationFlow = OIDAuthState.authState(byPresenting:
1068 request, uiCoordinator: coordinator) { authState, error in
1069             if let authState = authState {
1070                 self.assignAuthState(authState: authState)
1071                 self.segueToChat()
1072             } else {
1073                 print("Authorization error: \(error?.localizedDescription ??
1074 "DEFAULT_ERROR")")
1075                 self.assignAuthState(authState: nil)
1076             }
1077         }
1078     func saveState(){
1079         // for production usage consider using the OS Keychain instead
1080         if authState != nil{
1081             let archivedAuthState = NSKeyedArchiver.archivedData(withRootObject:
1082 authState!)
1083             UserDefaults.standard.set(archivedAuthState, forKey:
1084 kAppAuthExampleAuthStateKey)
1085         }
1086         else{
1087             UserDefaults.standard.set(nil, forKey: kAppAuthExampleAuthStateKey)
1088         }
1089         UserDefaults.standard.synchronize()
1090     }
1091
1092     func loadState(){
1093         // loads OIDAuthState from NSUserDefaults
1094         guard let archivedAuthState = UserDefaults.standard.object(forKey:
1095 kAppAuthExampleAuthStateKey) as? NSData else{
1096             return
1097         }
1098         guard let authState = NSKeyedUnarchiver.unarchiveObject(with: archivedAuthState
1099 as Data) as? OIDAuthState else{
1100             return
1101         }
1102         assignAuthState(authState: authState)
1103     }
1104
1105     func assignAuthState(authState:OIDAuthState?) {
1106         if (self.authState == authState) {
1107             return;
1108         }
1109         self.authState = authState
1110         self.authState?.stateChangeDelegate = self
```

```

1111         self.saveState()
1112     }
1113
1114     func didChange(_ state: OIDAuthState) {
1115         authState = state
1116         authState?.stateChangeDelegate = self
1117         self.saveState()
1118     }
1119
1120     func authState(_ state: OIDAuthState, didEncounterAuthorizationError error: Error)
1121     {
1122         print("Received authorization error: \(error)")
1123     }
1124 }
1125 =====

```

2.3.2.5 Fetching and Using the Access Token

1126 The access token can be retrieved from the authState object. If the access token has expired, the
 1127 application may need to use a refresh token to obtain a new access token or initiate a new authorization
 1128 request if it does not have an active refresh token. Access tokens are typically used in accordance with
 1129 RFC 6750 [20], most commonly in the Authorization header of a Hypertext Transfer Protocol (HTTP)
 1130 request to an API server. The following example shows a simple usage of an access token to call an API:

```

1132 =====
1133 public func requestChatRooms() {
1134     let urlString = "\(protocolIdentifier)://\(ipAddress):\(port)/getChatRooms"
1135     print("URLString \(urlString)")
1136     guard let url = URL(string: urlString) else { return }
1137     let token: String? = self.authState?.lastTokenResponse?.accessToken
1138     var request = URLRequest(url: url)
1139     request.httpMethod = "GET"
1140     request.setValue("Bearer \(token)", forHTTPHeaderField: "Authorization")
1141     URLSession.shared.dataTask(with: request) { (data, response, error) in
1142         if error != nil {
1143             print(error!.localizedDescription)
1144         }
1145         else {
1146             guard let data = data else { return }
1147             let json = try? JSONSerialization.jsonObject(with: data, options: [])
1148
1149             if let array = json as? [Any] {
1150                 if let firstObject = array.first {
1151                     if let dictionary = firstObject as? [String: String] {
1152                         self.chatRooms = dictionary
1153                         self.loadRooms()
1154                     }
1155                 }
1156             }
1157         }
1158     }.resume()

```

1159 }
1160 ======
1161 AppAuth also provides a convenience function, `performActionWithFreshTokens`, which will
1162 automatically handle token refresh if the current access token has expired.

1163 **3 How to Install and Configure the OAuth 2 AS**

1164 **3.1 Platform and System Requirements**

1165 Ping Identity is used as the AS for this build. The AS issues access tokens to the client after successfully
1166 authenticating the resource owner and obtaining authorization as specified in RFC 6749, The OAuth
1167 Authorization Framework [\[21\]](#).

1168 The requirements for Ping Identity can be categorized into three groups: software, hardware, and
1169 network.

1170 **3.1.1 Software Requirements**

1171 The software requirements are as follows:

- 1172 ▪ OS: Microsoft Windows Server, Oracle Enterprise Linux, Oracle Solaris, Red Hat Enterprise, SUSE
1173 Linux Enterprise
- 1174 ▪ Virtual systems: VMware, Xen, Windows Hyper-V
- 1175 ▪ Java environment: Oracle Java Standard Edition
- 1176 ▪ Data integration: Ping Directory, Microsoft Active Directory (AD), Oracle Directory Server,
1177 Microsoft Structured Query Language (SQL) Server, Oracle Database, Oracle MySQL 5.7,
1178 PostgreSQL

1179 **3.1.2 Hardware Requirements**

1180 The minimum hardware requirements are as follows:

- 1181 ▪ Intel Pentium 4, 1.8-gigahertz (GHz) processor
- 1182 ▪ 1 gigabyte (GB) of Random Access Memory (RAM)
- 1183 ▪ 1 GB of available hard drive space

1184 A detailed discussion on this topic and additional information can be found at
1185 [https://documentation.pingidentity.com/pingfederate/pf82/index.shtml#gettingStartedGuide/concept/
1186 systemRequirements.html](https://documentation.pingidentity.com/pingfederate/pf82/index.shtml#gettingStartedGuide/concept_systemRequirements.html).

1187 **3.1.3 Network Requirements**

1188 Ping Identity identifies several ports to be open for different purposes. These purposes can include
1189 communication with the administrative console, runtime engine, cluster engine, and Kerberos engine.

1190 A detailed discussion on each port can be found at

1191 https://documentation.pingidentity.com/pingfederate/pf84/index.shtml#gettingStartedGuide/pf_t_inst
1192 [allPingFederateRedHatEnterpriseLinux.html](#).

1193 In this implementation, we needed ports to be opened to communicate with the administrative console
1194 and the runtime engine.

1195 For this experimentation, we have used the configuration identified in the following subsections.

1196 **3.1.3.1 Software Configuration**

1197 The software configuration is as follows:

- 1198 ▪ OS: CentOS Linux Release 7.3.1611 (Core)
- 1199 ▪ Virtual systems: Vmware ESXI 6.5
- 1200 ▪ Java environment: OpenJDK Version 1.8.0_131
- 1201 ▪ Data integration: AD

1202 **3.1.3.2 Hardware Configuration**

1203 The hardware configuration is as follows:

- 1204 ▪ Processor: Intel(R) Xeon(R) central processing unit (CPU) E5-2420 0 at 1.90 GHz
- 1205 ▪ Memory: 2 GB
- 1206 ▪ Hard drive: 25 GB

1207 **3.1.3.3 Network Configuration**

1208 The network configuration is as follows:

- 1209 ▪ 9031: This port allows access to the runtime engine; this port must be accessible to client
1210 devices and federation partners.
- 1211 ▪ 9999: This port allows the traffic to the administrative console; only PingFederate administrators
1212 need access.

1213 3.2 How to Install the OAuth 2 AS

1214 Before the installation of Ping Identity AS, the prerequisites identified in the following subsections need
1215 to be fulfilled.

1216 3.2.1 Java Installation

1217 Java 8 can be installed in several ways on CentOS 7 using *yum*. Yum is a package manager on the
1218 CentOS 7 platform that automates software processes, such as installation, upgrade, and removal, in a
1219 consistent way.

1220 1. Download the Java Development Kit (JDK) in the appropriate format for your environment, from
1221 Oracle's website; for CentOS, the Red Hat Package Manager (RPM) download can be used:
1222 <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>.

1223 2. As root, install the RPM by using the following command, substituting the actual version of the
1224 downloaded file:

1225 `rpm -ivh jdk-8u151-linux-x64.rpm`

1226 3. Alternatively, the JDK can be downloaded in *.tar.gz* format and unzipped in the appropriate
1227 location (i.e., */usr/share* on CentOS 7).

1228 3.2.2 Java Post Installation

1229 The `alternatives` command maintains symbolic links determining default commands. This command
1230 can be used to select the default Java command. This is helpful even in cases where there are multiple
1231 installations of Java on the system.

1232 1. Use the following command to select the default Java command:

1233 `alternatives --config java`

1234 There are three programs that provide "java."

1235 Selection Command

1236 -----

1237 1 /usr/java/jre1.8.0_111/bin/java

1238 *+ 2 java-1.8.0-openjdk.x86_64 (/usr/lib/jvm/java-1.8.0-openjdk-
1239 1.8.0.131-3.b12.el7_3.x86_64/jre/bin/java)

1240 3 /usr/java/jdk1.8.0_131/jre/bin/java

1241 Enter to keep the current selection[+], or type selection number:

1242 This presents the user with a configuration menu for choosing a Java instance. Once a selection
1243 is made, the link becomes the default command system wide.

- 1244 2. To make Java available to all users, the JAVA_HOME environment variable was set by using the
1245 following command:

1246 echo export JAVA_HOME="/usr/java/latest" > /etc/profile.d/javaenv.sh

- 1247 3. For cryptographic functions, download the *Java Cryptography Extension (JCE) Unlimited Strength
1248 Jurisdiction Policy Files 8* from
1249 <http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html>.

- 1250 4. Decompress and extract the downloaded file. The installation procedure is described in the
1251 Readme document. In the lab, *local_policy.jar* was extracted to the default location, <*java-
1252 home*>/lib/security.Network Configuration.

- 1253 5. Check if the firewall is running or not by using the command below. If it is up, it will return a
1254 status that shows it is running:

1255 firewall-cmd --state

- 1256 a. If it is not running, activate the firewall by using the following command:

1257 sudo systemctl start firewalld.service

- 1258 6. Check if the required ports, 9031 and 9999, are open by using the following command:

1259 firewall-cmd --list-ports

- 1260 a. This command will return the following values:

1261 6031/tcp 9999/udp 9031/tcp 6031/udp 9998/udp 9031/udp 9999/tcp 9998/tcp
1262 8080/tcp

1263 From the returned ports, we can determine which ports and protocols are open.

- 1264 b. In case the required ports are not open, issue the command below. It should return
1265 success.

1266 **firewall-cmd --zone=public --permanent --add-port=9031/tcp**

1267 success

- 1268 7. Reload the firewall by using the following command to make the rule change take effect:

1269 **firewall-cmd --reload**

1270 Success

- 1271 a. Now, when the open ports are listed, the required ports should show up:

1272 **firewall-cmd --zone=public --list-ports**

1273 6031/tcp 9999/udp 9031/tcp 6031/udp 9998/udp 9031/udp 9999/tcp 9998/tcp
1274 8080/tcp 5000/tcp

1275 **3.2.3 PingFederate Installation**

1276 Ping installation documentation is available at

1277 https://docs.pingidentity.com/bundle/pf_sm_installPingFederate_pf82/page/pf_t_installPingFederateRedHatEnterpriseLinux.html?#.

1279 Some important points are listed below:

- 1280 ▪ Obtain a Ping Identity license. It can be acquired from
<https://www.pingidentity.com/en/account/sign-on.html>.
- 1282 ▪ For this experiment, installation was done using the zip file. Installation was done at */usr/share*.
- 1283 ▪ The license was updated.
- 1284 ▪ 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?#

1290 The following configuration procedures are completed in the PingFederate administrative console,
1291 which is available at *https://<ping-server-hostname>:9999/pingfederate/app*.

1292 **3.2.4 Certificate Installation**

1293 During installation, PingFederate generates a self-signed TLS certificate, which is not trusted by desktop
1294 or mobile device browsers. A certificate should be obtained from a trusted internal or external CA and
1295 should be installed on the PingFederate server. The private key and signed certificate can be uploaded
1296 and activated for use on the run-time server port and the admin port by navigating to **Server Settings** in
1297 the console and clicking on **SSL Server Certificates**.

1298 In addition, most server roles described in this guide will require the creation of a signing certificate. This
1299 is required for a SAML or OIDC IdP, and for an OAuth AS if access tokens will be issued as JWTs. To
1300 create or import a signing certificate, under **Server Configuration—Certificate Management**, click **Signing**
1301 & **Decryption Keys & Certificates**. A self-signed certificate can be created, or a trusted certificate can be
1302 obtained and uploaded there.

1303 **3.3 How to Configure the OAuth 2 AS**

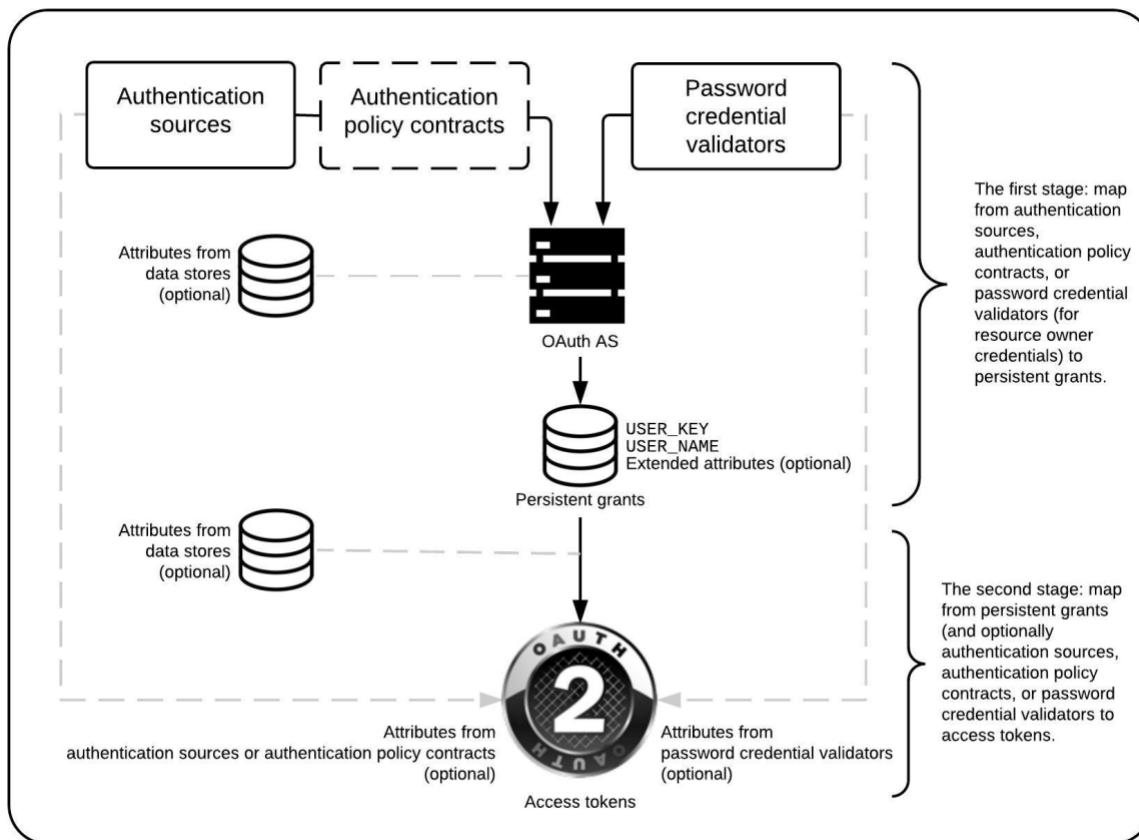
1304 Configuration of a Ping OAuth 2 AS is described at

1305 https://documentation.pingidentity.com/pingfederate/pf82/index.shtml#concept_usingOauthMenuSelections.html.

1307 This guide documents the configuration for an AS serving the role of the *idm.sandbox* server hosted in
 1308 the Motorola Solutions cloud instance, as depicted in Figure 1-1. This AS is configured to support the
 1309 three usage scenarios—local user authentication at the AS, redirection to a SAML IdP, and redirection to
 1310 an OIDC IdP—and to initiate the correct login flow based on an IdP discovery mechanism.

1311 An understanding of the PingFederate OAuth implementation helps provide context for the
 1312 configurations documented in this guide. PingFederate supports several different authentication flows
 1313 and mechanisms, but there is a common framework for how user attributes are mapped into OAuth
 1314 tokens. This framework is depicted in Figure 3-1, which is taken from PingFederate’s documentation at
https://documentation.pingidentity.com/pingfederate/pf83/index.shtml#concept_mappingOauthAttributes.html.
 1316

1317 **Figure 3-1 Access Token Attribute Mapping Framework**



1318
 1319 The overall OAuth processing flow at the AS is as follows:
 1320 1. The AS receives an OAuth authorization request from an unauthenticated user.

1321 2. The AS authenticates the user through the configured authentication adapters, IdP connections,
1322 and/or authentication policies.

1323 3. Information from adapters or policy contracts, optionally combined with user information
1324 retrieved from data stores such as Lightweight Directory Access Protocol (LDAP), are used to
1325 build a persistent grant context. The two mandatory attributes in the persistent grant context are
1326 listed below:

- 1327 ▪ **USER_KEY**—This is a globally unique user identifier. For ASs that interact with multiple
1328 IdPs, this name should be resistant to naming collisions across user organizations (e.g.,
1329 email address or distinguished name).
- 1330 ▪ **USER_NAME**—If the user is prompted to authorize the request, this name will be
1331 displayed on the page, so a user-friendly name, such as [givenName lastName], could be
1332 used here; the name does not need to be unique.

1333 4. If authorization prompts are enabled, the user is prompted to approve the authorization
1334 request; for this lab build, these prompts were disabled on the assumption that fast access to
1335 applications is a high priority for the PSFR community.

1336 5. If the request is authorized, a second mapping process takes place to populate the access token
1337 with information from the persistent grant and, optionally, from adapters, policy contracts, or
1338 data stores.

1339 Note that persistent grant attributes are stored and can be retrieved and reused when the client uses a
1340 refresh token to obtain a new access token, whereas attributes that are looked up in the second stage
1341 would be looked up again during the token refresh request. Storing attributes in the persistent grant can
1342 therefore reduce the need for repeated directory queries; however, it may be preferable to always
1343 query some attributes that are subject to change (like account status) again when a new access token is
1344 requested. In addition, it is important to note that storing persistent grant attributes requires a
1345 supported relational database or LDAP data store.

1346 The following steps go through the configuration of the AS.

1347 1. Enable the PingFederate installation to work as an AS. This can be done in the following steps:

1348 a. Under **Main**, click the **Server Configuration** section tab, and then click **Server Settings**.

1349 b. In **Server Settings**, click the **Roles & Protocols** tab. The Roles & Protocols screen will
1350 appear as shown in Figure 3-2.

1351 i. Click **ENABLE OAUTH 2.0 AUTHORIZATION SERVER (AS) ROLE**.

1352 ii. Click **ENABLE IDENTITY PROVIDER (IDP) ROLE AND SUPPORT THE FOLLOWING**,
1353 and then under it, click **SAML 2.0**. Although this server does not act as a SAML
1354 IdP, it is necessary to enable the IdP role and at least one protocol to configure
1355 the local user authentication use case.

- 1356 iii. Click **ENABLE SERVICE PROVIDER (SP) ROLE AND SUPPORT THE FOLLOWING**,
1357 and then under it, click **SAML 2.0** and **OPENID CONNECT**; this enables
1358 integration with both types of IdPs.

1359 Figure 3-2 Server Roles for AS

The screenshot shows the 'Server Settings' page for the Authorization Server (AS) role in PingFederate. The left sidebar has a 'MAIN' category with 'IdP Configuration', 'SP Configuration', 'OAuth Settings', and 'Server Configuration'. 'Server Configuration' is selected and highlighted in blue. The main content area has a header 'Server Settings' with tabs: System Administration, System Info, Runtime Notifications, Runtime Reporting, and Account Management. The 'System Administration' tab is active. Below it is a sub-header 'Roles & Protocols' with tabs: Roles & Protocols, Federation Info, System Options, Metadata Signing, and Metadata Lifetime. The 'Roles & Protocols' tab is active. A sub-sub-header 'Summary' is also present. The main content includes sections for enabling roles and 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):**

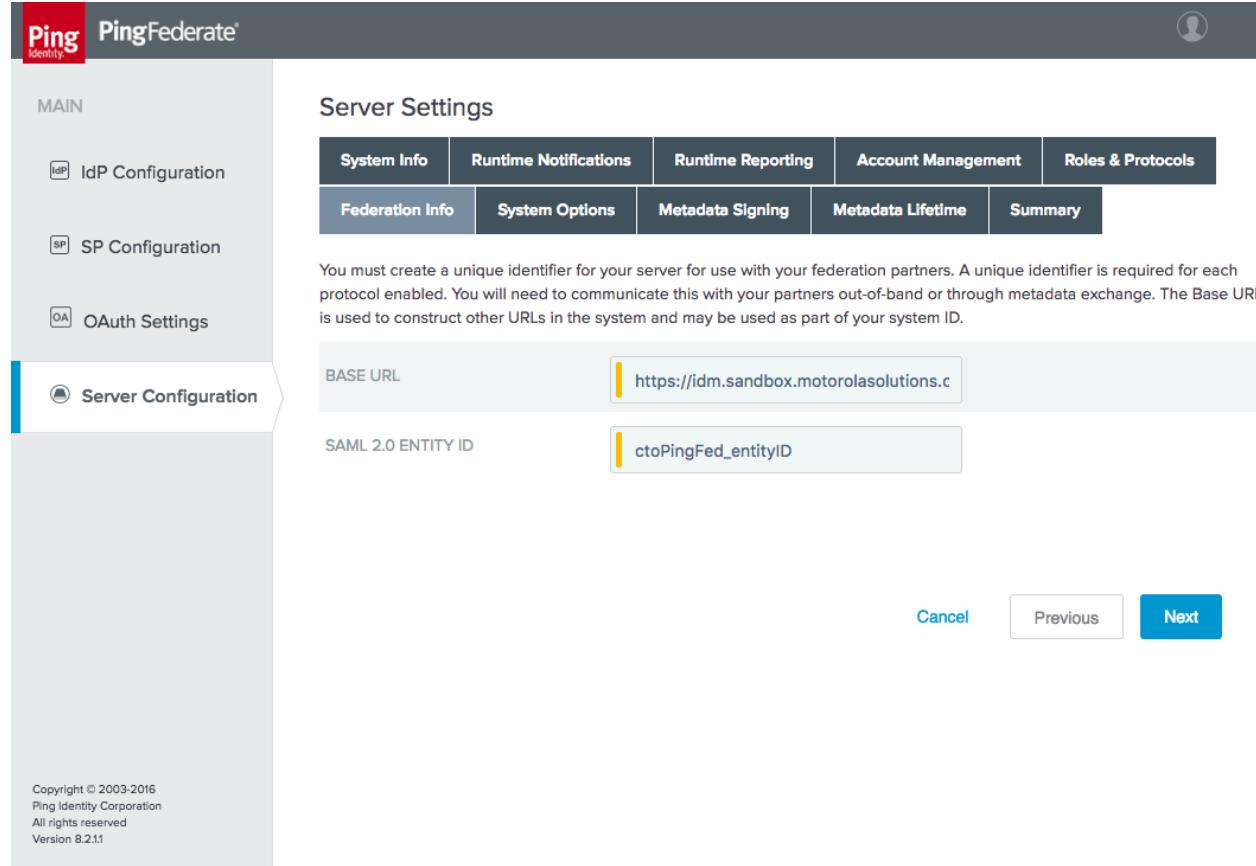
At the bottom right are buttons for 'Cancel', 'Previous', 'Next', and 'Save'.

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

1360

- 1361 c. Also under **Server Settings**, on the **Federation Info** tab, enter the **BASE URL** and **SAML**
 1362 **2.0 ENTITY ID** (Figure 3-3). The **BASE URL** should use a public DNS name that is
 1363 resolvable by any federation partners. The **SAML 2.0 ENTITY ID** is simply an identifier
 1364 string that must be unique among federation partners; it is recommended to be a
 1365 Uniform Resource Identifier (URI), per the SAML 2.0 Core specification [\[22\]](#).

1366 **Figure 3-3 Federation Info**



The screenshot shows the 'PingFederate' interface. The left sidebar has a 'MAIN' section with 'IdP Configuration', 'SP Configuration', 'OAuth Settings', and 'Server Configuration' (which is selected and highlighted in blue). The main content area is titled 'Server Settings' and contains a navigation bar with tabs: 'System Info', 'Runtime Notifications', 'Runtime Reporting', 'Account Management', 'Roles & Protocols', 'Federation Info' (which is selected and highlighted in blue), 'System Options', 'Metadata Signing', 'Metadata Lifetime', and 'Summary'. Below this, there is a note: '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.' There are two input fields: 'BASE URL' containing 'https://idm.sandbox.motorolasolutions.c' and 'SAML 2.0 ENTITY ID' containing 'ctoPingFed_entityID'. At the bottom right are 'Cancel', 'Previous', and 'Next' buttons. In the bottom left corner of the main content area, there is a copyright notice: 'Copyright © 2003-2016 Ping Identity Corporation All rights reserved Version 8.2.1.1'.

- 1367 2. The next step is to configure the OAuth AS. Click the **OAuth Settings** section tab under **Main**.
 1368 a. Click **Authorization Server Settings** under the **Authorization Server** header. This displays
 1369 the **Authorization Server Settings** (Figure 3-4).

1371 Figure 3-4 AS Settings

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)	<input type="text" value="60"/>
AUTHORIZATION CODE ENTROPY (BYTES)	<input type="text" value="30"/>

Refresh Token and Persistent Grant Settings

PERSISTENT GRANT LIFETIME (BLANK FOR INDEFINITE)	<input type="text"/>	Days	<input type="button" value="▼"/>
--	----------------------	------	----------------------------------

REFRESH TOKEN LENGTH (CHARACTERS)	<input type="text" value="42"/>
ROLL REFRESH TOKEN VALUES (DEFAULT POLICY)	<input checked="" type="checkbox"/>
MINIMUM INTERVAL TO ROLL REFRESH TOKENS (HOURS)	<input type="text" value="0"/>
REUSE EXISTING PERSISTENT ACCESS GRANTS FOR GRANT TYPES	<input checked="" type="checkbox"/> IMPLICIT <input checked="" type="checkbox"/> AUTHORIZATION CODE <input type="checkbox"/> RESOURCE OWNER PASSWORD CREDENTIALS
BYPASS AUTHORIZATION FOR PREVIOUSLY APPROVED PERSISTENT GRANTS	<input type="checkbox"/>
ALLOW UNIDENTIFIED CLIENTS TO MAKE RESOURCE OWNER PASSWORD CREDENTIALS GRANTS	<input type="checkbox"/>
ALLOW UNIDENTIFIED CLIENTS TO REQUEST EXTENSION GRANTS	<input type="checkbox"/>

Persistent Grant Extended Attributes

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

OAuth Administrative Web Services Settings

PASSWORD CREDENTIAL VALIDATOR	<input type="button" value="- SELECT -"/>
-------------------------------	---

1372

1373 The default settings are suitable for the lab build architecture; organizations may wish
1374 to customize these default settings in accordance with organizational security policy or
1375 usage requirements. Some notes on individual settings are provided below:

- 1376 ■ **AUTHORIZATION CODE TIMEOUT (SECONDS):** Once an authorization code has
1377 been returned to a client, it must be exchanged for an access token within this
1378 interval. This reduces the risk of an unauthorized client obtaining an access
1379 token through brute-force guessing or intercepting a valid client's code. *Proof*
1380 *Key for Code Exchange (PKCE)* [23], as implemented by the AppAuth library, is
1381 another useful mechanism to protect the authorization code.
 - 1382 ■ **AUTHORIZATION CODE ENTROPY (BYTES):** length of the authorization code
1383 returned by the AS to the client, in bytes
 - 1384 ■ **REFRESH TOKEN LENGTH (CHARACTERS):** length of the refresh token, in
1385 characters
 - 1386 ■ **ROLL REFRESH TOKEN VALUES (DEFAULT POLICY):** When selected, the OAuth
1387 AS generates a new refresh token value when a new access token is obtained.
 - 1388 ■ **MINIMUM INTERVAL TO ROLL REFRESH TOKENS (HOURS):** the minimum
1389 number of hours that must pass before a new refresh token value can be issued
 - 1390 ■ **REUSE EXISTING PERSISTENT ACCESS GRANTS FOR GRANT TYPES:**
 - 1391 • **IMPLICIT:** Consent from the user is requested only for the first OAuth
1392 resource request associated with the grant.
 - 1393 • **AUTHORIZATION CODE:** Same as above if the **BYPASS AUTHORIZATION**
1394 **FOR PREVIOUSLY APPROVED PERSISTENT GRANTS** is selected; this can
1395 be used to prompt the user for authorization only once to avoid
1396 repeated prompts for the same client.
 - 1397 ■ **PASSWORD CREDENTIAL VALIDATOR:** Required for HTTP Basic authentication if
1398 the OAuth Representational State Transfer Web Service is used for managing
1399 client applications; this functionality was not used for this build.
- 1400 3. Next, configure scopes, as required, for the application. Click the **OAuth Settings** section tab,
1401 and then click **Scope Management**. The specific scope values will be determined by the client
1402 application developer. Generally speaking, scopes refer to different authorizations that can be
1403 requested by the client and granted by the user. Access tokens are associated with the scopes
1404 for which they are authorized, which can limit the authorities granted to clients. Figure 3-5
1405 shows several scopes that were added to the AS for this lab build that have specific meanings in
1406 the PSX applications suite.

1407 **Figure 3-5 Scopes**

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

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

1408

- 1409 4. Define an Access Token Management Profile. This profile determines whether access tokens are
 1410 issued as simple reference token strings or as JWTs. For this lab build, JWTs were used. JWTs are
 1411 signed and optionally encrypted, so resource servers can validate them locally and they can
 1412 contain user attributes and other information. Reference tokens are also a viable option, but
 1413 resource servers must contact the AS's introspection endpoint to determine whether they are
 1414 valid and must obtain the granted scopes and any other information associated with them. The
 1415 Access Token Management Profile also defines any additional attributes that will be associated
 1416 with the token.
- 1417 a. Create an Access Token Manager by following these steps:
- 1418 i. Click the **OAuth Settings** section tab, click **Access Token Management**, and then
 1419 click **Create New Instance**.
- 1420 ii. On the **Type** tab, give the instance a meaningful name and ID, and select the
 1421 token type (Figure 3-6).

1422 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

1423

- 1424 5. On the next tab, **Instance Configuration**, select a symmetric key or certificate to use for JWT
 1425 signing (Figure 3-7). In this instance, a signing certificate was created as described in
 1426 [Section 3.2.4](#). Tokens can also optionally be encrypted using JSON Web Encryption (JWE) [24]; in
 1427 this case, the client developer would provide a certificate in order to receive encrypted
 1428 messages. JWE was not used in the lab build.

1429 Figure 3-7 Access Token Manager Instance Configuration

PingFederate^{Identity}

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 (An identifier for the given key)	KEY (Encoded symmetric key)	ENCODING (How the binary key is encoded as a string)	Action
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 (An identifier for the given key)	CERTIFICATE (Requires an EC key or RSA key length of at least 2048 bits)	Action
jwt signer	CN=as1.cpsd.mso, OU=NCCoE, O=NIST, L=Rockville, ST=Maryland, C=US	Edit Delete
Add a new row to 'Certificates'		

Copyright © 2003-2016
Ping Identity Corporation
All rights reserved
Version 8.2.0

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**

1430

1431 6. On the **Access Token Attribute Contract** tab, add the two values **realm** and **sub** to the attribute
 1432 contract (Figure 3-8).

1433 **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'.

1434
 1435 7. The **Resource URIs** and **Access Control** tabs were not used for this build. Click **Save** to complete
 1436 the Access Token Manager.

1437 8. Next, one or more OAuth clients need to be registered with the AS. In the Motorola Solutions
 1438 use case, the PSX Cockpit application is registered as a client. OAuth Client registration is
 1439 described for PingFederate at:

1440 https://documentation.pingidentity.com/pingfederate/pf82/index.shtml#concept_configuringClient.html.

1442 To create a new client, click the **OAuth Settings** section tab, click **Clients**, and then click **Create**
 1443 **New**. Clients are displayed on the rightmost side of the screen in the **OAuth Settings** window.
 1444 Once **Create New** is clicked, the screen shown in Figure 3-9 and Figure 3-10 will appear. Due to
 1445 the vertical size of the pages of this document, the screenshot is divided into two parts for
 1446 legibility.

1447 Figure 3-9 OAuth Client Registration, Part 1

The screenshot shows the PingFederate interface for managing OAuth clients. The left sidebar has a 'MAIN' section with 'IdP Configuration', 'SP Configuration', 'OAuth Settings' (which is selected and highlighted in blue), and 'Server Configuration'. The main content area is titled 'Client' and describes managing configuration and policy information about a client. It shows a 'CLIENT ID' of 'ssoclient_nist', 'CLIENT AUTHENTICATION' set to 'NONE' (radio button unselected), and 'CLIENT SECRET' set to 'CLIENT SECRET' (radio button selected). A 'SECRET' field contains a series of dots ('.....'), and a 'Generate Secret' button is available. Below this, a 'CHANGE SECRET' link is shown. There is also a 'CLIENT TLS CERTIFICATE' section with a radio button unselected. The 'ISSUER' section has a dropdown menu labeled '- SELECT -'. The 'SUBJECT DN' section has a text input field and a note: 'You can also extract the Subject DN from a certificate file.' with 'Choose file' and 'Extract' buttons. At the bottom left, there is copyright information: 'Copyright © 2003-2016 Ping Identity Corporation. All rights reserved. Version 8.2.1'.

1448

1449 Figure 3-10 OAuth Client Registration, Part 2

	<p>NAME</p> <input type="text" value="ssoclient_nist"/> <p>DESCRIPTION</p> <div style="border: 1px solid #ccc; height: 100px; margin-top: 10px;"></div>									
	<p>REDIRECT URIS</p> <table border="0"> <tr> <td style="width: 30%;">Redirection URIs</td> <td style="width: 40%;"><input type="text" value="http://localhost/"/></td> <td>Action Edit Delete</td> </tr> <tr> <td></td> <td><input type="text" value="napps://localhost/"/></td> <td>Action Edit Delete</td> </tr> <tr> <td></td> <td><input type="text"/></td> <td>Add</td> </tr> </table> <p>LOGO URL</p> <div style="border: 1px solid #ccc; width: 100%; height: 30px; margin-top: 10px;"></div>	Redirection URIs	<input type="text" value="http://localhost/"/>	Action Edit Delete		<input type="text" value="napps://localhost/"/>	Action Edit Delete		<input type="text"/>	Add
Redirection URIs	<input type="text" value="http://localhost/"/>	Action Edit Delete								
	<input type="text" value="napps://localhost/"/>	Action Edit Delete								
	<input type="text"/>	Add								
	<p>BYPASS AUTHORIZATION APPROVAL</p> <p><input checked="" type="checkbox"/> Bypass</p> <p><input type="checkbox"/> Restrict</p> <p>RESTRICT SCOPES</p> <p><input checked="" type="checkbox"/> Authorization Code</p> <p><input type="checkbox"/> Resource Owner Password Credentials</p> <p><input checked="" type="checkbox"/> Refresh Token</p> <p><input checked="" type="checkbox"/> Implicit</p> <p><input type="checkbox"/> Client Credentials</p> <p><input type="checkbox"/> Access Token Validation (Client is a Resource Server)</p> <p><input type="checkbox"/> Extension Grants</p>									
	<p>ALLOWED GRANT TYPES</p> <p>DEFAULT ACCESS TOKEN MANAGER</p> <div style="border: 1px solid #ccc; padding: 2px; margin-top: 10px;">fidoJwt</div> <p>PERSISTENT GRANTS EXPIRATION</p> <p><input checked="" type="radio"/> Use Global Setting</p> <p><input type="radio"/> Grants Do Not Expire</p> <p><input type="radio"/> <input type="text"/> Days</p>									
	<p>REFRESH TOKEN ROLLING POLICY</p> <p><input checked="" type="radio"/> Use Global Setting <input type="radio"/> Don't Roll <input type="radio"/> Roll</p> <p>OPENID CONNECT</p> <p>ID Token Signing Algorithm</p> <div style="border: 1px solid #ccc; padding: 2px; margin-top: 10px;">HMAC using SHA-256</div> <p>Policy</p> <div style="border: 1px solid #ccc; padding: 2px; margin-top: 10px;">fidoPolicy</div> <p><input type="checkbox"/> Grant Access to Session Revocation API</p>									

1450

- 1451 The following are notes on the parameters on this screen:
- 1452 ▪ **CLIENT ID:** This is a required parameter. This is the unique identifier accompanied with
1453 each request that is presented to the AS's token and authorization endpoints. For this
1454 lab build, Motorola Solutions assigned a client ID of "ssoclient_nist" for the instances of
1455 their applications on the test devices.
- 1456 ▪ **CLIENT AUTHENTICATION:** May be set to **NONE**, **CLIENT SECRET** (for HTTP basic
1457 authentication), or **CLIENT TLS CERTIFICATE**. For native mobile application clients, there
1458 is no way to protect a client secret or private key and provide it to all instances of the
1459 application with any guarantee of confidentiality, as a user might be able to
1460 reverse-engineer the application to obtain any secrets delivered with it, or to debug the
1461 application to capture any secrets delivered at run-time. Therefore, a value of **NONE** is
1462 acceptable for native mobile applications, when mitigated with the use of PKCE. For web
1463 clients, servers are capable of protecting secrets; therefore, some form of client
1464 authentication should be required.
- 1465 ▪ **REDIRECT URIS:** Redirect URIs are the URIs to which the OAuth AS may redirect the
1466 resource owner's user-agent after authorization is obtained. A redirect URI is used with
1467 the **Authorization Code** and **Implicit** grant types. This value is typically provided by the
1468 application developer to the AS administrator.
- 1469 ▪ **ALLOWED GRANT TYPES:** These are the allowed grant types for the client. For this lab
1470 build, the **Authorization Code** grant type was used exclusively.
- 1471 ▪ **DEFAULT ACCESS TOKEN MANAGER:** This is the Access Token Manager profile to be
1472 used for this client.
- 1473 ▪ **PERSISTENT GRANTS EXPIRATION:** This setting offers the option to override the global
1474 AS persistent grants settings for this client.
- 1475 ▪ **REFRESH TOKEN ROLLING POLICY:** This setting offers the option to override the global
1476 AS token rolling policy settings for this client.

1477 Once these values are set, click **Save** to store the client.

1478 This completes the required configuration for the AS's interactions with OAuth clients. The following
1479 section outlines the steps to set up the AS to authenticate users.

1480 3.4 How to Configure the OAuth 2 AS for Authentication

1481 In this section, the AS is configured to authenticate users locally or through federation with a SAML or
1482 OIDC IdP. These settings depend on the selection of roles and protocols, as shown in Figure 3-2,
1483 therefore, ensure that has been completed before proceeding.

1484 [**3.4.1 How to Configure Direct Authentication**](#)

1485 The AS was configured to authenticate users with FIDO UAF authentication. This depends on the NNAS,
1486 Nok Nok Labs Gateway, and Nok Nok Labs UAF Plugin for PingFederate. See [Section 5](#) for the installation
1487 and configuration instructions for those components. This section assumes that those components have
1488 already been installed and configured.

1489 [**3.4.1.1 Configure Adapter Instance**](#)

- 1490 1. First, an instance of the FIDO UAF adapter must be configured. Click the **IdP Configuration**
1491 section tab, and then click **Adapters** under **Application Integration**.
- 1492 2. Click **Create New Instance** to create an IdP adapter instance. This will bring up the new tabbed
1493 screen shown in Figure 3-11.
 - 1494 a. On the **Type** tab, the **INSTANCE NAME** and **INSTANCE ID** are internal identifiers and can
1495 be set to any meaningful values. The **TYPE** selection, “FIDO Adapter,” will not appear
1496 until the Nok Nok Labs UAF plugin has been successfully installed on the PingFederate
1497 server as described in [Section 5](#).

1498 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

1499

- 1500 b. On the **IdP Adapter** tab, specify the URLs for the Nok Nok Labs API and Gateway
 1501 endpoints (Figure 3-12).
- 1502 i. The **NNL SERVER POLICY NAME** field can be used to select a custom policy, if
 1503 one has been defined on the Nok Nok Labs server; for this build, the default
 1504 policy was used.

1505 Figure 3-12 FIDO Adapter Settings

The screenshot shows the 'Manage IdP Adapter Instances | Create Adapter Instance' page in the PingFederate interface. The left sidebar lists 'MAIN' categories: 'IdP Configuration' (selected), 'SP Configuration', 'OAuth Settings', and 'Server Configuration'. The main content area is titled 'Set the details necessary for FIDO adapter configuration'. It contains a table with the following data:

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

At the bottom left, there is a copyright notice: 'Copyright © 2003-2016 Ping Identity Corporation. All rights reserved. Version 8.2.2.0'. At the bottom right are 'Cancel', 'Previous', and 'Next' buttons.

1506

- 1507 c. The **Extended Contract** tab was also left as the default for the adapter, which provides
 1508 the **riskscore**, **transactionid**, **transactiontext**, and **username** values (Figure 3-13). If
 1509 desired, additional attributes could be added to the contract and looked up in a user
 1510 directory, based on the username returned from the adapter.

1511 Figure 3-13 FIDO Adapter Contract

1512

- 1513 d. On the **Adapter Attributes** tab, select the **Pseudonym** checkbox for **username**.
 1514 Pseudonyms were not used in the lab build, but a selection is required on this tab.
- 1515 e. There is no need to configure an adapter contract, unless attributes have been added on
 1516 the **Extended Contract** tab. Clicking **Done** and then **Save** completes the configuration of
 1517 the adapter. Clicking the adapter name in the list of adapters brings up the Adapter
 1518 Instance **Summary** tab, which lists all of the configured settings (Figure 3-14).

1519 **Figure 3-14 FIDO Adapter Instance Summary**

The screenshot shows the PingFederate interface with the title "Manage IdP Adapter Instances | Create Adapter Instance". The left sidebar has tabs for MAIN, IdP Configuration (selected), SP Configuration, OAuth Settings, and Server Configuration. The main area has tabs for Type, IdP Adapter, Extended Contract, Adapter Attributes, Adapter Contract Mapping, and Summary. The "Create Adapter Instance" dialog is open, showing the following configuration:

Type	Value
Instance Name	fidonly
Instance Id	fidonly
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/nniV2/auth
NNI Gateway API Endpoint	https://noknok.sandbox.motorolasolutions.com:8443/nngateway/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)

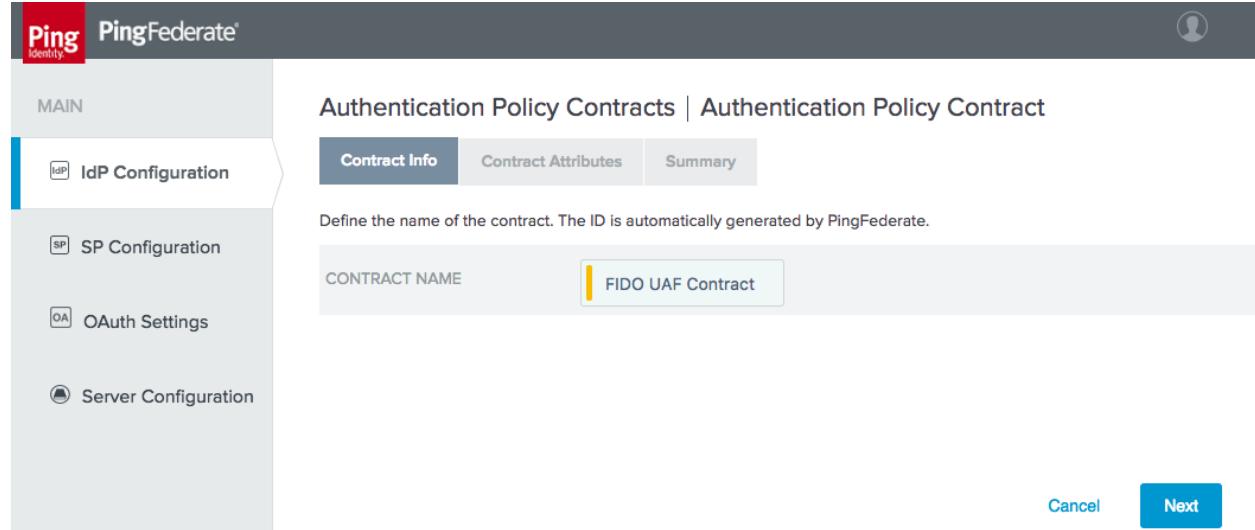
1520

1521 Some additional configurations are needed to tie this authentication adapter to the issuance of an
 1522 OAuth token. It is possible to directly map the adapter to the access token context, but because the
 1523 adapter will be incorporated into an authentication policy in this case, an Authentication Policy Contract
 1524 Mapping is used instead.

1525 *3.4.1.2 Create Policy Contract*

- 1526 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.
- 1529 2. Click **Create New Contract**.
 - a. On the **Contract Info** tab, give the contract a meaningful name (Figure 3-15).

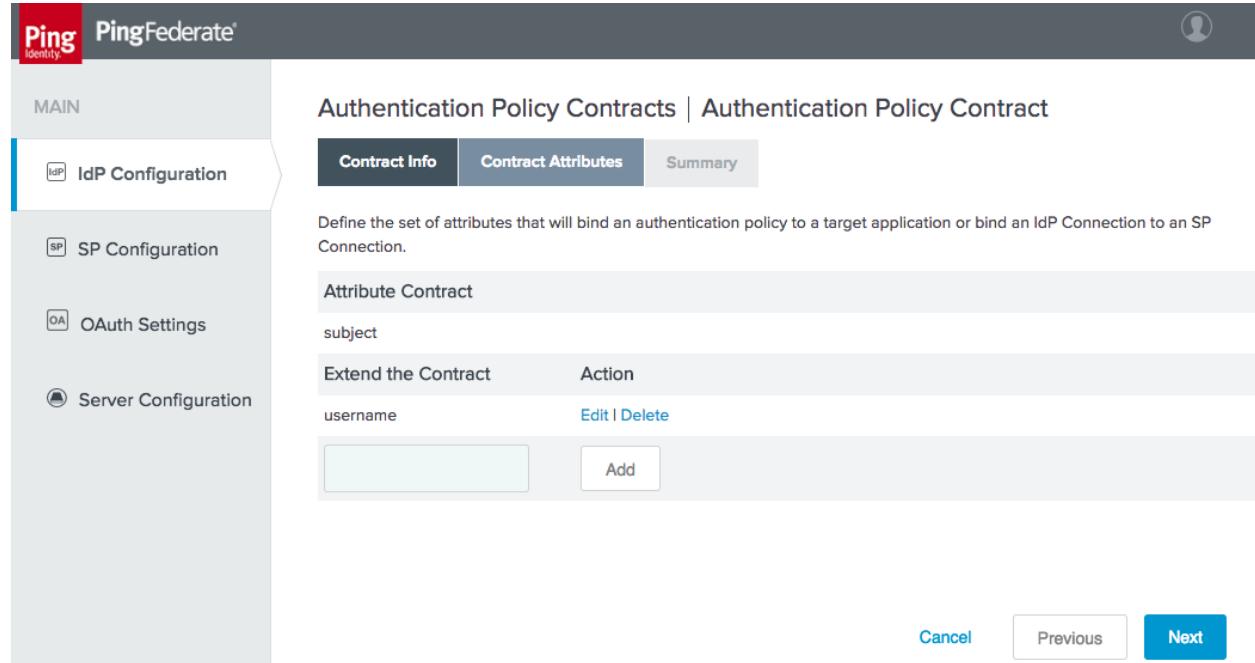
1531 **Figure 3-15 Policy Contract Information**



1532

1533 b. On the **Contract Attributes** tab, add a value called **username** (Figure 3-16).

1534 **Figure 3-16 Policy Contract Attributes**



1535

1536 c. Click **Done**, and then click **Save** to save the new contract.

1537 **3.4.1.3 Create Policy Contract Mapping**

- 1538 1. Create a mapping from the policy contract to the OAuth persistent grant. Click the **OAuth Settings** section tab, and then click **Authentication Policy Contract Mapping** under **Token & Attribute Mapping**.
- 1541 a. Select the newly created policy contract, and then click **Add Mapping** (Figure 3-17).

1542 **Figure 3-17 Create Authentication Policy Contract Mapping**

- 1543
- 1544 2. An attribute source could be added at this point to look up additional user attributes, but this is
1545 not necessary. Click **Save**.
- 1546 3. Skip the **Attribute Sources & User Lookup** tab.
- 1547 4. On the **Contract Fulfillment** tab, map both **USER_KEY** and **USER_NAME** to the **subject** value
1548 returned from the policy contract (Figure 3-18).

1549 **Figure 3-18 Authentication Policy Contract Fulfillment**

MAIN

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

1550

1551 5. No issuance criteria were specified. Click **Next**, and then click **Save** to complete the mapping.

1552 *3.4.1.4 Create Access Token Mapping*

1553 Finally, an access token mapping needs to be created. In this simple case, the adapter only provides a
1554 single attribute (username) and it is stored in the persistent grant, so a default attribute mapping can be
1555 used.

- 1556 1. On the **OAuth Settings** section tab, under **Token & Attribute Mapping**, click **Access Token**
1557 **Mapping**.
- 1558 a. Select **Default** for the **CONTEXT** (Figure 3-19).
- 1559 b. Select the **ACCESS TOKEN MANAGER** created previously (Figure 3-19).

1560 Figure 3-19 Create Access Token Attribute Mapping

MAIN

- IdP IdP Configuration
- SP SP Configuration
- OA OAuth Settings
- SC 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

1561

- 1562 c. Click **Add Mapping**.
- 1563 d. Click **Next** to skip the **Attribute Sources & User Lookup** tab.
- 1564 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**.
- 1565
- 1566

1567 Figure 3-20 Access Token Mapping Contract Fulfillment

MAIN

- IdP IdP Configuration
- SP SP Configuration
- OA OAuth Settings
- SC 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

Cancel Previous Next

1568

- 1569 f. Click **Next** through the **Issuance Criteria** tab, and then click **Save**.
- 1570 2. To complete the setup for direct authentication, the FIDO UAF adapter needs to be included in
1571 an authentication policy as described in [Section 3.4.4.2](#).

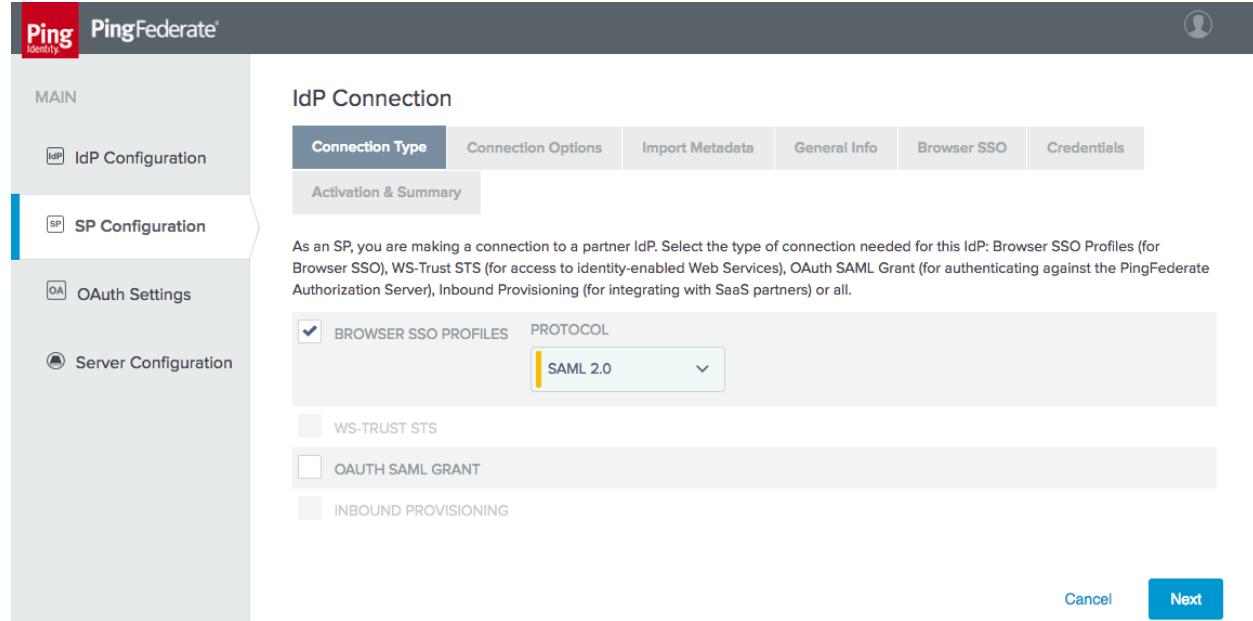
1572 [3.4.2 How to Configure SAML Authentication](#)

1573 This section explains how to configure the AS to accept SAML authentication assertions from a SAML 2.0
1574 IdP. This configuration is for RP-initiated SAML web browser SSO, where the authentication flow begins
1575 at the AS and the user is redirected to the IdP. Here, it is assumed that all of the steps outlined in
1576 [Section 3.4](#) have been completed, particularly enabling the SP role and protocols.

1577 [3.4.2.1 Create IdP Connection](#)

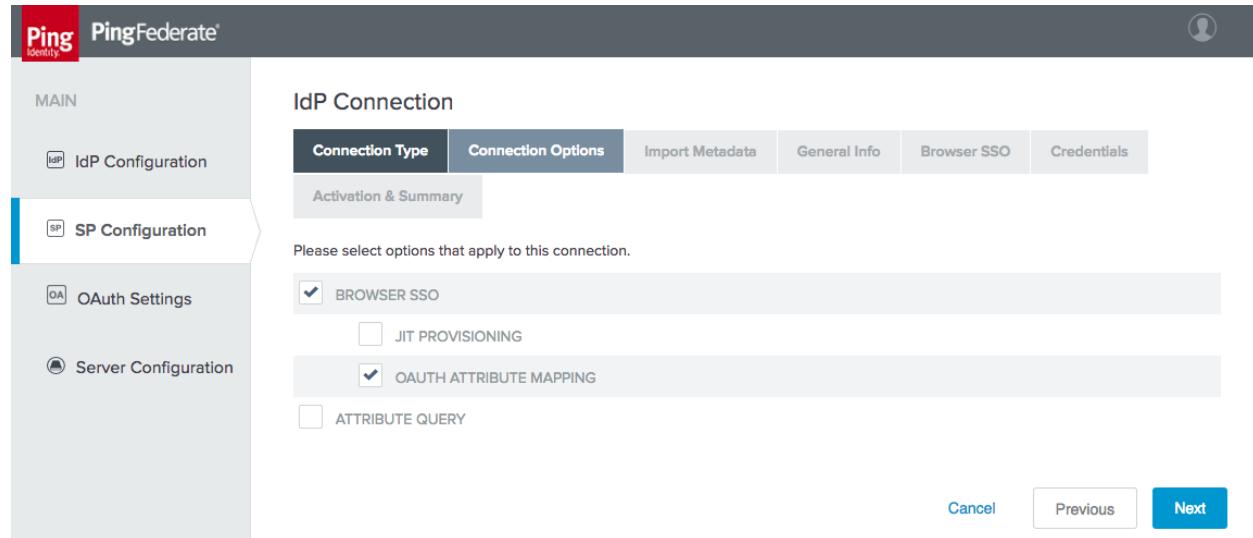
1578 Establishing the relationship between the AS and IdP requires coordination between the administrators
1579 of the two servers, which will typically belong to two separate organizations. The administrators of the
1580 SAML IdP and RP will need to exchange their **BASE URL** and **SAML 2.0 ENTITY ID** values (available on the
1581 **Federation Info** tab under **Server Settings**) to complete the configuration. The IdP administrator must
1582 also provide the signing certificate of the IdP. If assertions will be encrypted, the AS administrator will
1583 need to provide the IdP administrator with the certificate to be used for the public key. Alternatively,
1584 administrators can export their SAML metadata and provide it to the other party to automate parts of
1585 the setup.

- 1586 1. On the **SP Configuration** section tab, click **Create New** under **IdP Connections**.
- 1587 a. On the **Connection Type** tab, select **BROWSER SSO PROFILES**, and choose **SAML 2.0** for
1588 the **PROTOCOL** (Figure 3-21). If these options are not present, ensure that the roles are
1589 selected correctly in **Server Settings**.

1590 **Figure 3-21 Create IdP Connection**

1591

- 1592 b. On the **Connection Options** tab, select **BROWSER SSO**, and then under it, **OAUTH ATTRIBUTE MAPPING** (Figure 3-22).
- 1593

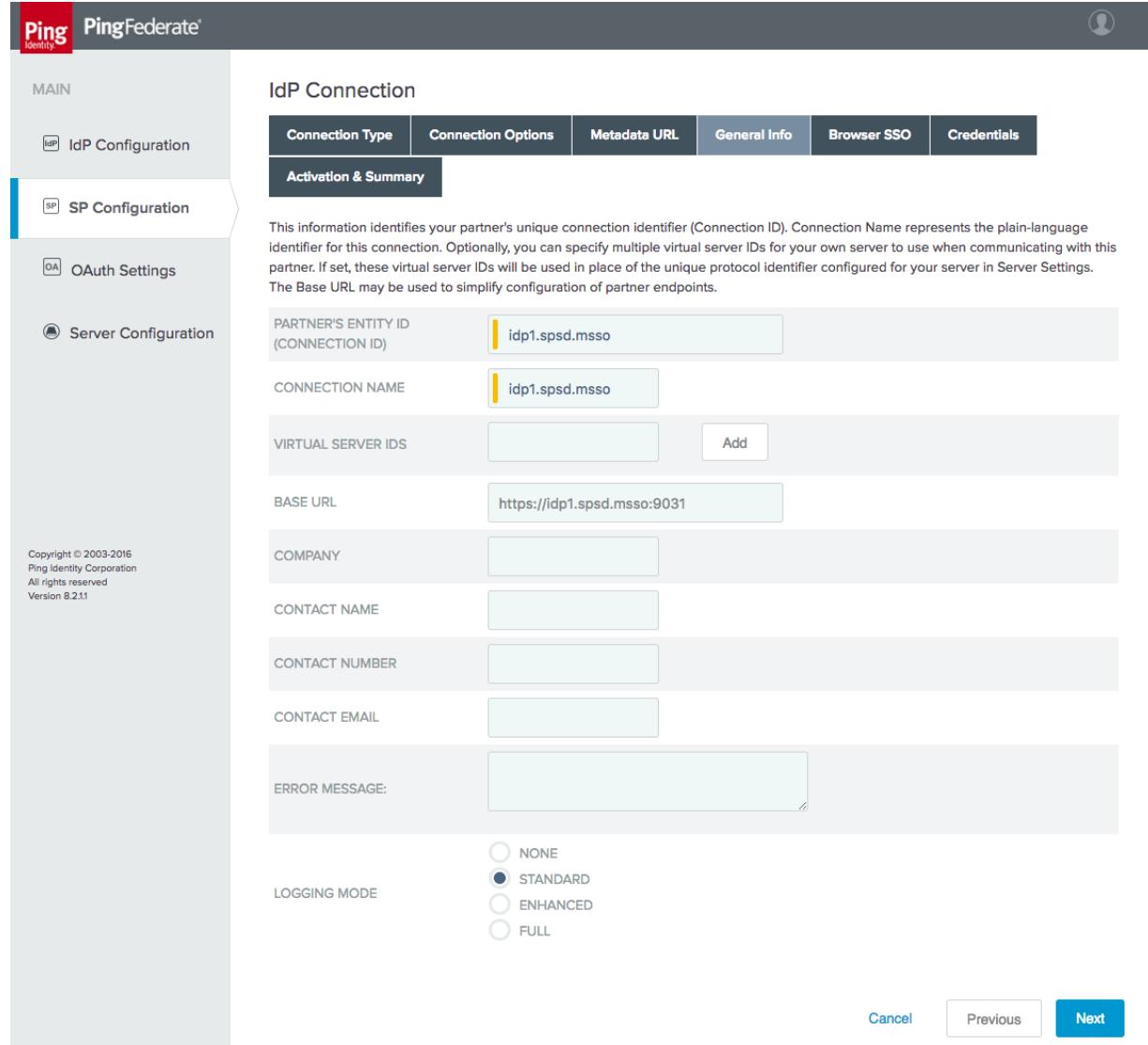
1594 **Figure 3-22 IdP Connection Options**

1595

- 1596 c. Metadata import was not configured for the lab build; therefore, skip the **Import Metadata** tab.
- 1597

- 1598 d. On the **General Info** tab, enter the **PARTNER'S ENTITY ID (CONNECTION ID)** and **BASE URL**
 1599 of the IdP, and provide a **CONNECTION NAME** (Figure 3-23).

1600 **Figure 3-23 IdP Connection General Info**



The screenshot shows the 'IdP Connection' configuration interface. The left sidebar has tabs for 'MAIN' (selected), 'IdP Configuration' (disabled), 'SP Configuration' (selected), 'OAuth Settings' (disabled), and 'Server Configuration' (disabled). The bottom left of the sidebar displays copyright information: 'Copyright © 2003-2016 Ping Identity Corporation. All rights reserved. Version 8.2.11'. The main content area has a title 'IdP Connection' and a sub-header 'Activation & Summary'. It contains several input fields and dropdowns:

- PARTNER'S ENTITY ID (CONNECTION ID):** idp1.spsd.mss0
- CONNECTION NAME:** idp1.spsd.mss0
- VIRTUAL SERVER IDs:** (empty input field) with an 'Add' button.
- 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 text area)
- LOGGING MODE:** A dropdown with options: NONE (unchecked), STANDARD (checked), ENHANCED (unchecked), and FULL (unchecked).

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

- 1601
 1602 e. On the **Browser SSO** tab, click **Configure Browser SSO**. The Browser SSO setup has
 1603 multiple sub-pages.
 1604 i. On the **SAML Profiles** tab, select **SP-Initiated SSO**. The **User-Session Creation**
 1605 settings are summarized on the **Summary** tab; they extract the user ID and
 1606 email address from the SAML assertion (Figure 3-24).

1607 Figure 3-24 IdP Connection–User-Session Creation

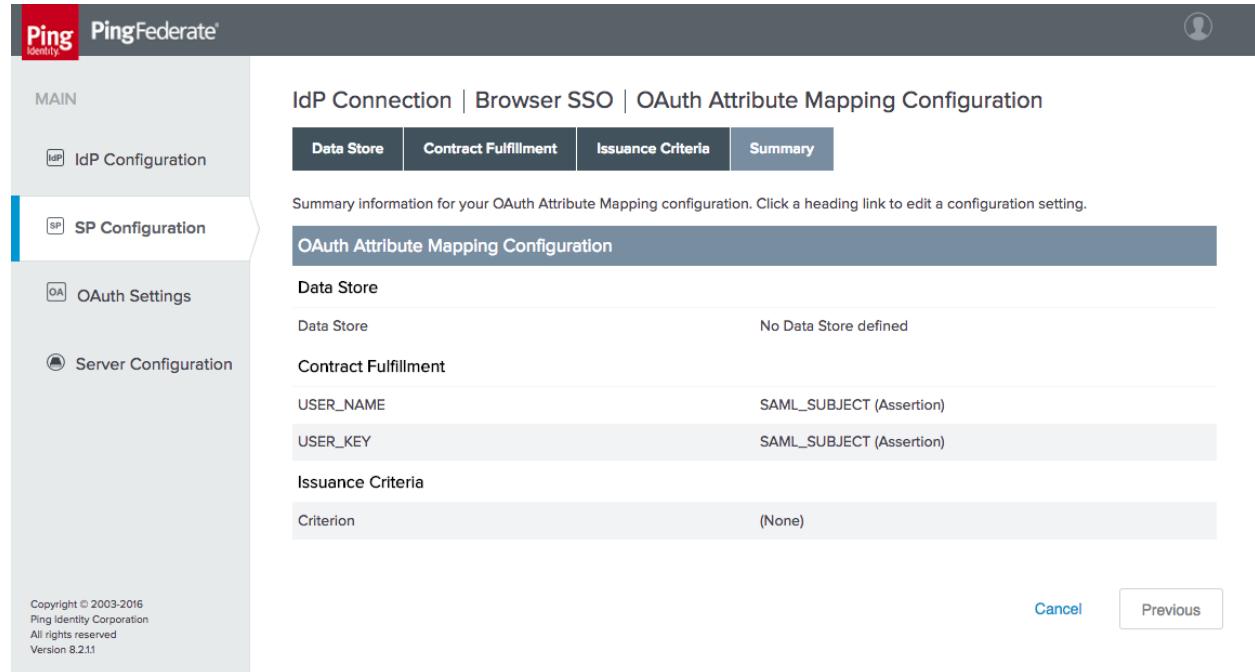
The screenshot shows the 'PingFederate' interface for 'IdP Connection | Browser SSO | User-Session Creation'. The left sidebar has sections for MAIN (IdP Configuration, SP Configuration, OAuth Settings, Server Configuration), and a copyright notice for Ping Identity Corporation. The main content area is titled 'User-Session Creation' and contains several configuration tabs: Identity Mapping, Attribute Contract, Target Session Mapping, and Summary. The 'Identity Mapping' tab is active, showing settings like 'Enable Account Mapping' (true) and 'Attribute Contract' (SAML_SUBJECT). Other tabs show configurations for Target Session Mapping, Adapter Instance, Adapter Data Store, Adapter Contract Fulfillment, Issuance Criteria, Authentication Policy Contract, Attribute Retrieval, Contract Fulfillment, and Issuance Criteria. At the bottom right are 'Cancel' and 'Previous' buttons.

Identity Mapping	
Enable Account Mapping	true
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)

1608

- 1609 ii. On the **OAuth Attribute Mapping Configuration** tab, select **MAP DIRECTLY INTO**
 1610 **PERSISTENT GRANT**. Configure the OAuth attribute mapping as shown in Figure
 1611 3-25. This maps both required values in the persistent grant context to the
 1612 SAML subject. Click **Next**, then **Next** again to skip the **Issuance Criteria** tab. Click
 1613 **Save**.

1614 **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 includes tabs for 'Data Store', 'Contract Fulfillment', 'Issuance Criteria', and 'Summary'. The 'Summary' tab is active. A note says 'Summary information for your OAuth Attribute Mapping configuration. Click a heading link to edit a configuration setting.' Below this is a section titled 'OAuth Attribute Mapping Configuration' with three sections: 'Data Store' (No Data Store defined), 'Contract Fulfillment' (USER_NAME mapped to SAML_SUBJECT (Assertion) and USER_KEY mapped to SAML_SUBJECT (Assertion)), and 'Issuance Criteria' (Criterion set to '(None)'). At the bottom right are 'Cancel' and 'Previous' buttons.

- 1615 iii. Click **Next** to proceed to the **Protocol Settings** tab. The **Protocol Settings**
 1616 configure specifics of the SAML protocol, such as the allowed bindings.
 1617 Configure these as shown in Figure 3-26. When finished, click **Save**, which will
 1618 return you to the **Browser SSO** tab of the **IdP Connection** settings.

1620 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
--------	----------

Cancel Previous

1621

- 1622 f. Click **Next**. On the **Credentials** tab, the IdP's signing certificate can be uploaded. This is
 1623 not necessary if the certificate is signed by a trusted CA.

1624 **3.4.2.2 Create Policy Contract**

- 1625 1. Create a policy contract as described in [Section 3.4.1.2](#), with the attributes **subject**, **mail**, and **uid** (Figure 3-27).

1627 **Figure 3-27 Policy Contract for SAML RP**

The screenshot shows the PingFederate administrative 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' with the following details:

Contract Info	
Contract Name	myContractName
Contract Attributes	
Attribute	subject
Attribute	mail
Attribute	uid

At the bottom right of the main area are 'Cancel' and 'Previous' buttons. The footer of the page includes copyright information: 'Copyright © 2003-2016 Ping Identity Corporation. All rights reserved. Version 8.2.1.1'.

1628

1629 **3.4.2.3 Create Policy Contract Mapping**

- 1630 1. Create an OAuth policy contract mapping for the newly created policy as described in [Section 3.4.1.3](#), mapping **USER_NAME** and **USER_KEY** to **subject** (Figure 3-28).

1632 **Figure 3-28 Contract Mapping for SAML RP**

1633

- 1634 2. To complete the setup for SAML authentication, kspd.mss0 adapter needs to be included in an
1635 authentication policy as described in [Section 3.4.4.2](#).

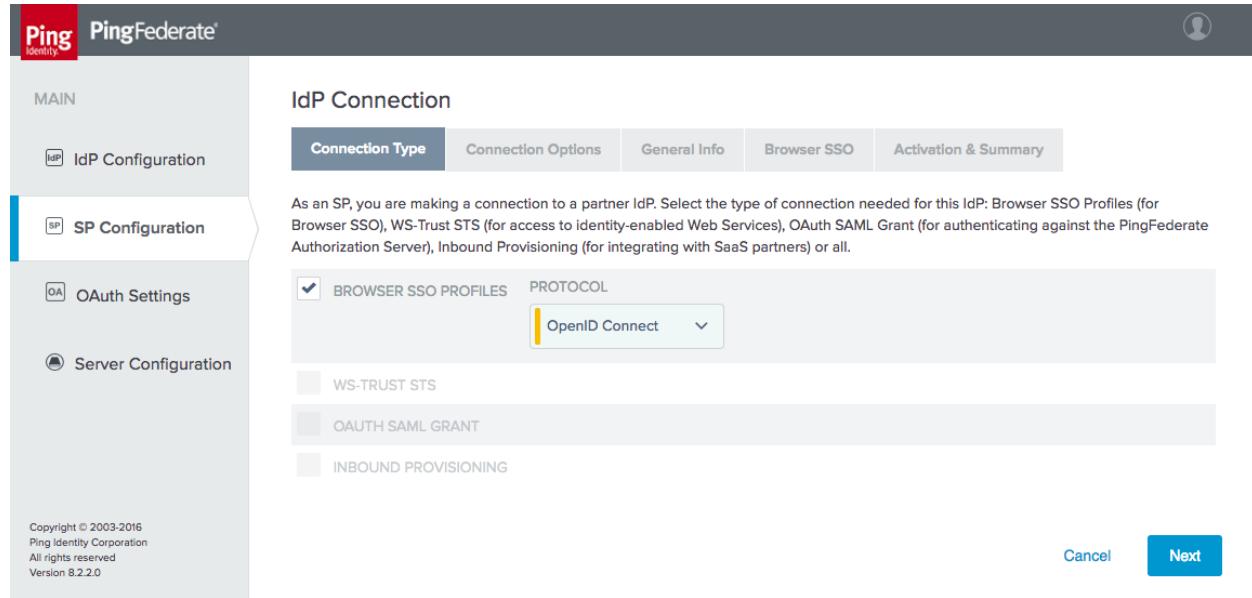
1636 [3.4.3 How to Configure OIDC Authentication](#)

1637 As with the configuration of a SAML IdP connection, integrating the AS with an OIDC IdP requires
1638 coordination between the administrators of the two systems. The administrator of the IdP must create
1639 an OIDC client registration before the connection can be configured on the AS side. The AS administrator
1640 must provide the redirect URI and, if encryption of the ID Token is desired, a public key. Unlike with
1641 SAML, there is no metadata file to exchange; however, if the IdP supports the OIDC discovery endpoint,
1642 the client can automatically obtain many of the required configuration settings from the discovery URL.

1643 This section assumes that the AS role and OIDC SP support have been enabled via **Server Settings**, as
1644 described in [Section 3.4](#). This section also uses the same authentication policy contract as the SAML
1645 authentication implementation. Create the policy contract as described in [Section 3.4.2.2](#), if it does not
1646 already exist.

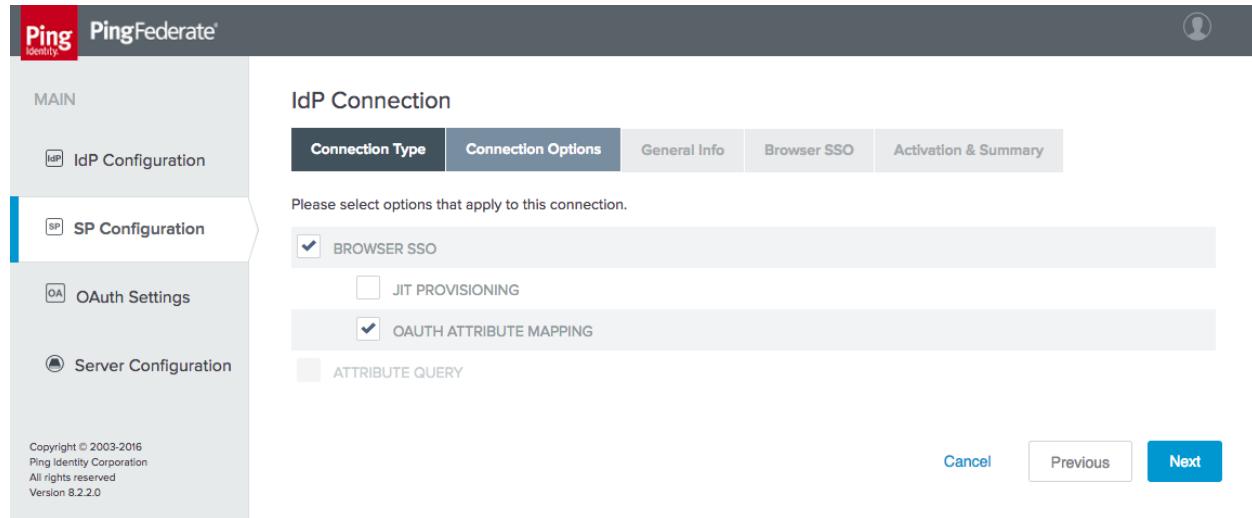
1647 [3.4.3.1 Create IdP Connection](#)

- 1648 1. On the **SP Configuration** section tab, click **Create New** under **IdP Connections**.
 - 1649 a. On the **Connection Type** tab, select **BROWSER SSO PROFILES**, and then under it, select
1650 **OpenID Connect** for the **PROTOCOL** (Figure 3-29).

1651 **Figure 3-29 IdP Connection Type**

1652

- 1653 b. On the **Connection Options** tab, select **BROWSER SSO**, and then under it, select **OAUTH ATTRIBUTE MAPPING** (Figure 3-30).
- 1654

1655 **Figure 3-30 IdP Connection Options**

1656

- 1657 c. On the **General Info** tab, enter the **ISSUER** value for the IdP (Figure 3-31). This is the **BASE URL** setting available on the **Federation Info** tab, under the **Server Configuration** section tab on the IdP. Then click **Load Metadata**, which causes the AS to query the IdP's discovery endpoint. The message "Metadata successfully loaded" should appear.
- 1658
- 1659
- 1660

1661 Provide a **CONNECTION NAME**, and enter the **CLIENT ID** and **CLIENT SECRET** provided by
 1662 the IdP administrator.

1663 **Figure 3-31 IdP Connection General Info**

MAIN

IdP Configuration

SP Configuration

OAuth Settings

Server Configuration

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

IdP Connection

General Info

ISSUER: https://op1.ipsd.msso:9031 Load Metadata Metadata successfully loaded.

CONNECTION NAME: op1.ipsd.msso

CLIENT ID: MotorolaAS

CLIENT SECRET: *****

BASE URL:

COMPANY:

CONTACT NAME:

CONTACT NUMBER:

CONTACT EMAIL:

ERROR MESSAGE: errorDetail.spSsoFailure

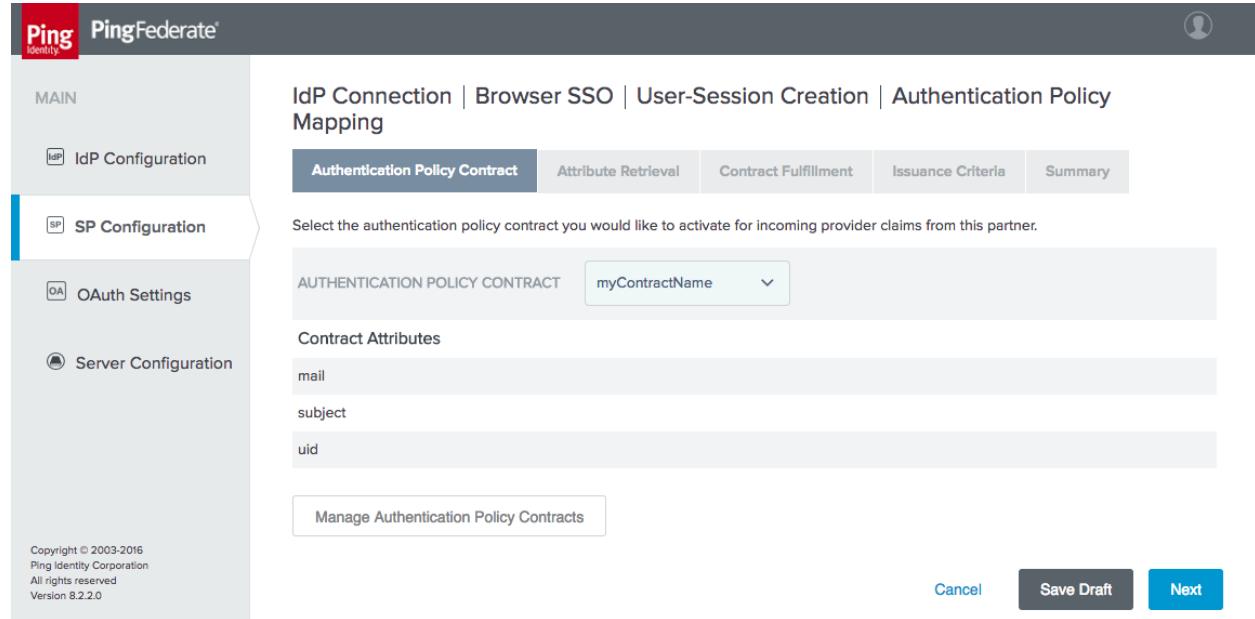
LOGGING MODE: STANDARD NONE ENHANCED FULL

Cancel Previous Next Save

- 1664
- 1665 d. On the **Browser SSO** tab, click **Configure Browser SSO**, then click **Configure User-Session Creation**. The **User-Session Creation** page will appear.
- 1666
- 1667 i. On the **Target Session Mapping** tab, click **Map New Authentication Policy**.

- 1668 ii. On the **Authentication Policy Contract** tab, select the **AUTHENTICATION POLICY**
 1669 **CONTRACT** created in [Section 3.4.2.2](#) (in the example shown in Figure 3-32, it is
 1670 called **myContractName**). If the policy contract has not been created, click
 1671 **Manage Authentication Policy Contracts**, and create it now.

1672 Figure 3-32 IdP Connection Authentication Policy Contract



- 1673 iii. On the **Attribute Retrieval** tab, leave the default setting (use only the attributes available in the provider claims).
 1674 iv. On the **Contract Fulfillment** tab, map the **mail**, **subject**, and **uid** attributes to the **email**, **sub**, and **sub** provider claims (Figure 3-33).

1678 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

1679

- 1680 v. No **Issuance Criteria** were configured; therefore, skip the **Issuance Criteria** tab.
- 1681 vi. Click **Next**, then **Done**, and then click **Done** again to exit the **User-Session**
1682 **Creation** tab.
- 1683 vii. On the **OAuth Attribute Mapping Configuration** tab, select **Map Directly into**
1684 **Persistent Grant**, and then click **Configure OAuth Attribute Mapping**.
- 1685 viii. Click **Next** to skip the Data Store tab. On the **Contract Fulfillment** tab, map both
1686 **USER_NAME** and **USER_KEY** to the **sub** provider claim (Figure 3-34).

1687 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 copyright notice for Ping Identity Corporation. The main content area is titled 'IdP Connection | Browser SSO | OAuth Attribute Mapping Configuration'. It displays two rows of attribute 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.

1688

- 1689 ix. Click **Done** to exit the **OAuth Attribute Mapping Configuration** setup. The
 1690 **Protocol Settings** should be automatically populated through the information
 1691 gathered from the discovery endpoint (Figure 3-35). If necessary, the scopes to
 1692 be requested can be customized on the **Protocol Settings** tab; in the lab, these
 1693 settings were left at the default.

1694 Figure 3-35 IdP Connection Protocol Settings

Protocol Settings

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

Overrides

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

Cancel Save Draft Previous Done

1695

- 1696 x. Click **Done** to exit the **Browser SSO** configuration setup.
- 1697 e. On the **Activation & Summary** tab, a **Redirect URI** will be generated (Figure 3-36).
1698 Provide this information to the IdP administrator, as it needs to be configured in the
1699 OpenID Client settings on the IdP side.
- 1700 i. The **Connection Status** can also be configured to **ACTIVE** or **INACTIVE** on this
1701 tab.

1702 **Figure 3-36 IdP Connection Activation and Summary**

The screenshot shows the PingFederate interface with the following details:

- MAIN Sidebar:**
 - IdP Configuration** (selected)
 - SP Configuration** (highlighted in blue)
 - OAuth Settings**
 - Server Configuration**
- Header:** PingFederate®
- Top Navigation:** IdP Connection, Connection Type, Connection Options, General Info, Browser SSO, Activation & Summary.
- Activation & Summary Section:**
 - 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
- Summary Section:**
 - IdP Connection** (selected)
 - Connection Type**
 - Connection Role: IdP
 - Browser SSO Profiles: true
 - Protocol: OpenID Connect
 - WS-Trust STS: false
 - OAuth SAML Grant: false
 - Inbound Provisioning: false
- Copyright:** Copyright © 2003-2016 Ping Identity Corporation. All rights reserved. Version 8.2.1.1

1703

1704 f. Click **Save** to complete the **IdP Connection** setup.1705 **3.4.3.2 Create the Policy Contract Mapping**1706 The same policy contract mapping created earlier for the SAML integration can also be used for OIDC
1707 integration, as the attribute names are identical. If this policy contract mapping has not already been
1708 created, refer to [Section 3.4.2.3](#) to create it.1709 **3.4.4 How to Configure the Authentication Policy**1710 **3.4.4.1 Install the Domain Selector Plugin**1711 When a single AS is integrated with multiple IdPs, it needs a means of determining which IdP can
1712 authenticate each user. In the lab build, a domain selector is used to determine whether the AS should
1713 authenticate the user locally, redirect to the SAML IdP, or redirect to the OIDC IdP. The domain selector
1714 prompts the user to enter the user's email address or domain. The specified domain is used to select
1715 which branch of the authentication policy should be applied. Upon successful authentication, the
1716 domain selector sets a cookie in the browser to record the domain selection to avoid prompting the user
1717 each time that the user authenticates.

1718 PingFederate includes sample code for a Domain Selector plugin. Before the Domain Selector can be
 1719 used in an authentication policy, it must be built. The source code for the selector is located under the
 1720 PingFederate directory, in the directory `sdk/plugin-src/authentication-selector-example`.

1721 1. Complete the following steps to build the selector:

1722 a. Edit the `build.local.properties` file in the PingFederate SDK directory to set the
 1723 target plugin as follows:

1724 `target-plugin.name=authentication-selector-example`

1725 b. Run the following commands to build and install the plugin:

1726 `$ ant clean-plugin`

1727 `$ ant jar-plugin`

1728 `$ ant deploy-plugin`

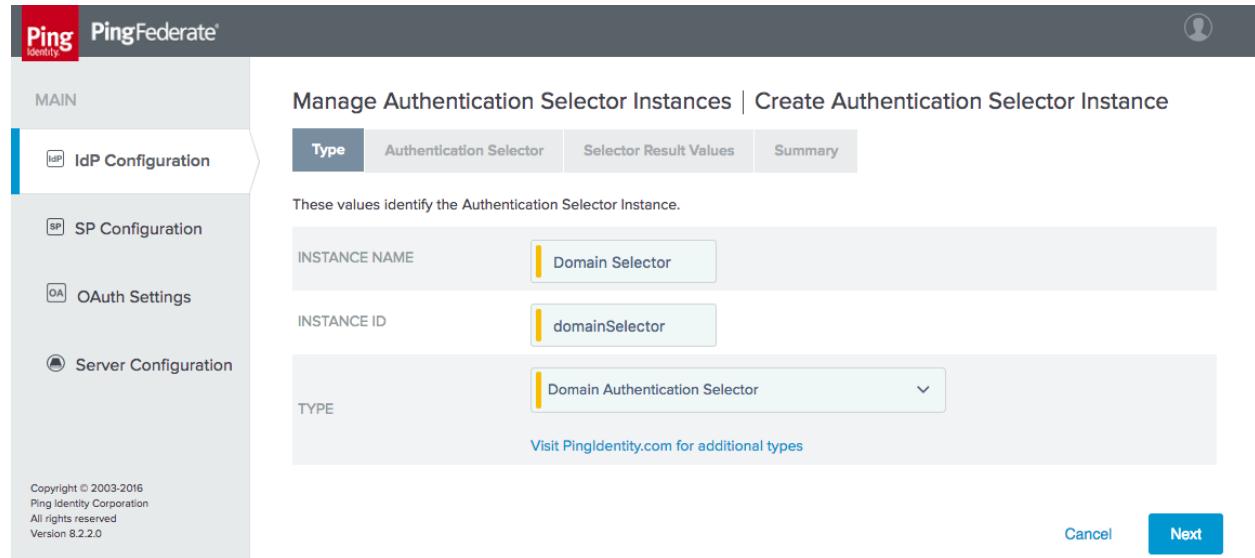
1729 `$ sudo service pingfederate restart`

1730 2. Once installed, the Domain Selector can be configured with the required values. On the **IdP**
 1731 **Configuration** section tab, click **Selectors** under **Authentication Policies**.

1732 3. Click **Create New Instance**.

1733 a. On the **Type** tab, provide a meaningful name and ID for the selector instance (Figure
 1734 3-37). For the **TYPE**, select **Domain Authentication Selector**.

1735 **Figure 3-37 Authentication Selector Instance**



1736

- 1737 b. The next tab, **Authentication Selector**, prompts for the HyperText Markup Language
 1738 (HTML) template for the page that will prompt the user to enter the domain or email
 1739 address (Figure 3-38). The default value will use the template delivered with the
 1740 adapter; if desired, a custom template can be used instead to modify the appearance of
 1741 the page. Provide a cookie name, which will be used to persist the domain selection.
 1742 Finally, the age of the cookie can be modified. By default, users will be prompted again
 1743 to enter their domain after 30 days.

1744 **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.

- 1745 c. On the **Selector Result Values** tab, specify the expected domain values (Figure 3-39).
 1746 When the domain selector is used in an access policy, different policy branches will be
 1747 created for each of these values. In this case, if the domain is *motorolasolutions.com*,
 1748 the user will be authenticated locally; if it is *lpsd.mss0* or *spsd.mss0*, the user will be
 1749 redirected to the corresponding IdP to authenticate.
 1750

1751 **Figure 3-39 Selector Result Values**

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

1752

1753 d. Click **Done**, and then click **Save** to complete the selector configuration.

3.4.4.2 Define the Authentication Policy

1755 1. On the IdP Configuration page, click **Policies** under **Authentication Policies**.

1756 a. Select the three checkboxes at the top of the **Manage Authentication Policies** page,
1757 which are shown in Figure 3-40.

1758 **Figure 3-40 Policy Settings**

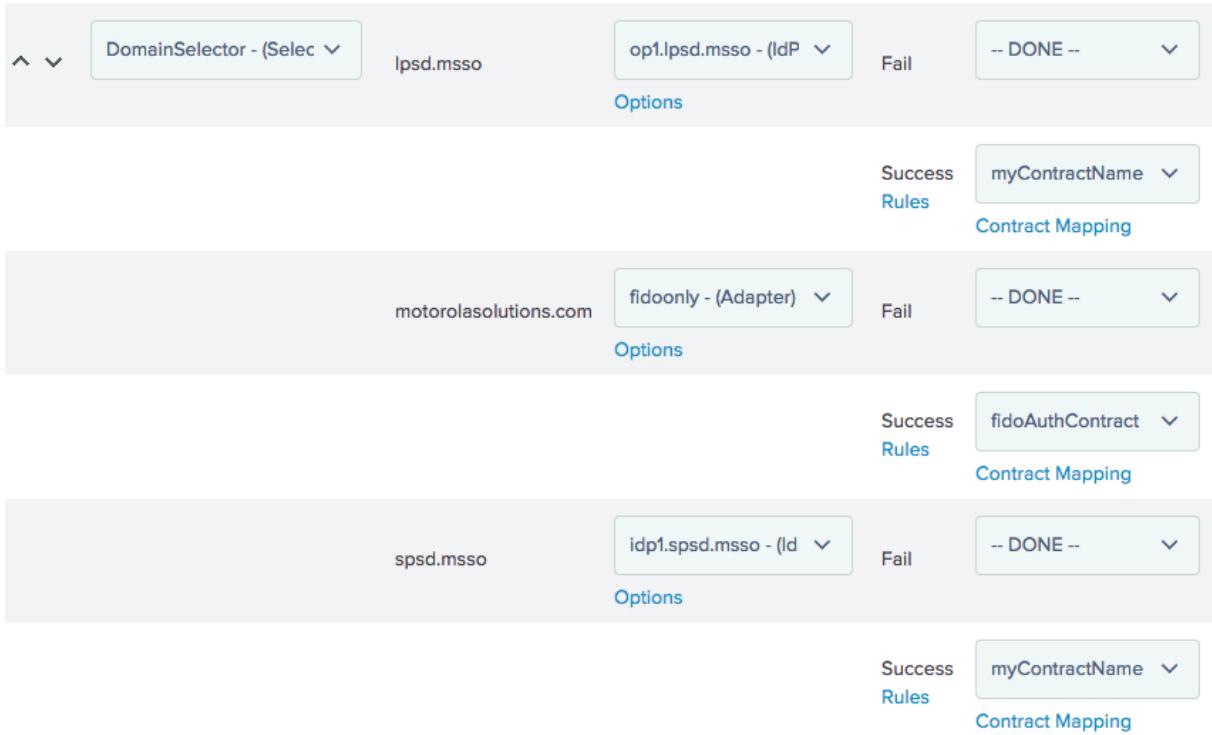
- ENABLE IDP AUTHENTICATION POLICIES
- ENABLE SP AUTHENTICATION POLICIES
- FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

1759

1760 b. Select the **Domain Selector** as the first element in the policy (Figure 3-41). This will
1761 create policy branches for the three values defined for the policy selector.

1762 i. Select the corresponding authentication mechanism for each domain. The
1763 example shown in Figure 3-41 uses the IdP connections for the **lpsd.msso** and
1764 **spsd.msso**, as well as the “fidoonly” adapter for local authentication of users in
1765 the **motorolasolutions.com** domain.

1766 Figure 3-41 Authentication Policy



1767

- 1768 ii. There is no need to specify **Options** or **Success Rules**. For the two IdP
 1769 connections, apply the **myContractName** policy contract upon success, with the
 1770 contract mapping configured as shown in Figure 3-42.

1771 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

1772

- 1773 c. For the “fidoonly” adapter, apply the **fidoAuthContract** with the contract mapping shown in Figure 3-43.
- 1774

1775 **Figure 3-43 Policy Contract Mapping for Local Authentication**

The screenshot shows the 'PingFederate' interface with a navigation sidebar on the left containing 'MAIN', 'IdP Configuration', 'SP Configuration' (which is selected), 'OAuth Settings', and 'Server Configuration'. The main content area is titled 'Manage Authentication Policies | Authentication Policy Contract Mapping' and includes tabs for 'Attribute Sources & User Lookup', 'Contract Fulfillment', 'Issuance Criteria', and 'Summary'. The 'Contract Fulfillment' tab is active. It displays a summary of 'Authentication Policy Contract Mapping' with sections for 'Attribute Sources & User Lookup', 'Contract Fulfillment', and 'Issuance Criteria'. Under 'Contract Fulfillment', there are entries for 'subject' and 'username'. Under 'Issuance Criteria', there is an entry for 'Criterion'. At the bottom right are 'Cancel' and 'Previous' buttons.

1776

1777 This completes the configuration of the AS.

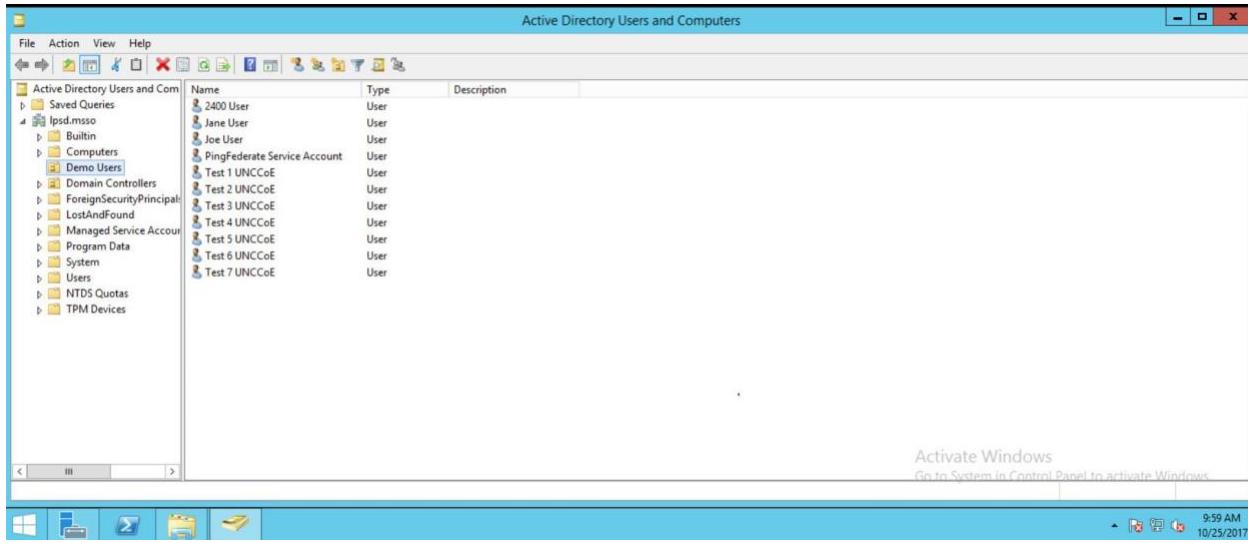
4 How to Install and Configure the Identity Providers

1779 PingFederate 8.3.2.0 was used for the SAML and OIDC IdP installs. The system requirements and
 1780 installation process for PingFederate are identical to the OAuth AS installation documentation in
 1781 [Section 3.1](#) and [Section 3.2](#). The IdP configuration sections pick up the installation process after the
 1782 software has been installed, at the selection of roles and protocols.

4.1 How to Configure the User Store

1784 Each IdP uses its own AD forest as a user store. AD was chosen due to its widespread use across many
 1785 organizations. For the purposes of this project, any LDAP directory could have served the same purpose,
 1786 but in a typical organization, AD would be used for other functions, such as workstation login and
 1787 authorization to applications, shared drives, printers, and other services. The **Active Directory Users and**
 1788 **Computers** console (Figure 4-1) was used to create user accounts and set attributes.

1789 Figure 4-1 Active Directory Users and Computers

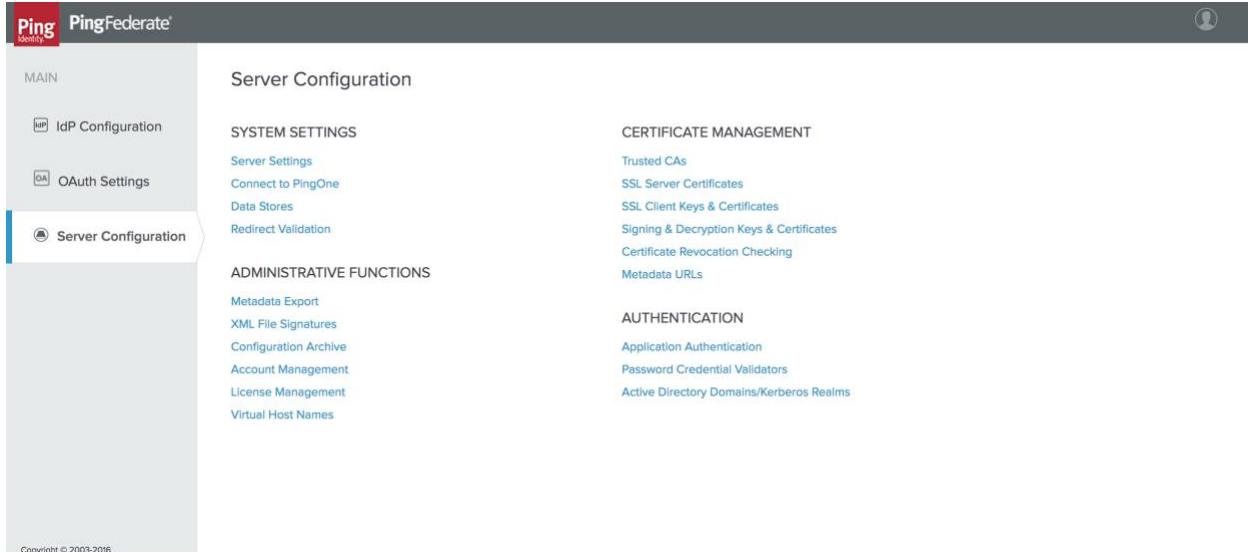


1790

1791 In addition to the user accounts that log in to the lab applications, a service account must be created to
 1792 enable the IdP to access and query the AD. This user's LDAP Distinguished Name (DN) and password (in
 1793 the example shown in Figure 4-1) are used in the PingFederate directory integration described below.

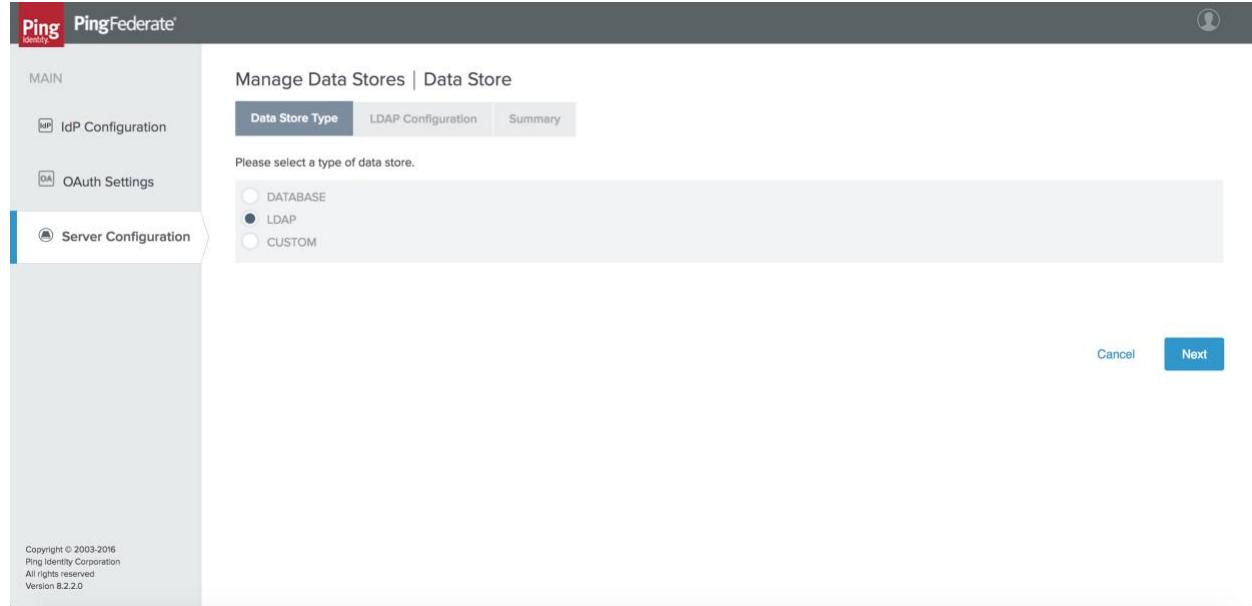
1794 The procedure for connecting a PingFederate IdP to an LDAP directory is the same for a SAML or OIDC
 1795 IdP. Documentation is provided at
 1796 https://documentation.pingidentity.com/pingfederate/pf82/index.shtml#concept_configuringLdapConnection.html.
 1797

1798 1. To start the process, click the **Server Configuration** section tab on the left side of the
 1799 PingFederate administrative console. The screen shown in Figure 4-2 will appear.

1800 **Figure 4-2 Server Configuration**

1801

- 1802 2. Click **Data Stores** under **SYSTEM SETTINGS**.
- 1803 3. On the next screen, click **Add New Data Store**.
- 1804 a. The screen shown in Figure 4-3 will appear. On the **Data Store Type** tab, select **LDAP** for
1805 the data store type.
- 1806 i. Click **Next**.

1807 **Figure 4-3 Data Store Type**

1808

- 1809 b. On the **LDAP Configuration** tab, enter the connection parameters for your AD or LDAP
 1810 environment (Figure 4-4). Some notes on the fields on this tab are provided below. Click
 1811 **Save** to exit the LDAP configuration screen once the required settings have been
 1812 entered.
- 1813 ■ **HOST NAME(S)**: Enter the Fully Qualified Domain Name (FQDN) or the complete
 1814 Internet Protocol (IP) address of an AD domain controller. A port number can be
 1815 specified if AD is running on non-standard ports.
- 1816 ■ **LDAP TYPE**: This is the LDAP server in use—AD in this case.
- 1817 ■ **BIND ANONYMOUSLY**: For AD environments, allowing anonymous BIND
 1818 (Berkeley Internet Name Domain) is not recommended.
- 1819 ■ **USER DN**: This is the Distinguished Name of the PingFederate user account
 1820 created in AD; in this build architecture, this account is used only for querying
 1821 AD, so it does not require any special privileges.
- 1822 ■ **PASSWORD**: This is the password for the PingFederate AD user.
- 1823 ■ **USE LDAPS**: This can be enabled if AD is configured to serve LDAP over TLS.
- 1824 ■ **MASK VALUES IN LOG**: This prevents attributes returned from this data source
 1825 from being exposed in server logs.

1826 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

1827

1828

4.2 How to Install and Configure the SAML Identity Provider

- 1829 1. On the **Server Configuration** screen, click **Server Settings**.
- 1830 a. On the **Roles & Protocols** tab, enable roles and protocols to configure the server as a
1831 SAML IdP (Figure 4-5).

1832 Figure 4-5 Server Roles for SAML IdP

The screenshot shows the 'PingFederate' interface with the 'Server Configuration' tab selected. The main menu on the left includes 'MAIN', 'IdP Configuration', 'SP Configuration', and 'Server Configuration'. The 'Server Settings' tab is active, showing tabs for 'System Administration', 'System Info', 'Runtime Notifications', 'Runtime Reporting', and 'Account Management'. Under 'System Administration', the 'Roles & Protocols' tab is selected. The page displays configuration options for the Identity Provider (IdP) role, including checkboxes for enabling various protocols and service provider (SP) roles.

Select the role(s) and protocol(s) that you intend to use with your federation partners.	
<input type="checkbox"/> ENABLE OAUTH 2.0 AUTHORIZATION SERVER (AS) ROLE	
<input checked="" type="checkbox"/> ENABLE IDENTITY PROVIDER (IDP) ROLE AND SUPPORT THE FOLLOWING:	
<input checked="" type="checkbox"/> SAML 2.0	
<input type="checkbox"/> AUTO-CONNECT PROFILE	
<input type="checkbox"/> SAML 1.1	
<input type="checkbox"/> SAML 1.0	
<input type="checkbox"/> WS-FEDERATION	
<input type="checkbox"/> OUTBOUND PROVISIONING	
<input type="checkbox"/> WS-TRUST	
<input checked="" type="checkbox"/> ENABLE SERVICE PROVIDER (SP) ROLE AND SUPPORT THE FOLLOWING:	
<input checked="" type="checkbox"/> SAML 2.0	
<input type="checkbox"/> AUTO-CONNECT PROFILE	
<input type="checkbox"/> ATTRIBUTE REQUESTER MAPPING FOR X.509 ATTRIBUTE SHARING PROFILE (XASP)	
<input type="checkbox"/> SAML 1.1	
<input type="checkbox"/> SAML 1.0	
<input type="checkbox"/> WS-FEDERATION	
<input type="checkbox"/> WS-TRUST	
<input type="checkbox"/> INBOUND PROVISIONING	
<input type="checkbox"/> OPENID CONNECT	
<input type="checkbox"/> ENABLE IDP DISCOVERY ROLE (SAML 2.0 ONLY)	

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

1833

- 1834 b. On the **Federation Info** tab, specify the **BASE URL** and **SAML 2.0 ENTITY ID** of the IdP (Figure 4-6). The **BASE URL** should be a URL resolvable by your mobile clients. The **ENTITY ID** should be a meaningful name that is unique among federation partners; in this case, the FQDN of the server is used.

1838 **Figure 4-6 SAML IdP Federation Info**

The screenshot shows the PingFederate interface with the title "PingFederate". On the left, a sidebar titled "MAIN" has three tabs: "IdP Configuration", "SP Configuration", and "Server Configuration", with "Server Configuration" being the active tab. 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", with "Federation Info" being the active tab. Below this is another navigation bar with tabs: "Roles & Protocols", "Federation Info", "System Options", "Metadata Signing", "Metadata Lifetime", and "Summary", with "Federation Info" being the active tab. 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 shown: "BASE URL" containing "https://idp1.spsd.mss:9031" and "SAML 2.0 ENTITY ID" containing "idp1.spsd.mss". At the bottom right are buttons for "Cancel", "Previous", "Next", and "Save". A copyright notice at the bottom left reads: "Copyright © 2003-2017 Ping Identity Corporation. All rights reserved. Version 8.3.20".

1839

1840 **4.2.1 Configuring Authentication to the IdP**1841 This example configures an authentication policy that requires the user to authenticate with username
1842 and password and then with a FIDO U2F token.1843 **4.2.1.1 Configure the Password Validator**

- 1844 1. On the **Server Configuration** section tab, click **Password Credential Validators** under
1845 **Authentication**.
- 1846 2. Click **Create New Instance**.
 - 1847 a. On the **Type** tab, for the **TYPE**, choose **LDAP Username Password Credential Validator**
1848 (**Figure 4-7**). This example will authenticate AD usernames and passwords by using the
1849 AD data store defined in [Section 4.1](#).

1850 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

1851

- 1852 b. On the **Instance Configuration** tab, specify the parameters for searching the LDAP directory for user accounts (Figure 4-8). Select the data store created in [Section 4.1](#), and enter the appropriate search base and filter. This example will search for a *sAMAccountName* matching the username entered on the login form.
- 1853
- 1854
- 1855

1856 Figure 4-8 Credential Validator Configuration

Manage Credential Validator Instances | Create Credential Validator Instance

Type	Instance Configuration	Extended Contract	Summary		
Complete the configuration necessary for this Password Credential Validator to check username/password pairs. This configuration was designed into, and is specific to, the selected Credential Validator plug-in.					
This password credential validator provides a means of verifying credentials stored in a directory server via the LDAP protocol. Additional user attributes from the directory can also be returned by this PCV by adding the desired attribute names to the Extended Contract.					
AUTHENTICATION ERROR OVERRIDES (A table of LDAP authentication error codes and customized matching expressions that will match the error code to an LDAP error message. These entries override the default individual mappings of messages to codes. Use the localization features to customize the error messages displayed to end users.)					
MATCH EXPRESSION (The expression matched against the LDAP error message returned by the server.)		ERROR	Action		
Add a new row to 'Authentication Error Overrides'					
Field Name	Field Value	Description			
LDAP DATASTORE	dc1.spsd.msso	Select the LDAP Datastore.			
SEARCH BASE	OU=Demo Users,DC=spsd,DC=msso	The location in the directory from which the LDAP search begins.			
SEARCH FILTER	sAMAccountName=\${username}	You may use \${username} as part of the query. Example (for Active Directory): sAMAccountName=\${username}.			
SCOPE OF SEARCH	<input type="radio"/> One Level <input checked="" type="radio"/> Subtree				
CASE-SENSITIVE MATCHING	<input checked="" type="checkbox"/>	Allows case-sensitive expression and LDAP error matching.			
Manage Data Stores Show Advanced Fields		Cancel	Previous	Next	Done

1857

- 1858 c. The **Extended Contract** tab enables the retrieval of additional attributes from the LDAP
 1859 server, which can be used in assertions to RPs (Figure 4-9). The example shown in
 1860 Figure 4-9 adds several AD attributes to the contract.

1861 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
<input type="button"/>	Add

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

Cancel Previous Next Done

1862

- 1863 d. Finally, the **Summary** tab shows all of the values for the configured validator
1864 (Figure 4-10).

1865 **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 is selected.
- Type** tab is 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.

1866

1867 e. Click **Done**, and then click **Save** to complete the setup of the password validator.1868 *4.2.1.2 Configure the HTML Form Adapter*1869 1. On the **IdP Configuration** section tab, click **Adapters**.1870 2. Click **Create New Instance**.1871 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).
1872

1873 Figure 4-11 HTML Form Adapter Instance

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	HTML Form Adapter
INSTANCE ID	HTMLFormAdapter
TYPE	HTML Form IdP Adapter
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

1874

- 1875 b. On the **IdP Adapter** tab, add the **Password Validator** instance created in the previous
 1876 section (Figure 4-12). This tab provides several options for customizing the login page
 1877 and supporting password resets and password recovery that would be relevant to a
 1878 Production deployment. In the lab, password resets were not supported, and these
 1879 fields were left at their default values.

1880 Figure 4-12 Form Adapter Settings

PingFederate®

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	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	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	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		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

- 1882 c. On the **Extended Contract** tab, the same attributes returned from AD by the Password
 1883 Validator are added to the adapter contract, to make them available for further use by
 1884 the IdP (Figure 4-13).

1885 **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	
givenName	Edit Delete
mail	Edit Delete
memberOf	Edit Delete
objectGUID	Edit Delete
sn	Edit Delete
userPrincipalName	Edit Delete

- 1886
- 1887 d. On the **Adapter Attributes** tab, select the **Pseudonym** checkbox for the **username** attribute.
- 1888
- 1889 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.
- 1890
- 1891

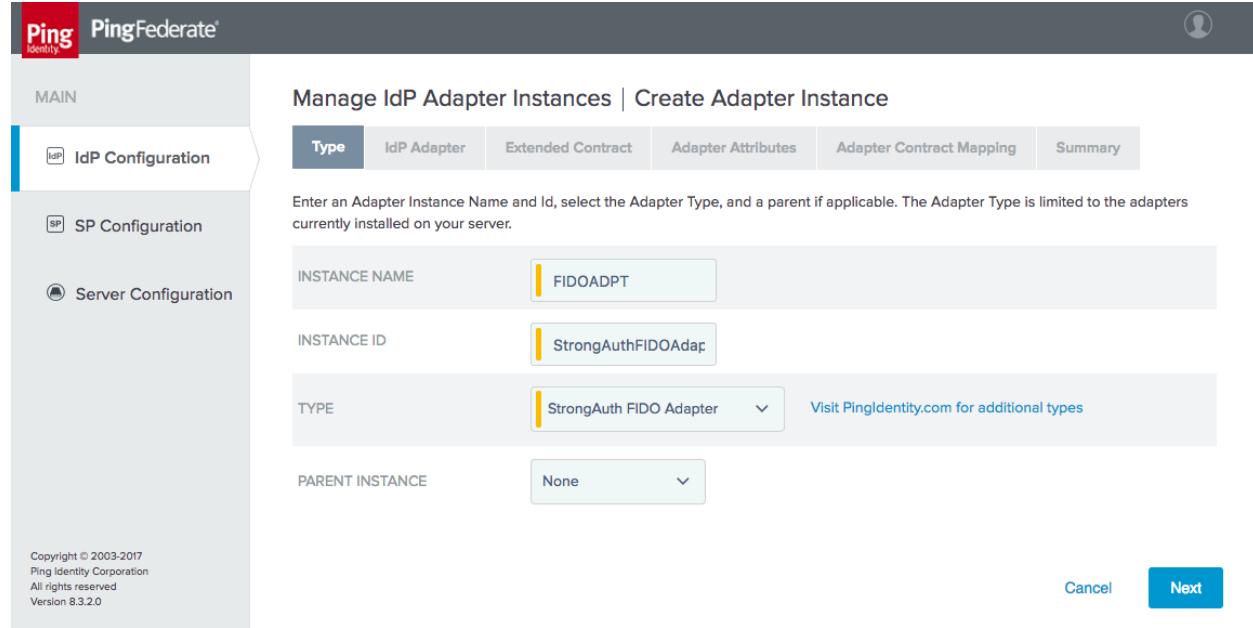
1892 *4.2.1.3 Configure the FIDO U2F Adapter*

1893 Before this step can be completed, the FIDO U2F server, StrongKey CryptoEngine (SKCE), must be
 1894 installed and configured, and the StrongKey U2F adapter for PingFederate must be installed on the IdP.
 1895 See [Section 6](#) for details on completing these tasks.

- 1896 1. On the **IdP Configuration** section tab, click **Adapters**.
- 1897 2. Click **Create New Instance**.

- 1898 a. Enter meaningful values for **INSTANCE NAME** and **INSTANCE ID**. For the **TYPE**, select
 1899 “StrongAuth FIDO Adapter.” Click **Next**.

1900 **Figure 4-14 Create U2F Adapter Instance**



The screenshot shows the 'Manage IdP Adapter Instances | Create Adapter Instance' page. The left sidebar has tabs for MAIN, IdP Configuration (selected), SP Configuration, and Server Configuration. The main area has tabs for Type (selected), IdP Adapter, Extended Contract, Adapter Attributes, Adapter Contract Mapping, and Summary. 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 'FIDOADPT'. The 'INSTANCE ID' field contains 'StrongAuthFIDOAdap'. The 'TYPE' dropdown is set to 'StrongAuth FIDO Adapter' with a link to 'Visit Pingidentity.com for additional types'. The 'PARENT INSTANCE' dropdown is set to 'None'. At the bottom right are 'Cancel' and 'Next' buttons.

- 1901
 1902 b. On the **IdP Adapter** tab, keep the default value of the **HTML FORM TEMPLATE NAME** to
 1903 use the template that is provided with the StrongKey U2F plugin, or specify a custom
 1904 template if desired to change the design of the user interface (Figure 4-15). The **FIDO**
 1905 **SERVER URL**, **DOMAIN ID**, **SKCE SERVICE USER**, and **SKCE SERVICE USER PASSWORD** are
 1906 determined in the setup of the SKCE; refer to [Section 6](#) for details.

1907 **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

1908

- 1909 c. There is no need to extend the contract for the U2F adapter; therefore, skip the
1910 **Extended Contract** tab.
- 1911 d. On the **Adapter Attributes** tab, select the **Pseudonym** checkbox for the **username**
1912 attribute.
- 1913 e. There is also no need for an **Adapter Contract Mapping**; therefore, skip the **Adapter**
1914 **Contract Mapping** tab.
- 1915 f. Click **Done**, and then click **Save**.

1916 **4.2.1.4 Configure the Authentication Policies**

- 1917 1. On the **IdP Configuration** page, click **Policies**.
- 1918 a. Under **Manage Authentication Policies**, click the **ENABLE IDP AUTHENTICATION**
1919 **POLICIES** checkbox, and create a policy that starts with the **HTML Form Adapter** action
1920 (Figure 4-16).

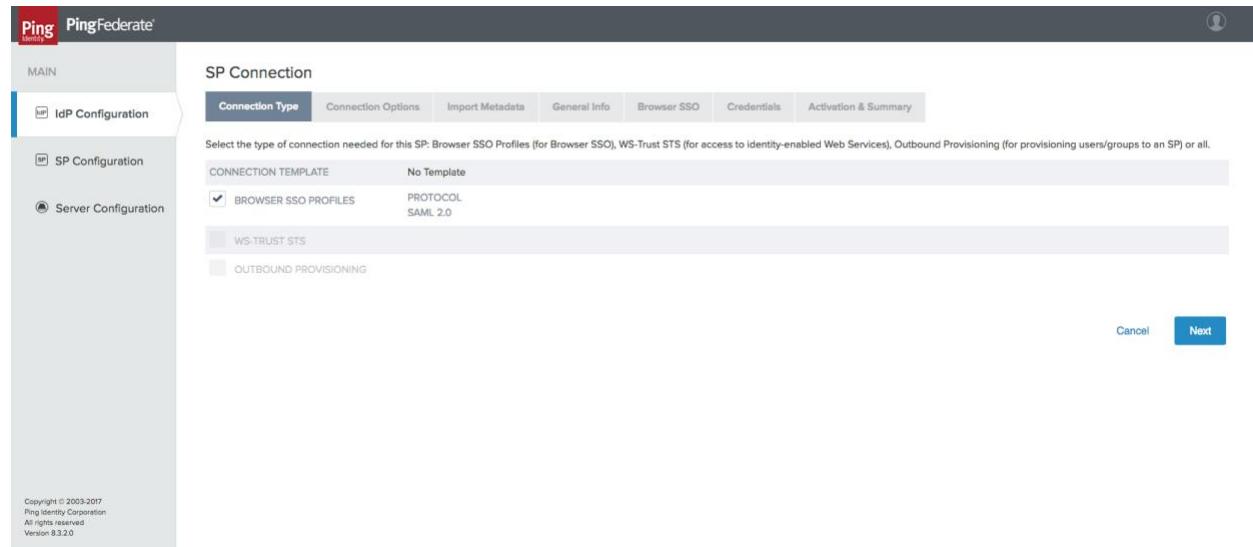
- 1921 i. On the **Success** branch, add the FIDO U2F adapter (**FIDOADPT**) for the **Action**.
 1922 ii. Click **Save**.

1923 **Figure 4-16 IdP Authentication Policy**

- 1924
- 1925 **4.2.2 Configure the SP Connection**
 1926 Each RP that will receive authentication assertions from the IdP must be configured as an SP connection.
 1927 As explained in [Section 3.4.2.1](#), this activity requires coordination between the administrators of the IdP
 1928 and the RP to provide the necessary details to configure the connection. Exchanging metadata files can
 1929 help automate some of the configuration process.
- 1930 This section documents the configuration for the SP connection between the SAML IdP in the NCCoE lab
 1931 and the OAuth AS in the Motorola Solutions cloud instance.

- 1932 1. To create a new SP connection, click the **IdP Configuration** section tab, and then click **Create New** under **SP Connections**.
- 1934 a. On the **Connection Type** tab, select **BROWSER SSO PROFILES**, and select the **SAML 2.0** protocol (Figure 4-17). In this case, SAML 2.0 is pre-selected because no other protocols are enabled on this IdP.

1937 **Figure 4-17 SP Connection Type**



- 1938 b. On the **Connection Options** tab, only **BROWSER SSO** needs to be selected.
- 1939 c. If metadata for the SP is available, it can be imported on the **Import Metadata** tab. This metadata can be specified in the form of a file upload or URL.
- 1940 d. On the **General Info** tab, enter the **PARTNER'S ENTITY ID (CONNECTION ID)** (Figure 4-18); this must match the **ENTITY ID** configured on the **Federation Info** tab in the **Server Configuration** of the SP. The SP's **BASE URL** should also be added on this **General Info** tab.

1946 **Figure 4-18 SP Connection General Info**

The screenshot shows the PingFederate interface with the following details:

- Left Sidebar:** MAIN, IdP Configuration (selected), SP Configuration, Server Configuration.
- Top Bar:** PingFederate, User icon.
- Page Title:** SP Connection.
- Tab Bar:** Connection Type, Connection Options, Metadata URL, General Info (selected), Browser SSO, Credentials.
- Activation & Summary:** This section contains a note about connection identifiers and virtual server IDs.
- Form Fields (General Info):**
 - PARTNER'S ENTITY ID (CONNECTION ID): ctoPingFed_entityID
 - CONNECTION NAME: ctoPingFed_entityID
 - VIRTUAL SERVER IDS: Input field with an 'Add' button.
 - BASE URL: https://idm.sandbox.motorolasolutions.co
 - COMPANY: Input field.
 - CONTACT NAME: Input field.
 - CONTACT NUMBER: Input field.
 - CONTACT EMAIL: Input field.
 - APPLICATION NAME: Input field.
 - APPLICATION ICON URL: Input field.
 - LOGGING MODE: Radio buttons for NONE (selected), STANDARD, ENHANCED, and FULL.
- Buttons:** Cancel, Previous, Next, Save.

1947

1948 e. On the **Browser SSO** tab, click **Configure Browser SSO**. This opens another multi-tabbed configuration screen.
19491950 i. On the **SAML Profiles** tab, different SSO and Single Log-Out (SLO) profiles can be
1951 enabled (Figure 4-19). Only **SP-INITIATED SSO** is demonstrated in this lab build.

1952 Figure 4-19 SP Browser SSO Profiles

Ping Federate

MAIN

IdP IdP Configuration

SP SP Configuration

Server Server Configuration

SP Connection | Browser SSO

SAML Profiles Assertion Lifetime Assertion Creation Protocol Settings Summary

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

Cancel Next Done Save

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

1953

- 1954 ii. On the **Assertion Lifetime** tab, time intervals during which SPs should consider assertions valid can be configured in minutes before and after assertion creation. In the lab, these were both set to the default of five minutes.

1955

1956

1957 iii. On the **Assertion Creation** tab, click **Configure Assertion Creation**. This opens a new multi-tabbed configuration screen.

1958

1959 1) On the **Identity Mapping** tab, select the **STANDARD** mapping (Figure 4-20).
1960 The other options are more suitable for situations where identifiers are
1961 sensitive or where there are privacy concerns over the tracking of users.

1962 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

1963

1964

1965

1966

1967 Figure 4-21 Assertion Attribute Contract

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

Attribute Contract	Subject Name Format
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

Add

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

Cancel Previous Next Done Save

1968

- 1969 3) On the **Authentication Source Mapping** tab, attributes provided by
 1970 authentication adapters and policy contracts can be mapped to the
 1971 assertion attribute contract, identifying which data will be used to
 1972 populate the assertions. The FIDO U2F adapter and the HTML Form
 1973 Adapter should appear under **Adapter Instance Name**. Select the HTML
 1974 Form Adapter, as it can provide the needed attributes from LDAP via the
 1975 Password Validator and the AD data store connection. This brings up
 1976 another multi-tabbed configuration screen.
- 1977 a) The **Adapter Instance** tab shows the attributes that are returned by
 1978 the selected adapter. Click **Next**.
- 1979 b) The **Mapping Method** tab provides options to query additional data
 1980 stores to build the assertions, but in this case, all of the required
 1981 attributes are provided by the HTML Form Adapter. Select **USE ONLY**
 1982 **THE ADAPTER CONTRACT VALUES IN THE SAML ASSERTION**.
- 1983 c) On the **Attribute Contract Fulfillment** tab, map the **SAML SUBJECT**,
 1984 **mail**, and **uid** attributes to the **username**, **mail**, and
 1985 **userPrincipalName** adapter values (Figure 4-22).

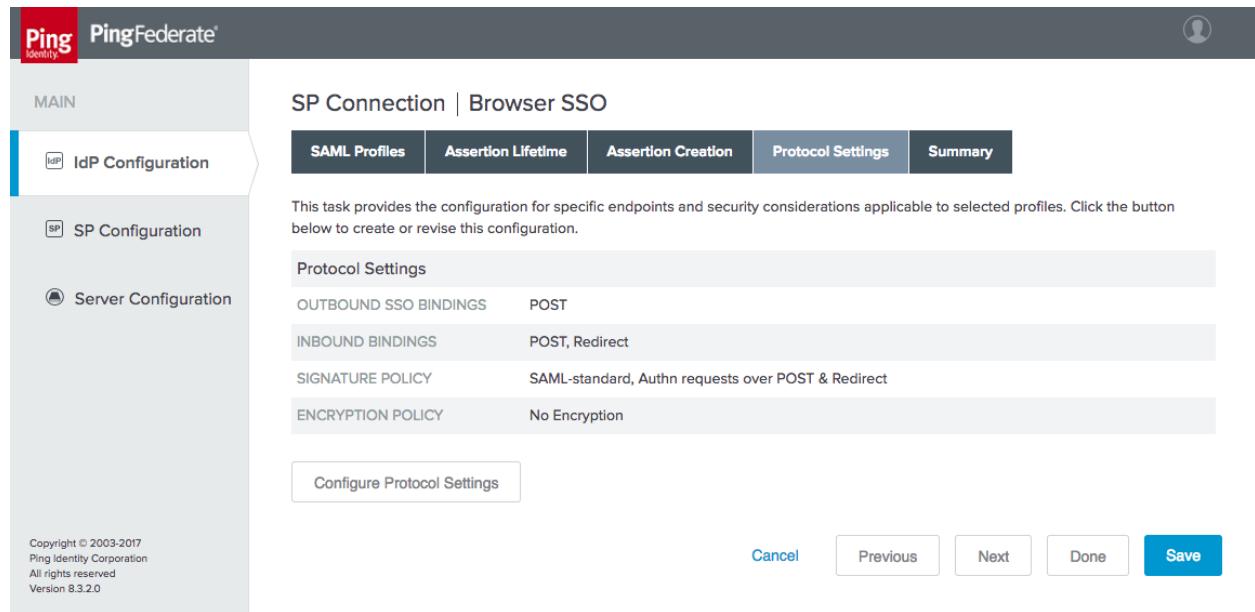
1986 Figure 4-22 Assertion Attribute Contract Fulfillment

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

1987

- 1988 d) No **Issuance Criteria** are required; therefore, skip the **Issuance Criteria**
 1989 tab.
- 1990 e) Click **Done** to exit the IdP Adapter Mapping.
- 1991 4) Click **Done** to exit the Assertion Creation.
- 1992 iv. On the **Protocol Settings** tab, options such as additional SAML bindings,
 1993 signature policy details, and assertion encryption policies can be specified
 1994 (Figure 4-23). For the lab build, these values were left at their default settings.

1995 Figure 4-23 Browser SSO Protocol Settings



- 1996 v. Click **Done** to exit Browser SSO.
- 1997 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.
- 1998 g. On the **Activation & Summary** tab, the connection status can be set to **ACTIVE**. All configured settings for the SP connection are also displayed for verification.
- 1999 h. Click **Save** to complete the SP connection configuration.
- 2000 This completes the configuration of the SAML IdP.

2006 4.3 How to Install and Configure the OIDC Identity Provider

- 2007 1. On the **Server Configuration** section tab, click **Server Settings**.
- 2008 a. On the **Roles & Protocols** tab, enable the roles and protocols as shown in Figure 4-24.
- 2009 Although the OIDC IdP does not actually use the SAML protocol, some required configuration settings are unavailable if the IdP role is not enabled.
- 2010

2011 **Figure 4-24** OIDC IdP Roles

The screenshot shows the 'PingFederate' interface. The left sidebar has tabs for 'MAIN' (selected), 'IdP Configuration', 'OAuth Settings', and 'Server Configuration' (selected). The main content area is titled 'Server Settings' and contains a 'System Administration' tab bar with 'Roles & Protocols' (selected), 'System Info', 'Runtime Notifications', 'Runtime Reporting', and 'Account Management'. Below the tabs, a section titled 'Select the role(s) and protocol(s) that you intend to use with your federation partners.' contains several checkboxes. Under 'ENABLE OAUTH 2.0 AUTHORIZATION SERVER (AS) ROLE', 'OPENID CONNECT' is checked. Under 'ENABLE IDENTITY PROVIDER (IDP) ROLE AND SUPPORT THE FOLLOWING:', 'SAML 2.0' is checked, while 'AUTO-CONNECT PROFILE', 'SAML 1.1', 'SAML 1.0', 'WS-FEDERATION', 'OUTBOUND PROVISIONING', and 'WS-TRUST' are unchecked. There are also sections for 'ENABLE SERVICE PROVIDER (SP) ROLE AND SUPPORT THE FOLLOWING:' and 'ENABLE IDP DISCOVERY ROLE (SAML 2.0 ONLY)'. At the bottom right are 'Cancel', 'Previous', 'Next', and a blue 'Save' button.

- 2012
- 2013 b. On the **Federation Info** tab, specify the **BASE URL** and **SAML 2.0 ENTITY ID**. The **BASE URL** must be a URL that is exposed to clients.
- 2014
- 2015 2. On the **OAuth Settings** section tab, click **Authorization Server Settings** to configure general OAuth and OIDC parameters. The OIDC IdP's settings on this page are identical to those for the OAuth AS; refer to [Section 3.3](#) for notes on these settings.
- 2016
- 2017

2018 3. On the **OAuth Settings** section tab, click **Scope Management**.

2019 a. Add the scopes defined in the OpenID Connect Core specification [\[25\]](#):

2020 ▪ openid

2021 ▪ profile

2022 ▪ email

2023 ▪ address

2024 ▪ phone

2025 [**4.3.1 Configuring Authentication to the OIDC IdP**](#)

2026 In the lab architecture, the OIDC IdP supports FIDO UAF authentication through integration with the
2027 NNAS and the Nok Nok Labs Gateway, using the Nok Nok FIDO UAF adapter for PingFederate.

2028 Configuring UAF authentication to the OIDC IdP cannot be completed until the Nok Nok Labs servers are
2029 available and the UAF plugin has been installed on the IdP server as specified in [Section 5](#).

2030 [**4.3.1.1 Configure the FIDO UAF Plugin**](#)

2031 The steps to configure the FIDO UAF plugin for the OIDC IdP are identical to those documented in
2032 [Section 3.4.1.1](#) for direct authentication using UAF at the AS. The only difference in the lab build was the
2033 URLs for the NNAS and the Nok Nok Labs Gateway, as the AS and the OIDC IdP used two different
2034 instances of the Nok Nok Labs server.

2035 [**4.3.1.2 Configure an Access Token Management Instance**](#)

2036 1. On the **OAuth Settings** section tab, click **Access Token Management**.

2037 2. Click **Create New Instance**.

2038 a. On the **Type** tab, provide an **INSTANCE NAME** and **INSTANCE ID** (Figure 4-25).

2039 i. Select **Internally Managed Reference Tokens** for the **TYPE**.

2040 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
PARENT INSTANCE	None

Visit Pingidentity.com for additional types

Cancel Next

2041

2042 Although we have selected reference tokens, the ID Token is always issued in
 2043 the form of a JWT. The token that is being configured here is not the ID Token,
 2044 but rather the access token that will be issued to authorize the RP to call the
 2045 userinfo endpoint at the IdP to request additional claims about the user.
 2046 Because this access token only needs to be validated by the OIDC IdP itself,
 2047 reference tokens are sufficient. In the Authorization Code flow, the RP obtains
 2048 both the ID Token and the access token in exchange for the authorization code
 2049 at the IdP's token endpoint.

- 2050 b. Click the **Instance Configuration** tab to configure some security properties of the access
 2051 token, such as its length and lifetime (Figure 4-26). For the lab build, the default values
 2052 were accepted.

2053 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

2054

- 2055 c. On the **Access Token Attribute Contract** tab, extend the contract with any attributes
 2056 that will be included in the ID Token (Figure 4-27). In the example shown in Figure 4-27,
 2057 several attributes that will be queried from AD have been added.

2058 **Figure 4-27 Access Token Attribute Contract**

The screenshot shows the 'Access Token Management' section of the PingFederate interface. The 'Access Token Attribute Contract' tab is active. A table lists attributes such as department, email, family_name, given_name, name, phone_number, postal_code, preferred_username, state, street_address, sub, title, and updated_at, each with an 'Edit | Delete' link. There is also an 'Add' button at the bottom right of the list.

Type	Action
department	Edit Delete
email	Edit Delete
family_name	Edit Delete
given_name	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

2059

- 2060 d. There is no need to configure the **Resource URIs** or **Access Control** tabs; these tabs can
2061 be skipped.
- 2062 e. Click **Done**, and then click **Save**.

2063 *4.3.1.3 Configure an IdP Adapter Mapping*

2064 The IdP Adapter Mapping determines how the persistent grant attributes are populated using
2065 information from authentication adapters.

- 2066 1. Click the **OAuth Settings** section tab, and then click **IdP Adapter Mapping**.
- 2067 2. Select the UAF adapter instance created in [Section 4.3.1.1](#), and then click **Add Mapping**.

- 2068 a. On the **Contract Fulfillment** tab, map both **USER_KEY** and **USER_NAME** to the
 2069 **username** value returned from the adapter (Figure 4-28).

2070 **Figure 4-28 Access Token Contract Fulfillment**

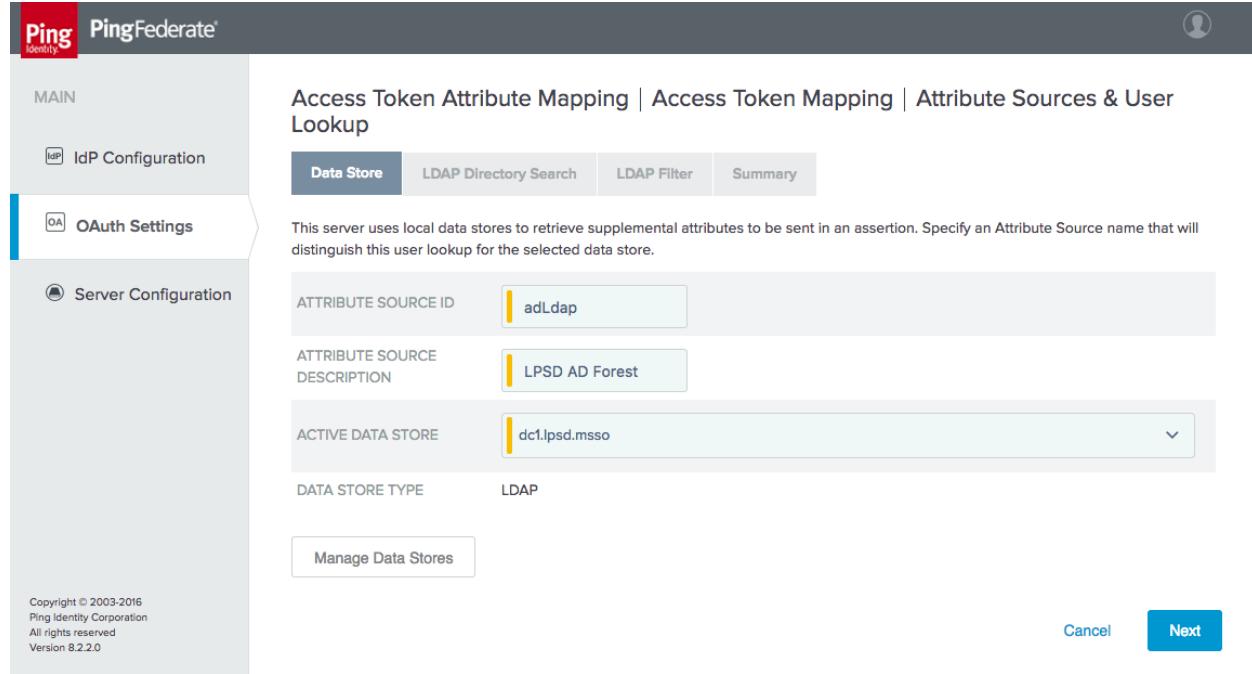
2071

2072 *[4.3.1.4 Configure an Access Token Mapping](#)*

2073 The Access Token Mapping determines how the access token attribute contract is populated. In this
 2074 example, the values returned from the adapter are supplemented with attributes retrieved from AD,
 2075 and issuance criteria are used to require the user to be actually found in AD for a token to be issued.
 2076 Depending on the credential and access life-cycle processes used in a given organization, there may be a
 2077 lag in deactivating the authenticator or the AD account when a user's access is terminated.
 2078 Organizations' authentication policies should account for these conditions and should allow or deny
 2079 access appropriately.

- 2080 1. On the **OAuth Settings** section tab, click **Access Token Mapping**.
- 2081 2. Under **CONTEXT** and **ACCESS TOKEN MANAGER**, select the IdP Adapter and Access Token
 2082 Manager created in the preceding steps, and click **Add Mapping**.
 - 2083 a. On the **Attribute Sources & User Lookup** tab, click **Add Attribute Source**. This brings up
 2084 another multi-tabbed configuration.
 - 2085 i. On the **Data Store** tab, give the attribute source an ID and description
 2086 (Figure 4-29). For **ACTIVE DATA STORE**, select the user store created in
 2087 [Section 4.1](#).

2088 Figure 4-29 Data Store for User Lookup



2089

- 2090 ii. On the **LDAP Directory Search** tab, specify the **BASE DN** and **SEARCH SCOPE**,
2091 and add the AD attributes to be retrieved (Figure 4-30). When specifying
2092 attributes, it is necessary to first select the root object class that contains the
2093 attribute. Common attributes associated with user accounts may be derived
2094 from the **User** or **OrganizationalPerson** class, for example. Refer to Microsoft's
2095 AD Schema documentation [\[26\]](#) to identify the class from which a given
2096 attribute is derived.

2097 Figure 4-30 Attribute Directory Search

The screenshot shows the 'PingFederate' interface for 'Attribute Directory Search'. The left sidebar has sections for MAIN (IdP Configuration, OAuth Settings, Server Configuration), and a copyright notice for Ping Identity Corporation. The main content area has tabs for Data Store, LDAP Directory Search (selected), LDAP Filter, and Summary. It displays configuration for directory search, including:

- BASE DN:** OU=Demo Users,DC=lpss,DC=mss0
- SEARCH SCOPE:** Subtree
- Attributes to return from search:**

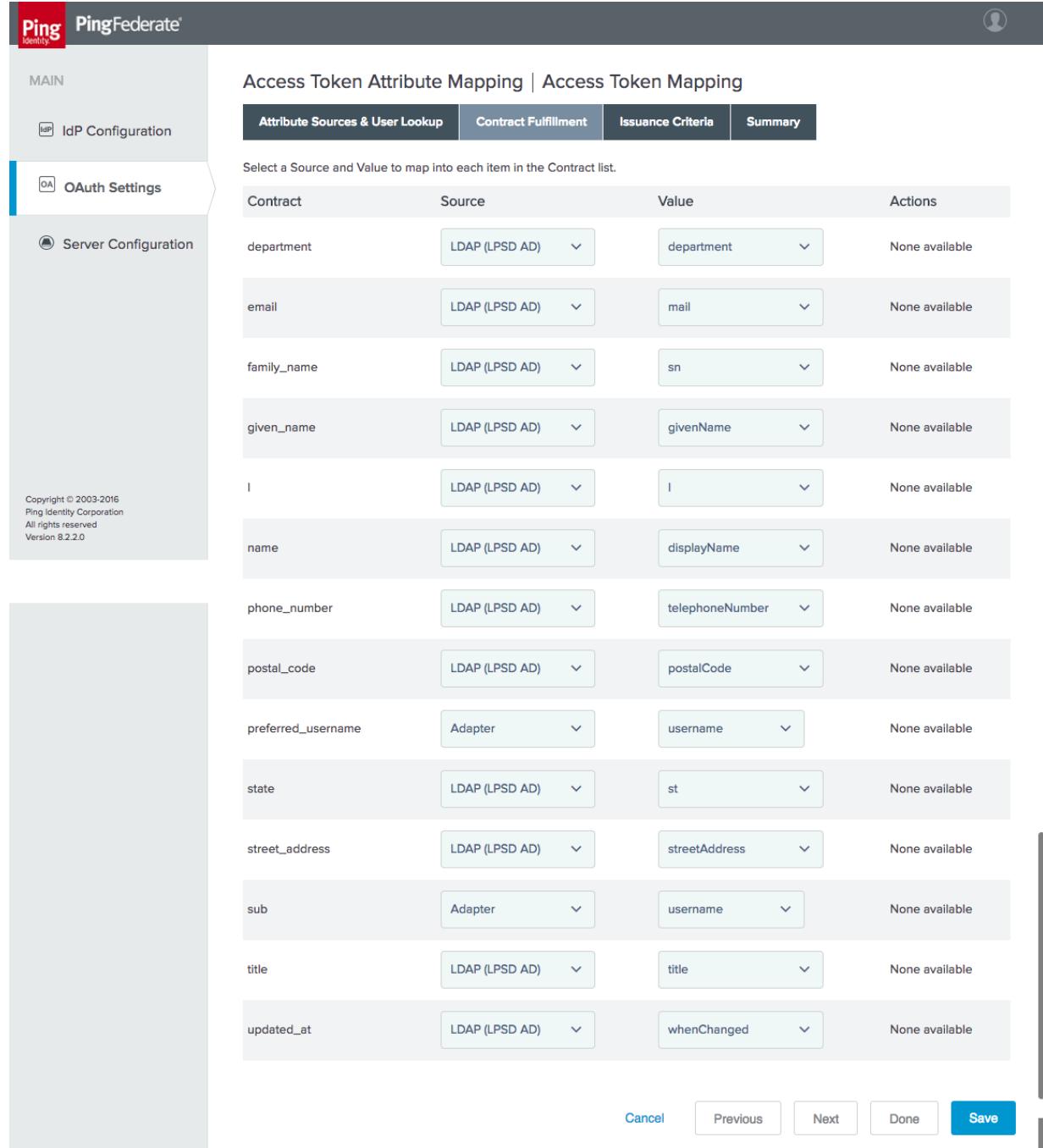
ROOT OBJECT CLASS	ATTRIBUTE	Action
	Subject DN	
	department	Remove
	displayName	Remove
	givenName	Remove
I	mail	Remove
	objectClass	Remove
	postalCode	Remove
	sn	Remove
	st	Remove
	streetAddress	Remove
	telephoneNumber	Remove
	title	Remove
	whenChanged	Remove
<input type="button" value="- SELECT -"/> <input type="button" value="Add Attribute"/>		
- Buttons at the bottom:** Cancel, Previous, Next, Done, Save.

2098

- 2099 iii. On the **LDAP Filter** tab, create the filter to select the relevant user account. In
2100 this example, the username from the adapter is matched against the AD SAM
2101 account name:
2102 `sAMAccountName=${adapter.username}`
2103 iv. Click **Done** to exit the attribute source configuration.

- 2104 b. On the **Contract Fulfillment** tab, specify the source and value to use for each attribute in
 2105 the access token attribute contract (Figure 4-31).

2106 **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 various 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.

2107

- 2108 c. On the **Issuance Criteria** tab, define a rule that will prevent token issuance if the user
 2109 account doesn't exist in AD (Figure 4-32). In this case, the **objectClass** attribute, which
 2110 all AD objects have, is checked for the **Value** called **user**. If no user account is found in
 2111 AD, this attribute will have no **Value**, the **Condition** will be false, and the specified **Error
 2112 Result** will appear in the PingFederate server log.

2113 **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

- 2114 d. Click **Done**, and then click **Save** to finish the Access Token Attribute Mapping
 2115 configuration.

2117 *4.3.1.5 Configure an OIDC Policy*

- 2118 1. On the **OAuth Settings** tab, click **OpenID Connect Policy Management**.
- 2119 2. Click **Add Policy**.
 - 2120 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
 2121 attributes will be included as claims in the ID Token JWT. The advantage of this
 2122 approach is that the RP can directly obtain user attributes from the ID Token without
 2123 making additional requests to the IdP. The alternative is to include only a subject claim
 2124 in the ID Token, and to have the RP call the IdP's userinfo endpoint to obtain additional
 2125 user attributes.

2127 Figure 4-33 OIDC Policy Creation

The screenshot shows the 'Policy Management | Policy' screen in the PingFederate interface. The left sidebar has tabs for 'MAIN', 'IdP Configuration', 'OAuth Settings' (which is selected), and 'Server Configuration'. The main area has tabs for 'Manage Policy', 'Attribute Contract', 'Attribute Sources & User Lookup', 'Contract Fulfillment', 'Issuance Criteria', and 'Summary'. The 'OAuth Settings' tab is active. The form fields include:

- 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.

2128

- 2129 b. On the **Attribute Contract** tab, the set of attributes in the contract can be edited
 2130 (Figure 4-34). The contract is automatically populated with the standard claims defined
 2131 in the OIDC Core specification. In the example shown in Figure 4-34, some claims have
 2132 been removed and others have been added to accommodate the attribute available
 2133 from AD.

2134 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 main content area is titled "Policy Management | Policy". At the top of the main area is a navigation bar with tabs: Manage Policy, Attribute Contract (which is selected and highlighted in blue), Attribute Sources & User Lookup, Contract Fulfillment, Issuance Criteria, and Summary. Below the navigation bar, 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." The "Attribute Contract" section lists attributes with "Edit | Delete" links: sub, address.locality, address.postal_code, address.street_address, department, email, family_name, given_name, name, phone_number, preferred_username, state, title, and updated_at. There are "Add" and "Cancel" buttons at the bottom of this section, along with "Previous", "Next", "Done", and a blue "Save" button.

2135

- 2136 c. Skip the **Attribute Sources & User Lookup** tab; there is no need to retrieve additional
2137 attributes.
- 2138 d. On the **Contract Fulfillment** tab, populate the OIDC attributes with the corresponding
2139 values from the Access Token context (Figure 4-35).

2140 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 attribute contracts:

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.

2141

- 2142 e. There is no need for additional issuance criteria; therefore, skip the **Issuance Criteria** tab.
- 2143
- 2144 f. Click **Save** to complete the OIDC Policy configuration.

2145 4.3.2 Configuring the OIDC Client Connection

2146 Registering a client at an OIDC IdP is analogous to creating an SP connection at a SAML IdP. Some
2147 coordination is required between the administrators of the two systems. The client ID and client secret
2148 must be provided to the RP, and the RP must provide the redirect URI to the IdP.

2149 1. To add a client, click the **OAuth Settings** section tab, and then click **Create New** under **Clients**.

2150 a. Create a **CLIENT ID** and **CLIENT SECRET** (Figure 4-36). If mutual TLS authentication is
2151 being used instead, the RP must provide its certificate, which can be uploaded to the
2152 client creation page. Only the **Authorization Code** grant type is needed for this
2153 integration. In the example shown in Figure 4-36, user prompts to authorize the sharing
2154 of the user’s attributes with the RP have been disabled in favor of streamlining access to
2155 applications.

SECOND DRAFT

2156 Figure 4-36 OIDC Client Configuration

The screenshot shows the 'Client' configuration page in PingFederate. The left sidebar has tabs for MAIN, IdP Configuration, OAuth Settings (which is selected), and Server Configuration. The main content area is titled 'Client' and describes managing configuration and policy information about a client. It shows a 'CLIENT ID' of 'MotorolaAS' and 'CLIENT AUTHENTICATION' set to 'NONE'. Under 'SECRET', there is a masked input field and a 'Generate Secret' button. A 'CHANGE SECRET' link is also present. The 'CLIENT TLS CERTIFICATE' section includes an 'ISSUER' dropdown set to '- SELECT -' and a 'SUBJECT DN' input field. Below these, a note says 'You can also extract the Subject DN from a certificate file.' with a 'Choose file' button and an 'Extract' button. The 'NAME' is 'Motorola's AS' and the 'DESCRIPTION' is a large empty text area. The 'REDIRECT URIS' section lists two URLs with 'Edit | Delete' links and an 'Add' button. The 'LOGO URL' is 'https://op1.lpsd.mso:9031/assets/image/'. Under 'BYPASS AUTHORIZATION APPROVAL', 'Bypass' is checked. In 'RESTRICT SCOPES', 'Restrict' is unchecked. The 'ALLOWED GRANT TYPES' section has 'Authorization Code' checked and other options like 'Resource Owner Password Credentials', 'Refresh Token', etc., unchecked. The 'DEFAULT ACCESS TOKEN MANAGER' is set to 'RDO UAF'. In 'PERSISTENT GRANTS EXPIRATION', 'Use Global Setting' is selected. The 'REFRESH TOKEN ROLLING POLICY' has 'Use Global Setting' checked. The 'OPENID CONNECT' section shows 'ID Token Signing Algorithm' set to 'Default'. The 'Policy' dropdown is set to 'Default'. A checkbox for 'Grant Access to Session Revocation API' is unchecked. At the bottom right are 'Cancel' and 'Save' buttons.

2157

2158

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 [27]. This section briefly describes the overall deployment architecture used for this build.

The Nok Nok Labs SDKs can be directly integrated into mobile applications, providing UAF client functionality directly within the application. This deployment would be more suitable to use cases that do not involve federation, where the requirement is to authenticate users directly at the application 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 application running the Nok Nok Labs SDK by using mobile platform technologies to open links directly in mobile applications (*App Links* for Android, or *Universal Links* for iOS). The FIDO client that processes the OOB authentication request can be either a custom application incorporating the Nok Nok Labs SDK, or the Nok Nok Labs Passport application, 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 application session management infrastructure
- Nok Nok Labs Gateway Tutorial Application – a demonstration web application implementation that provides simple U2F and UAF authentication and registration workflows

2194 In a typical production implementation, the gateway functions for authenticator management
2195 (registration and de-registration) would typically require strong authentication, implemented through
2196 the Gateway's session management integration. Nok Nok Labs' documentation for the PingFederate
2197 plugin provides examples for defining a "reg" OAuth scope to request authenticator registration. An
2198 OAuth Scope Authentication Selector could be used in a PingFederate authentication policy to trigger
2199 the required strong authentication process.

2200 **5.1 Platform and System Requirements**

2201 The following subsections list the hardware, software, and network requirements for the various Nok
2202 Nok Labs components.

2203 **5.1.1 Hardware Requirements**

2204 Nok Nok Labs specifies the following minimum hardware requirements for the NNAS and Nok Nok Labs
2205 Gateway components. The requirements for acceptable performance will depend on the anticipated
2206 user population and server load. See the *Enabling Scalability & Availability* section of the *Solution Guide*
2207 for architecture guidance on deploying the NNAS in a clustered configuration.

- 2208 ■ Processor: 1 CPU
- 2209 ■ Memory: 4 GB RAM
- 2210 ■ Hard disk drive size: 10 GB

2211 **5.1.2 Software Requirements**

2212 Complete software requirements for the NNAS are provided in the *Nok Nok Labs Authentication Server*
2213 *Administration Guide* [28]. The major requirements are summarized below:

- 2214 ■ OS: Red Hat Enterprise Linux 7 or CentOS 7
 - 2215 ■ Relational database system: MySQL 5.7.10 or later versions, Oracle Database 12c, or PostgreSQL
2216 9.2 or 9.4
 - 2217 ■ Application server: Apache Tomcat 8.0.x or 8.5.x
 - 2218 ■ Java: Oracle JDK Version 8
 - 2219 ■ Build tool: Apache Ant 1.7 or later versions
 - 2220 ■ For clustered deployments: Redis 2.8 or later versions
 - 2221 ■ Google Cloud Messenger (GCM) or Apple Push Notification System (APNS), if using push
2222 messages
- 2223 The Nok Nok Labs PingFederate Adapter is compatible with PingFederate 8.1.3 or later versions.
- 2224 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 application.
- 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 application, 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.
- Application link/universal link integration (optional) – In the lab, the default setting using an application link under <https://app.noknok.com> was used. This is acceptable for testing, but in a production deployment, an application link pointing to the IdP's actual domain name would typically be used. It should be noted that the FQDN for the application 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.

- 2260 • `uaf.facet.id` – There is no need to modify the Facet ID setting to enable the use of the
 2261 Passport application for OOB authentication; however, if other custom applications were
 2262 directly integrating the Nok Nok Labs SDK, they would need to be added here.
- 2263 ▪ For a production deployment, client certificate authentication to the Authentication Server
 2264 should be enabled. This is done by configuring the Tomcat HTTP connector to require client
 2265 certificates. This requires provisioning a client certificate for the gateway (and any other servers
 2266 that need to call the Nok Nok Labs APIs). See the notes in Section 5.3 of the *Administration*
 2267 *Guide* about configuring the Gateway to use client certificate authentication. A general
 2268 reference on configuring TLS in Tomcat 8 can be found at <https://tomcat.apache.org/tomcat-8.0-doc/ssl-howto.html>.

2270 5.3 How to Install and Configure the FIDO UAF Gateway Server

2271 The Nok Nok Labs Gateway application is delivered as a Web Archive (WAR) file that can be deployed to
 2272 a Tomcat server. For the lab build, it was deployed on the same server as the NNAS.

2273 Configure the required settings in the `nnlgateway.properties` file, including the settings listed below:

- 2274 ▪ `mfas_location` – NNAS URL
- 2275 ▪ `server.auth.enabled` – should be set to true; also requires configuring the trust-store settings
- 2276 ▪ `client.auth.enabled` – see notes in Section 5.2 above; should be enabled for strong client
 2277 authentication in production deployments; also requires configuring the keystore settings

2278 In addition, the Gateway Tutorial application was installed by deploying the `gwtutorial.war` file and
 2279 configuring the required URLs in `gwtutorial.properties`.

2280 5.4 How to Install and Configure the FIDO UAF Adapter for the OAuth 2 AS

2281 Nok Nok Labs provided a tar file containing a set of software tools for integration and testing with
 2282 PingFederate. Version 5.1.0.501 of the Ping Integration library was used for the lab build. The
 2283 installation process is summarized below; refer to the *Nok Nok PingFederate Adapter Integration Guide*
 2284 [\[29\]](#) for full details:

- 2285 1. Extract the `adapter` folder from the `nnl-ping-integration-5.1.0.501.tar` file onto the PingFederate
 2286 server where the adapter will be installed.
- 2287 2. Stop PingFederate if it is running, and run the installation script. The path to the PingFederate
 2288 installation is passed as an argument; run the script by using an account with write access to the
 2289 PingFederate installation:

```
$ ./adapter-deploy.sh /usr/share/pingfederate-8.2.2/pingfederate
```
- 2290 3. Configure the `adapter.properties` file (located in the PingFederate directory under
 2291 `server/default/conf`) as required for the server and client TLS authentication settings specified

2293 earlier in the Authentication Server configuration. If push notifications are enabled, configure
2294 the relevant settings.

- 2295 4. The *Configure Session Manager* and *Deploy Nok Nok Gateway OOB* sections of the *Integration*
2296 *Guide* provide settings to use PingFederate to protect the Registration endpoint on the Nok Nok
2297 Labs Gateway. This could be used in conjunction with the custom “reg” scope and a PingFederate
2298 authentication policy to require strong authentication prior to UAF authenticator registration.
2299 This configuration was not tested in the lab.

2300 The *Configure PingFederate Console* section of the *Integration Guide* walks through the complete
2301 configuration of a PingFederate OIDC provider. See [Section 4.3](#) of this guide for the procedure to
2302 configure the OpenID Provider.

2303 **6 How to Install and Configure the FIDO U2F 2304 Authentication Server**

2305 The SKCE from StrongKey performs the FIDO U2F server functionality in the build architecture.
2306 StrongKey’s main product is the StrongKey Tellaro Appliance, but the company also distributes much of
2307 its software under the *Lesser General Public License (LGPL)*, published by the Free Software Foundation.
2308 SKCE 2.0 Build 163 was downloaded from its repository on *Sourceforge* and was used for this build. For
2309 more information, documentation, and download links, visit the vendor’s site at
2310 <https://sourceforge.net/projects/skce/>.

2311 **6.1 Platform and System Requirements**

2312 The following subsections document the software, hardware, and network requirements for SKCE 2.0.

2313 **6.1.1 Software Requirements**

2314 StrongKey’s website lists the OSs on which SKCE has been tested:

- 2315 ▪ CentOS 6.X or 7.X, 64-bit
2316 ▪ Windows 7 Professional, 64-bit

2317 Since SKCE is a Java application, in theory it should be able to run on any OS that supports a compatible
2318 version of Java and the other required software. The application was built with the Oracle JDK Version 8,
2319 Update 72. For this build, SKCE was installed on a CentOS 7.4 server; therefore, these steps assume a
2320 Linux installation.

2321 SKCE can be installed manually or with an installation script included in the download. SKCE depends on
2322 other software components, including an SQL database, an LDAP directory server, and the Glassfish Java
2323 application server. By default, the script will install MariaDB, OpenDJ, and Glassfish all on a single server.
2324 SKCE can also utilize AD for LDAP.

2325 For this build, the scripted installation was used with the default software components. The required
2326 software components, which are listed below, must be downloaded prior to running the installation
2327 script:

- 2328 ■ Glassfish 4.1
2329 ■ Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 8
2330 ■ JDK 8, Update 121
2331 ■ OpenDJ 3.0.0
2332 ■ MariaDB 10.1.22
2333 ■ MariaDB Java Client

2334 See StrongKey's scripted installation instructions for details and download links:
2335 <https://sourceforge.net/p/skce/wiki/Install%20StrongKey%20CryptoEngine%202.0%20%28Build%20163%29%20Scripted/>.

2337 To download OpenDJ, you must register for a free account for *ForgeRock BackStage*.

2338 SKCE can also utilize an AD LDAP service. The LDAP directory contains system user accounts for
2339 managing the SKCE (generating cryptographic keys, etc.). Data pertaining to registered users and
2340 authenticators is stored in the SQL database, not in LDAP.

2341 **6.1.2 Hardware Requirements**

2342 StrongKey recommends installing SKCE on a server with at least 10 GB of available disk space and 4 GB
2343 of RAM.

2344 **6.1.3 Network Requirements**

2345 The SKCE API is hosted on Transmission Control Protocol (TCP) Port 8181. Any applications that request
2346 U2F registration, authentication, or deregistration actions from the SKCE need to be able to connect on
2347 this port. Glassfish runs an HTTPS service on this port. Use firewall-cmd, iptables, or any other system
2348 utility for manipulating the firewall to open this port.

2349 Other network services listen on the ports listed below. For the scripted installation, where all these
2350 services are installed on a single server, there is no need to adjust firewall rules for these services
2351 because they are only accessed from localhost.

- 2352 ■ 3306 – MariaDB listener
2353 ■ 4848 – Glassfish administrative console
2354 ■ 1389 – OpenDJ LDAP service

6.2 How to Install and Configure the FIDO U2F Authentication Server

StrongKey's scripted installation process is documented at

<https://sourceforge.net/p/skce/wiki/Install%20StrongKey%20CryptoEngine%202.0%20%28Build%20163%29%20Scripted/>.

The installation procedure consists of the following steps:

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

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

1. Download the software: Download and unzip the SKCE build to a directory on the server where SKCE is being installed. Download all installers as directed in the SKCE instructions to the same directory.
2. Change software versions as required in the install script: If different versions of any of the 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, is not recommended. For the lab build, JDK Version 8u151 was used instead of the version referenced in the instructions. This required updating the `JDK` and `JDKVER` settings in the file.
3. Change passwords in the install script: Changing the default passwords in the delivered script is strongly recommended. The defaults are readily discoverable, as they are distributed with the software. Passwords should be stored in a password vault or other agency-approved secure storage. Once the installation script has been run successfully, the script should be deleted or sanitized to remove passwords. The following lines in the install script contain passwords:

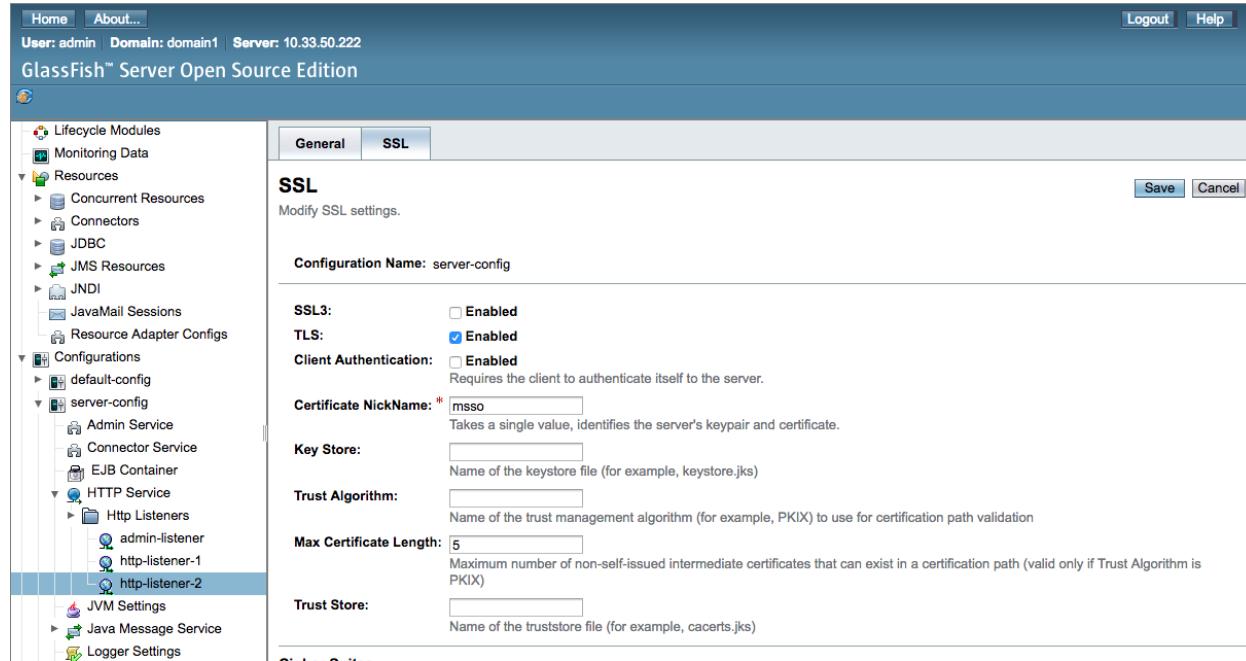
```
LINUX_PASSWORD=ShaZam123          # For 'strongauth' account  
GLASSFISH_PASSWORD=adminadmin      # Glassfish Admin password  
MYSQL_ROOT_PASSWORD=BigKahuna     # MySQL 'root' password  
MYSQL_PASSWORD=AbracaDabra        # MySQL 'skles' password  
SKCE_SERVICE_PASS=Abcd1234!       # Webservice user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

- 2387 SEARCH_LDAP_BIND_PASS=Abcd1234 !
- 2388 4. Set the Application ID URL: The Application ID setting in *install-skce.sh* should point to a URL that
2389 will be accessible to clients where the *app.json* file can be downloaded. The default location is a
2390 URL on the SKCE server, but the SKCE would not be exposed to mobile clients in a typical
2391 production deployment. In the lab, *app.json* was hosted on the PingFederate server hosting the
2392 IdP in the following location:
- 2393 */usr/share/pingfederate-8.3.2/pingfederate/server/default/conf/template/assets/scripts*
- 2394 which enables the file to be accessed by clients at the following URL:
2395 <https://idp1.spsd.mso:9031/assets/scripts/app.json>.
- 2396 5. Run the script: *install-skce.sh* must be run as the root user. If the install script terminates with an
2397 error, troubleshoot and correct any problems before continuing.
- 2398 6. (For CentOS 7) Create firewall rule: The install script attempts to open the required port using
2399 iptables, which does not work on CentOS 7. In that case, the following commands will open the
2400 port:
- 2401 # firewall-cmd --permanent --add-port 8181/tcp
2402 success
2403 # firewall-cmd --reload
2404 success
- 2405 7. Install additional libraries: Depending on how CentOS was installed, some additional libraries
2406 may be required to run the graphical key custodian setup tool. In the lab, the SKCE server did
2407 not include X11 or a graphical desktop, so the key custodian setup was run over Secure Shell
2408 (SSH) with X11 forwarding. To install additional libraries needed for this setup, run the following
2409 commands:
- 2410 # yum install libXrender
2411 # yum install libXtst
- 2412 Note that running the graphical configuration tool over SSH also requires configuring X11
2413 forwarding in the SSH daemon (**sshd**) on the server and using the **-x** command line option when
2414 connecting from an SSH client.
- 2415 8. Run the key custodian setup tool: In production deployments, the use of a Hardware Security
2416 Module (HSM) and USB drive for the security officer and key custodian credentials is strongly
2417 recommended. In the lab, the software security module was used. Also, the lab setup utilized a
2418 single SKCE server; in this case, all instructions pertaining to copying keys to a secondary
2419 appliance can be ignored.

- 2420 9. Restart Glassfish: On CentOS 7, run the following command:
- 2421 \$ sudo systemctl restart glassfishd
- 2422 10. Complete Steps 5.1 and 5.2 in the SKCE installation instructions to activate the cryptographic module.
- 2423
- 2424 11. Complete Step 5.3 in the SKCE installation instructions to create the domain signing key. When prompted for the Application ID, use the URL referenced above in the Application ID setting of the *install-skce.sh* script.
- 2425
- 2426
- 2427 12. Complete Step 6 if you are installing secondary SKCE instances; this was not done for this build but is recommended for a production installation.
- 2428
- 2429 13. Install a TLS certificate (optional): The SKCE installation script creates a self-signed certificate for the SKCE. It is possible to use the self-signed certificate, though PingFederate and any other servers that integrate with the SKCE would need to be configured to trust it. However, many organizations will have their own CAs, and will want to generate a trusted certificate for the SKCE for production use. To generate and install the certificate, follow the steps listed below:
- 2430
- 2431
- 2432
- 2433
- 2434 a. The keystore used by the SKCE Glassfish server is listed below:
- 2435 /usr/local/strongauth/glassfish4/glassfish/domains/domain1/config/keystor
2436 e.jks
- 2437 b. The default password for the keystore is “changeit”.
- 2438 c. Use keytool to generate a keypair and certificate signing request. For example, the
2439 following commands generate a 2048-bit key pair with the alias “mss0,” and export a
2440 Certificate Signing Request (CSR):
- 2441 \$ keytool -genkeypair -keyalg RSA -keysize 2048 -alias mss0 -keystore
2442 keystore.jks
- 2443 \$ keytool -certreq -alias mss0 -file strongauth.req -keystore
2444 keystore.jks
- 2445 d. Submit the CSR to your organization’s CA, and import the signed certificate along with
2446 the root and any intermediates:
- 2447 \$ keytool -import -trustcacerts -alias mss0-root -file lab-certs/root.pem
2448 -keystore keystore.jks
- 2449 \$ keytool -import -alias mss0 -file lab-certs/strongauth.lpsd.mss0.cer -
2450 keystore keystore.jks
- 2451 e. To configure the SKCE to use the new certificate, log in to the Glassfish administrative
2452 console on the SKCE server. The console runs on Port 4848; the username is “admin,”
2453 and the password will be whatever was configured for **GLASSFISH_PASSWORD** in the
2454 *install-skce.sh* script.

- 2455 i. Navigate to *Configurations, server-config, HTTP Service, Http Listeners, http-listener-2*, as shown in Figure 6-1. On the **SSL** tab, set the **Certificate NickName** to the alias that was created with the “keytool -genkeypair” command above.

2458 **Figure 6-1 Glassfish SSL Settings**



- 2459 f. Click **Save**, and then restart glassfish. If logged on as the glassfish user, run the following command:
- ```
2460 $ sudo service glassfishd restart
```
- 2462                   g. In a browser, access the SKCE web service on Port 8181, and ensure that it is using the newly created certificate.
- 2463                   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:
- ```
2464                   $ keytool -import -trustcacerts -file lab-certs/root.pem -keystore $JAVA_HOME/jre/lib/security/cacerts
```
- 2465 14. Test the FIDO Engine: Follow the testing instructions under Step 4 at the following URL:
<https://sourceforge.net/p/skce/wiki/Test%20SKCE%202.0%20Using%20a%20Client%20Program%20%28Build%20163%29/#4test-skcefido-engine.>

2473 There are additional tests on that web page to test the other cryptographic functions of the
2474 SKCE; however, only the FIDO Engine tests are critical for this build.

2475 If the FIDO Engine tests are completed without errors, proceed to Section 6.3 to integrate the SKCE with
2476 the IdP. If any errors are encountered, the Glassfish log file (located at
2477 `/usr/local/strongauth/glassfish4/glassfish/domains/domain1/logs/server.log`) should contain messages
2478 to aid in troubleshooting.

2479 **6.3 How to Install and Configure the FIDO U2F Adapter for the IdP**

2480 To incorporate FIDO U2F authentication into a login flow at the IdP, some integration is needed to
2481 enable the IdP to call the SKCE APIs. In the lab build architecture, FIDO U2F authentication was
2482 integrated into a SAML IdP. PingFederate has a plugin architecture that enables the use of custom and
2483 third-party adapters in the authentication flow. StrongKey provides a PingFederate plugin to enable
2484 PingFederate IdPs (or AS) to support U2F authentication. This section describes the installation of the
2485 plugin on a PingFederate server. For details on how to integrate U2F authentication to a login flow, see
2486 Section 4.2.1.3.

2487 The StrongKey plugin for PingFederate is delivered in a zip file containing documentation and all of the
2488 required program files.

- 2489 1. To begin the installation process, upload the zip file to the PingFederate server where the
2490 StrongKey plugin will be installed, and unzip the files.
- 2491 2. If Apache Ant is not already installed on the server, install it now by using the server's package
2492 manager. For CentOS, this can be done by running the following command:

2493

```
# yum install ant
```
- 2494 3. Once Apache Ant is installed, follow the "Installation" instructions in the *StrongKey – Ping*
2495 *Federate FIDO IdP Adapter Installation Guide* [30], which consist of copying the plugin files to
2496 the required directories in the PingFederate installation, and running *build.sh*. If the script runs
2497 successfully, it will build the plugin using Ant and restart PingFederate.
- 2498 4. Follow the steps in "Table 2: Configure the SKCE" in the *Installation Guide*. For this build, the
2499 *app.json* file needs to be copied to a browser-accessible location on the PingFederate server
2500 where the plugin is being installed. In the lab, we placed it under the following location:

2501 `/usr/share/pingfederate-8.3.2/pingfederate/server/default/conf/template/assets/scripts`
- 2502 5. This enables the *app.json* to be accessed at the URL
2503 `https://idp1.spsd.mss:9031/assets/scripts/app.json`. Note that Steps 4 and 5 in Table 2 of the
2504 *Installation Guide* are required only if the SKCE is using the default self-signed certificate; if a
2505 trusted certificate was installed as described in [Section 6.2](#), then those steps can be skipped.

- 2506 6. Download the JQuery 2.2.0 library at the URL below, and save it to the scripts folder referenced
2507 above: <https://code.jquery.com/jquery-2.2.0.min.js>.
- 2508 7. Follow the steps in “Table 3: Configure the Ping Federate Instance” in the *Installation Guide*. Importing the SKCE self-signed certificate is not required if a trusted certificate was created.
2509 Installation of the JCE unlimited policy was described in the PingFederate installation
2510 instructions in [Section 3](#), so that too can be skipped at this point, if it has already been done.
2511 Steps 7–9 should be completed in any case.
- 2512 8. Follow the steps in “Table 4: Configuring the FIDO Adapter” in the *Installation Guide*. In Step 5, the Domain ID typically should be set to “1,” unless you have defined multiple domains in the SKCE. For the username and password, use the values configured earlier in *install-skce.sh*.
- 2513 9. “Table 5: Ping Federate OAuth Configuration Steps” in the *Installation Guide* provides an example of how to incorporate U2F into a login flow, along with username/password form login, by creating a composite adapter that includes the login form and U2F adapters, and using a selector to activate the composite adapter whenever an OAuth authorization request includes the scope value “*Idap*.” Alternatively, the individual adapters can be called directly in an authentication policy. See Chapter 4 of the *Installation Guide* for additional examples of using U2F in authentication policies.

2523 6.3.1 FIDO U2F Registration in Production

2524 By default, the StrongKey Ping plugin enables the registration of U2F authenticators. In production, an authorized registration process should be established to provide adequate assurance in the binding of the authenticator to a claimed identity. If the FIDO adapter is accessible after single-factor password authentication, organizations may want to disable the registration functionality. See Section B.5 in Volume B of this guide for a discussion of FIDO enrollment.

2529 7 Functional Tests

2530 The MSSO architecture has a number of interoperate components, which can make troubleshooting difficult. This section describes tests than can be performed to validate that individual components are working as expected. If issues are encountered with the overall SSO flow, these tests may help identify the problem area.

2534 7.1 Testing FIDO Authenticators

2535 The FIDO Alliance implements a Functional Certification Program, in which products are evaluated for conformance to the UAF and U2F specifications. Purchasing FIDO-certified authenticators can help avoid potential authenticator implementation issues. Information on the certification program is available at <https://fidoalliance.org/certification/>, and the FIDO Alliance website also lists certified products.

2539 Some resources are available to help troubleshoot individual authenticators:

- 2540 ▪ The Yubico demonstration site provides an interface for testing registration and authentication
2541 with U2F authenticators: <https://demo.yubico.com/u2f>.
- 2542 ▪ The Nok Nok Labs Gateway Tutorial Application supports testing of the registration,
2543 authentication, and transaction verification functions of FIDO UAF authenticators.

2544 **7.2 Testing FIDO Servers**

2545 The StrongKey SKCE documentation includes instructions on testing U2F authenticator registration,
2546 authentication, de-registration, and other functions. See Step 14 in [Section 6.2](#).

2547 To test the NNAS, Nok Nok Labs provides the OnRamp mobile application in the Google Play Store and
2548 the Apple App Store to test the server APIs with UAF authenticators.

2549 **7.3 Testing IdPs**

2550 If federated authentication is failing, the issue may lie at the IdP or the AS. The PingFederate server log
2551 (located by default under <*pingfederate-directory*>/log/server.log), on both ends, should provide
2552 relevant messages.

2553 In some cases, it may be beneficial to look at the assertions being issued by the IdP and to check for the
2554 expected attributes. This could be done by integrating a demonstration application as a federation client
2555 and debugging the data returned in the assertion. For SAML, projects like SimpleSAMLphp
2556 (<https://simplesamlphp.org/>) provide an implementation that is easy to deploy. It is also possible to
2557 perform this testing without installing additional tools.

2558 One method for SAML is to use Chrome Remote Debugging for Android devices:
2559 <https://developers.google.com/web/tools/chrome-devtools/remote-debugging/>.

2560 By logging the authentication flow in the Network pane of Chrome's developer tools, the SAML response
2561 can be extracted and viewed. The authentication flow with the SAML IdP configured in this practice
2562 guide consists of a series of calls to the *SSO.ping* URL at the IdP. Because the SAML POST binding is used,
2563 the final *SSO.ping* response includes an HTML form that submits the SAML response back to the AS. The
2564 SAML response can be found in an input element in the page content:

```
2565 <input type="hidden" name="SAMLResponse"
2566 value="PHNhbWxwO1J1c3BvbN1I1FZ1cnNpb249Ij1uMC1gSUQ9Iko1T2xNN1ZxZW51VnpBU2doSH1saKFLY1I
2567 uOC1gSXNzdWVJbnN0YW50PSIyMDE3LTezLTEzVDEzOjQ5OjE3LjEwMFoiIEluUmVzcG9uc2VUbz0iS2RwMXvFZ
2568 HFPMH1NX2Z0YWV1dWJnRj1vMFBY1iBEZXNOaW5hdGlvbj0iaHR0cHM6Ly9pZG0uc2FuZGJveC5tb3Rvcn9sYXN
2569 vbHV0aW9ucy5jb20vc3AvQUNTLnNhbWwyTiB4bWxuczpZYW1scD0idXJuOm9hc21zOm5hbWVzOnRj01NBTUw6M
2570 i4wOnByb3RvY29sIj48c2FtbDpJc3N1ZX1geG1sbnM6c2FtbD0idXJuOm9hc21zOm5hbWVzOnRj01NBTUw6Mi4
2571 wOmFzc2VydG1vbiI+aWRwMS5zchNkLm1z288L3NhbWw6SXNzdWVypjxkczpTaWduYXR1cmUgeG1sbnM6ZHM9I
2572 mh0dHA6Ly93d3cudzMub3JnLzIwMDAvMDkveG1sZHNPzYMiPg08ZHM6U21nbmVksW5mbz4KPGRzOkNhbm9uaWN
2573 hbG16YXRpb25NZXRob2QgQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzEwL3htbC1leGMTYzE0b
```

2574 iMiLz4KPGRzO1NpZ25hdHVyZU1ldGhvZCBBbGdvcm10aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG1
 2575 sZHNPzY1tb3J1I3JzYS1zaGEyNTYiLz4KPGRzO1J1ZmVyZW5jZSBVUkk9IiNKNU9sTTZWCWVuZVZ6QVNnaEh5b
 2576 GpBS2JSLjgiPgo8ZHM6VHJhbnNmb3Jtcz4KPGRzO1RyYW5zZm9ybSBBbGdvcm10aG09Imh0dHA6Ly93d3cudzM
 2577 ub3JnLzIwMDAvMDkveG1sZHNpZyN1bnz1bG9wZWQtc21nbmF0dXJ1Ii8+CjxkczpUcmFuc2Zvcm0gQWxnb3Jpd
 2578 GhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzEwL3htbC1leGMtYzE0biMiLz4KPC9kczpUcmFuc2Zvcm1zPgo
 2579 8ZHM6RGlnZXN0TWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8wNC94bWx1bmMjc2hhM
 2580 jU2Ii8+CjxkczpEaWdlc3RWYWy1ZT4xd1FpcUNVNmlZYTMzd1FtKzcxbEVsVm1pUUh6T2U5cytBTTdQYTk4V1p
 2581 BPTwvZHM6RGlnZXN0VmFsdWU+CjwvZHM6UmVmZXJ1bmNlPgo8L2RzO1NpZ251ZE1uZm8+CjxkczpTaWduYXR1c
 2582 mVWYWy1ZT4KTHpSbUJhc1k2bndGS3ZyjdjdtL29WYWNJSWRJRUY4eUloV0JXT0NHZ3pyMWtONGVzVi9CU31LQ1N
 2583 XYjhKU1h3QzhWRHNNUnRXOENMNQpVRFV0NTV1OXRca05Wanh2NWR0NSTOYXQ5eWtmdnhXbU9kcGVJVTBzMXNuM
 2584 UJHdytkOTRoZU1CYVdJWE1ZOViRaD1nV3Q2S1100VFhCmRGdDzrRUY1s1NDS1FBQVN1bTEyT2xLV29GK2JSbG1
 2585 HNGVsbtVMTh1N0E3Wi9hRnZ1cDNDNmV5ZEpk1IxaStaK0F6NH1XdmMvNmEKYn1LMTBPZ05pLzBibnpnazd3L
 2586 0psdHk0Z1VEcVd6bxJyRFpwSEJ4ZkFMVW5UV2RPVDVJeko3bmpMQwtBYVN0NDYwWjUyblpBOGFBygpVbzA4T0t
 2587 EYnZVaS9UZ2xTcUzjcdJSYStCae9DbUR3OWJvTG9udz09CjwvZHM6U21nbmF0dXJ1VmFsdWU+CjwvZHM6U21nb
 2588 mF0dXJ1PjxzYw1scDpTdTGF0dXM+PHNhbwXw01N0YXR1c0NvZGUgVmFsdWU9InVybjpvyXNpczpuYW1lczp0Yzp
 2589 TQU1MojIuMDpzdGF0dXM6U3VjY2VzcyIvpjwvc2FtbHA6U3RhdtHvzPjxzYw1sOkFzc2VydG1vbiBJRD0iSF9tL
 2590 lDIR29UVVBELnjV1A0MvhDVvh4YkdLIiBjC3N1ZU1uc3RhbN9Q1jIwMTctMTEtMTNUMTM6NDk6MTcuMTU1WiI
 2591 gVmVyc2lvbj0iMi4wIb4bWxuczpZYw1sPSJ1cm46b2FzaXM6bmFtZXM6dGM6U0FNTDoyLjA6YXNzZXJ0aW9uI
 2592 j48c2FtbDpJc3N1ZXI+aWRwMS5zcHNkLm1zc288L3NhbWw6SXNzdWVYpjxzYw1s01N1Ymp1Y3Q+PHNhbw6TmF
 2593 tZU1IEZvcmlhd0idXJu0m9hc21z0m5hbWVzOnRj01NBTUw6MS4x0m5hbWVpZC1mb3JtYXQ6dW5zcGVjaWzPz
 2594 WQIPnVuY2NvZXR1c3Q0PC9zYw1s0k5hbWVJRD48c2FtbDpTdWJqZWN0Q29uZmlybWF0aW9uIE11dGhvZD0idXJ
 2595 uOm9hc21z0m5hbWVzOnRj01NBTUw6Mi4wOmNtOmJ1YXJ1ciI+PHNhbw6U3ViamVjdeENvbmZpcm1hdG1vbkRh
 2596 GEgUmVjaXBpZW50PSJodHRwczovL21kbS5zYw5kYm94Lm1vdG9yb2xhc29sdXRpb25zLmNvbS9zcC9BQ1Muc2F
 2597 tbDIiIE5vdE9uT3JBZnRlcj0iMjAxNy0xMS0xM1QxMz01NDoxNy4xNTVaIiBjblJ1c3BvbnN1VG89IktkcDF1X
 2598 2RxTzB5TV9mdGF1ZXViZ0Y5bzBQWCIvPjwvc2FtbDpTdWJqZWN0Q29uZmlybWF0aW9uPjwvc2FtbDpTdWJqZWN
 2599 0PjxzYw1s0kNvbmrpdG1vbnMgTm90QmVmb3J1PSIyMDE3LTExLTEzVDEzOjQ0OjE3LjE1NvoiIE5vdE9uT3JBZ
 2600 nRlcj0iMjAxNy0xMS0xM1QxMz01NDoxNy4xNTVaIj48c2FtbDpBdWRpZW5jZVJ1c3RyaWN0aW9uPjxzYw1sOkF
 2601 1ZG11bmNlPmN0b1BpbmdGZWRfZw50aXR5SUQ8L3NhbWw6QXVkaWVuY2U+PC9zYw1sOkF1ZG11bmN1UmVzdHjP
 2602 3RpB24+PC9zYw1s0kNvbmrpdG1vbnM+PHNhbw6QXV0aG5TdGF0Zw1lbnQgU2Vzc21vbkluzGV4PSJIX20uV0h
 2603 Hb1VRUEQuM2NWUDQxWENVWHhIR0s1IEF1dGhuSw5zdGFudD0iMjAxNy0xMS0xM1QxMz00TOxNy4xNTNaIj48c
 2604 2FtbDpBdXRobkNvbR1eHQ+PHNhbw6QXV0aG5Db250Zxh0Q2xhc3NSZWy+dXJu0m9hc21z0m5hbWVzOnRj01N
 2605 BTUw6Mi4wOmFj0mNsYXNzZXM6dW5zcGVjaWzPzQ8L3NhbWw6QXV0aG5Db250Zxh0Q2xhc3NSZWy+PC9zYw1sO
 2606 kF1dGhuQ29udGV4d48L3NhbWw6QXV0aG5TdGF0Zw1lbnQ+PHNhbw6QXR0cmlidXR1U3RhdGvtZw50PjxzYw1
 2607 s0kF0dHjpyNv0ZsBOYw11PSJ1aWQjIE5hbWVGb3JtYXQ9InVybjpvyXNpczpuYW1lczp0YzpTQU1MojIuMDphd
 2608 HRybmFtZS1mb3JtYXQ6YmFzaWMiPjxzYw1s0kF0dHjpyNv0ZVZhBHV1IHzaTp0eXB1PSJ4czpzdHJpbmcIiH
 2609 tbG5zOnhzPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxL1hNTFNjaGvtYSIgeG1sbnM6eHNPSPJodHRwOi8vd3d3L
 2610 ncZLm9yZy8yMDAxL1hNTFNjaGvtYSI1pbN0Yw5jZSI+dW5jY291dGVzdQ8L3NhbWw6QXR0cmlidXR1VmFsdW
 2611 +PC9zYw1s0kF0dHjpyNv0ZT48c2FtbDpBdHRYaWJ1dGvWYw1ZT48L3NhbWw6QXR0cmlidXR1VmFsdW
 2612 21z0m5hbWVzOnRj01NBTUw6Mi4wOmF0dHjyUw11LWvcm1hdDpiYXNpYyI+PHNhbw6QXR0cmlidXR1VmFsdW
 2613 geHNpOnR5cGU9InhzOnN0cmluZyIgeG1sbnM6eHM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvWE1MU2NoZW1hI
 2614 iB4bWxuczp4c2k9Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvWE1MU2NoZW1hLWluc3RhbN1Ij51bmNjb2V0Z
 2615 0Ndwvc2FtbDpBdHRYaWJ1dGvWYw1ZT48L3NhbWw6QXR0cmlidXR1Pjwvc2FtbDpBdHRYaWJ1dGVTdGF0Zw1b
 2616 nQ+PC9zYw1s0kFzc2VydG1vbj48L3NhbWxw01J1c3BvbnN1Pg=="/>

2617 The “value” string is the base64-encoded SAML response. A few lines of Python can get the SAML
 2618 response into a readable format. In this example, the value above has been saved to a file called
 2619 *samlresp.txt*:

2620 **\$ python**
 2621 Python 2.7.10 (default, Feb 7 2017, 00:08:15)
 2622 [GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin

```

2623 Type "help", "copyright", "credits" or "license" for more information.
2624 >>> import base64
2625 >>> import xml.dom.minidom
2626 >>> respFile = open("samlresp.txt", "r")
2627 >>> respStr = base64.b64decode(respFile.read())
2628 >>> respXml = xml.dom.minidom.parseString(respStr)
2629 >>> print(respXml.toprettyxml())
2630 <?xml version="1.0" ?>
2631 <samlp:Response Destination="https://idm.sandbox.motorolasolutions.com/sp/ACS.saml2"
2632 ID="J501M6VqeneVzASghHyljAKbR.8" InResponseTo="Kdp1u_dq00yM_ftaeueubgF9o0PX"
2633 IssueInstant="2017-11-13T13:49:17.100Z" Version="2.0"
2634 xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
2635     <saml:Issuer
2636     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">idp1.spds.mss</saml:Issuer>
2637         <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
2638             <ds:SignedInfo>
2639                 <ds:CanonicalizationMethod
2640                     Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2641
2642
2643                 <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-
2644 more#rsa-sha256"/>
2645                     <ds:Reference URI="#J501M6VqeneVzASghHyljAKbR.8">
2646                         <ds:Transforms>
2647                             <ds:Transform
2648                             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
2649                             <ds:Transform
2650                             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2651                         </ds:Transforms>
2652                         <ds:DigestMethod
2653                             Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
2654 <ds:DigestValue>1vQiqCU6iYa33vQm+711ElVmIqHzOe9s+AM7Pa98VZA=</ds:DigestValue>
2655                         </ds:Reference>
2656                     </ds:SignedInfo>
2657                     <ds:SignatureValue>
2658 LzRmBarY6nwFKrvv7S/oVacIIIdIEF8yIhWBWOCGgzrlkN4esV/BSyKCSWb8JSXwC8VDsMRtW8CL5
2659 UDUt55u9tBkNVjxv5dt5+Nat9ykfvxWmOdpeIU0s1sn1BGw+d94heIBaWIXMY9YQh9gWt6JYt9Qa
2660 dFt6kEF5KSCKQAASe1201KWoF+bRlmG4elm5LM8u7A7Z/aFvup3C6eydJp+R1i+Z+Az4yWvc/6a
2661 byK10OgNi/0bnzkk7w/Jlty4fUDqWzmrrDZpHBxfALUnTWdOT5IzJ7njLAkAaSt460Z52nZA8aAb
2662 Uo8OKDbvUi/TglSqFcp2Ra+BhOCmDw9boLonw==
2663 </ds:SignatureValue>
2664             </ds:Signature>
2665             <samlp:Status>
2666                 <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
2667             </samlp:Status>
2668             <saml:Assertion ID="H_m.WHGoUQPD.3cVP41XCUXxbGK" IssueInstant="2017-11-
2669 13T13:49:17.155Z" Version="2.0" xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
2670                 <saml:Issuer>idp1.spds.mss</saml:Issuer>
2671                 <saml:Subject>
2672                     <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
2673 format:unspecified">unccoetest4</saml:NameID>
2674                     <saml:SubjectConfirmation>
```

```
2675 Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
2676             <saml:SubjectConfirmationData
2677             InResponseTo="Kdp1u_dqO0yM_ftaeeubgF9o0PX" NotOnOrAfter="2017-11-13T13:54:17.155Z"
2678             Recipient="https://idm.sandbox.motorolasolutions.com/sp/ACS.saml2"/>
2679                 </saml:SubjectConfirmation>
2680             </saml:Subject>
2681             <saml:Conditions NotBefore="2017-11-13T13:44:17.155Z" NotOnOrAfter="2017-
2682 11-13T13:54:17.155Z">
2683                 <saml:AudienceRestriction>
2684             <saml:Audience>ctoPingFed_entityID</saml:Audience>
2685                 </saml:AudienceRestriction>
2686             <saml:Conditions>
2687                 <saml:AuthnStatement AuthnInstant="2017-11-13T13:49:17.153Z"
2688                 SessionIndex="H_m.WHGoUQPD.3cVP41XCUXxbGK">
2689                     <saml:AuthnContext>
2690             <saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified</saml:AuthnContextClassRef>
2691                     </saml:AuthnContext>
2692             </saml:AuthnStatement>
2693             <saml:AttributeStatement>
2694                 <saml:Attribute Name="uid"
2695                 NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
2696                     <saml:AttributeValue
2697                     xmlns:xs="http://www.w3.org/2001/XMLSchema"
2698                     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2699                     xsi:type="xs:string">unccoetest4</saml:AttributeValue>
2700                     </saml:Attribute>
2701                     <saml:Attribute Name="mail"
2702                     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
2703                         <saml:AttributeValue
2704                         xmlns:xs="http://www.w3.org/2001/XMLSchema"
2705                         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2706                         xsi:type="xs:string">unccoetest4</saml:AttributeValue>
2707                         </saml:Attribute>
2708                     </saml:AttributeStatement>
2709                 </saml:Assertion>
2710             </samlp:Response>
2711         >>>
2712
2713     In the above example, two attributes, uid and mail, are asserted, but the mail attribute does not
2714     contain a valid email address.
```

2716 For OIDC, because the ID Token is retrieved over a back-channel connection between the RP and the
2717 IdP, it cannot be observed in browser traffic. As with SAML, creating a test application is one method of
2718 testing, but manual testing is also possible by using a few software tools:

- 2719 1. Register an OIDC client with a client secret and a redirect URI that points to a nonexistent
2720 server. A redirect URI value like `https://127.0.0.1/test-url` will work, assuming that you do
2721 not have a web server running on your machine. In a desktop browser, submit an authentication
2722 request with a URL like the one listed below:

2723 `https://op1.lpsd.mss0:9031/as/authorization.oauth2?client_id=marktest&response_type=code&`
2724 `scope=openid%20address%20test%20phone%20openid%20profile%20name%20email`

- 2725 2. Replace the server name and client ID with the correct values for your environment; also make
2726 sure that the scope parameter includes `openid` and any other expected scopes. Authenticate to
2727 the IdP. In this case, because the FIDO UAF adapter is in use but is being accessed through a
2728 desktop browser, it initiates an OOB authentication, which can be completed on the mobile
2729 device. Once authentication is completed, the browser will attempt to access the redirect URL,
2730 which will result in a connection error because no web server is running on localhost. However,
2731 the authorization code can be extracted from the URL:

2732 `https://127.0.0.1/test-url?code=lv-pND_3o7_aJ5nFMcD-WbrVENrW7w5V75Cupx9G`

2733 The authorization code can be submitted to the IdP's token endpoint in a POST to obtain the ID Token.
2734 There are numerous ways to do this. Postman is a simple graphical-user-interface tool for testing APIs
2735 and can be used to submit the request: <https://www.getpostman.com>.

2736 Figure 7-1 shows Postman being used to retrieve an ID Token. A POST request is submitted to the OIDC
2737 IdP's token endpoint; by default, the token endpoint URL is the base URL, followed by `/as/token.oauth2`.
2738 The authorization code is included as a query parameter. The client ID and client secret are used as the
2739 HTTP basic authorization username and password.

2740 Figure 7-1 Using Postman to Obtain the ID Token

The screenshot shows the Postman interface with the following details:

- Request URL:** https://op1.lpsd.mss0:9031/as/token.oauth2?grant_type=authorization_code&code=P0OKZK9Lw5XqTl2jnx7
- Method:** POST
- Authorization:** Basic Auth (Username: marktest, Password: [redacted])
- Body:** JSON response (copied below)
- Status:** 200 OK, Time: 138 ms, Size: 1.17 KB

```

1  {
2     "access_token": "0FmVYX6zC71rA25dHLC6MU8M48I",
3     "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6Ik13ZUVzcExQTUR5STVIME1xUnVRY18ifQ.eyJzdWIiOiJ1bmN
4       jb2V0ZXN0NCIsInVwZGF0ZWRfYXQiOjE0OTk5ODM5NzgsIm5hbWUiOiJUZXN0IDQgVU5DQ29FIiwicH
5       J1ZmVycmVkJ3VzZXJuYW11IjoiidW5jY291dGVzdDQilCJnaXZlb19uYW11IjoiVGVzdCA0IiwiZmFta
6       Wx5X25hbWUiOjVTKndb0UiLCJ1bWFpbCI6InVuY2NvZXRLc3Q0QGxwc2QubXNzbbyIsImF1ZCI6Im1h
7       cmt0ZXN0IiwiianRpIjoiMmwya1FpT1VsNW9VaG5MeUEwVUxTZiIsImlzcyI6Imh0dHBzOlwvXC9vcDE
8       ubHBzZC5tc3NvOjkwMzeiLCJpYXQiOjE1MTA1ODU4MzUsImV4cCI6MTUxMDU4NjEzNX0.G0EzK7jxXz
9       sHHMpPbCk_e_rUF3MEW9JxMxvz1WW-
10      wu0i2gQHRPZUytR2RxfghfJaCilb9LNv-HT7Jfa8LAHjkII7AmHa4QDql0ne2UMbJlchraBKuoZt3z1
11      KhftTMx10gJPVAMP9L6DwXYmG1D2zM192s7dvkB7su-
12      6A2xAxyCynH7mIFwpCaJ3Nswk0TiXNCR0Ry6j_eJ9dFd9hFYCRw0LTvzGig073h058pIe-
13      xE47r_XhjDD5GiFGuoQhmPfCKxImibUmL3H4fhx9lMe1_oG7DF4divsfo6H5TC_9UBccKf0AUdQoT2K
  
```

2741

2742 The response body is a JSON object, including the ID Token as well as an access token that can be used
 2743 to access the userinfo endpoint. As with the SAML assertion, a few lines of Python can render the ID
 2744 Token (which is a JWT) into a readable format:

```

2745   $ python
2746   Python 2.7.10 (default, Feb 7 2017, 00:08:15)
2747   [GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
2748   Type "help", "copyright", "credits" or "license" for more information.
2749   >>> import jwt
2750   >>> import json
2751   >>> idTokenStr =
2752   "eyJhbGciOiJSUzI1NiIsImtpZCI6Ik13ZUVzcExQTUR5STVIME1xUnVRY18ifQ.eyJzdWIiOiJ1bmN
2753   jb2V0ZXN0NCIsInVwZGF0ZWRfYXQiOjE0OTk5ODM5NzgsIm5hbWUiOiJUZXN0IDQgVU5DQ29FIiwicH
2754   J1ZmVycmVkJ3VzZXJuYW11IjoiidW5jY291dGVzdDQilCJnaXZlb19uYW11IjoiVGVzdCA0IiwiZmFta
2755   Wx5X25hbWUiOjVTKndb0UiLCJ1bWFpbCI6InVuY2NvZXRLc3Q0QGxwc2QubXNzbbyIsImF1ZCI6Im1h
2756   cmt0ZXN0IiwiianRpIjoiMmwya1FpT1VsNW9VaG5MeUEwVUxTZiIsImlzcyI6Imh0dHBzOlwvXC9vcDE
2757   ubHBzZC5tc3NvOjkwMzeiLCJpYXQiOjE1MTA1ODU4MzUsImV4cCI6MTUxMDU4NjEzNX0.G0EzK7jxXz
2758   sHHMpPbCk_e_rUF3MEW9JxMxvz1WW-
2759   wu0i2gQHRPZUytR2RxfghfJaCilb9LNv-HT7Jfa8LAHjkII7AmHa4QDql0ne2UMbJlchraBKuoZt3z1
2760   KhftTMx10gJPVAMP9L6DwXYmG1D2zM192s7dvkB7su-
2761   6A2xAxyCynH7mIFwpCaJ3Nswk0TiXNCR0Ry6j_eJ9dFd9hFYCRw0LTvzGig073h058pIe-
2762   xE47r_XhjDD5GiFGuoQhmPfCKxImibUmL3H4fhx9lMe1_oG7DF4divsfo6H5TC_9UBccKf0AUdQoT2K
  
```

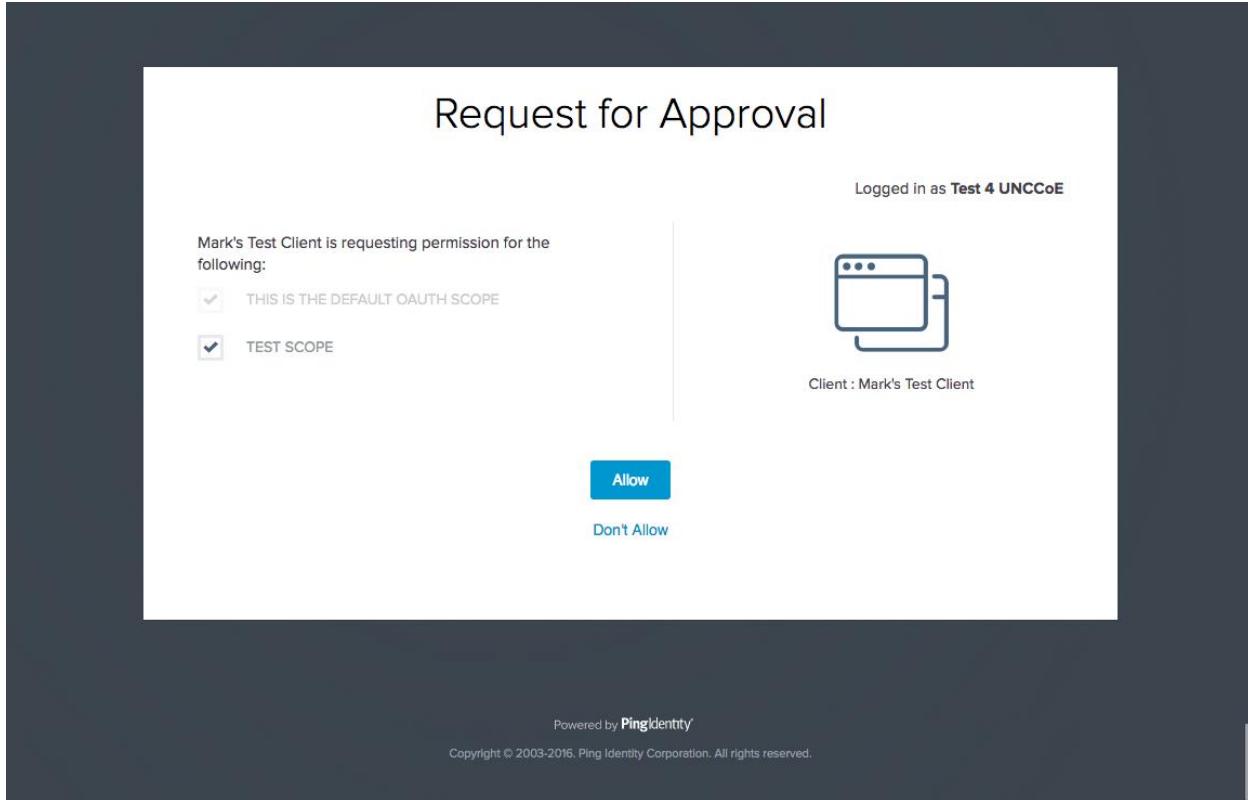
```
2763     x3PyTSYAdouYwfo6k1UYxoF-bjjfGpOg"
2764     >>> idToken = jwt.decode(idTokenStr, verify=False)
2765     >>> print json.dumps(idToken, indent=4)
2766     {
2767         "family_name": "UNCCoE",
2768         "aud": "marktest",
2769         "sub": "unccoetest4",
2770         "iss": "https://op1.lpsd.mssso:9031",
2771         "preferred_username": "unccoetest4",
2772         "updated_at": 1499983978,
2773         "jti": "212kQiNU15oUhnLyA0ULSf",
2774         "given_name": "Test 4",
2775         "exp": 1510586135,
2776         "iat": 1510585835,
2777         "email": "unccoetest4@lpsd.mssso",
2778         "name": "Test 4 UNCCoE"
2779     }
2780 >>>
```

2781 This merely decodes the claims in the JWT without verifying the signature. If there is an issue with
2782 signature validation or trust in the signing key, these errors will be reported in the PingFederate server
2783 log.

2784 **7.4 Testing the AS**

2785 One simple step that can help identify problems at the AS is turning on the authorization prompts. This
2786 can be done on a per-client basis by deselecting the **BYPASS AUTHORIZATION APPROVAL** setting on the
2787 client configuration page, in the **OAuth Settings** section in the AS console. If the authorization prompt is
2788 displayed (Figure 7-2), this demonstrates that authentication has succeeded, and the list of scopes being
2789 requested by the client is displayed and can be verified.

2790 **Figure 7-2 Authorization Prompt**



2791

2792 It is also possible to manually obtain an access token by using the same procedure that was used in the
2793 previous section to obtain an ID Token; the only difference is that an OAuth request typically would not
2794 include the `openid` scope. If the issued access token is a JWT, it can be analyzed using Python as
2795 described above.

2796 If the token is not a JWT (i.e., a Reference Token management scheme is in use), the access token can be
2797 submitted to the AS's introspection endpoint as specified in RFC 7662 [31]. The default location of the
2798 introspection endpoint for PingFederate is the base URL, followed by `/as/introspect.oauth2`. The request
2799 is submitted as a POST, with the access token in a query parameter called **token**. Basic authentication
2800 can be used with the client ID and secret as a username and password. The client must be authorized to
2801 call the introspection endpoint by selecting **Access Token Validation (Client is a Resource Server)** under
2802 **Allowed Grant Types** in the client configuration on the AS.

SECOND DRAFT

2803 Figure 7-3 shows a token introspection request and response in Postman.

2804 **Figure 7-3 Token Introspection Request and Response**

The screenshot shows the Postman application interface. On the left, there's a sidebar with a 'History' tab showing several recent API calls, and a 'Save to collection' section. The main area is titled 'Builder' and shows a POST request to 'https://as1.cpssp.mss:9031/as/introspect.oauth2?token=9lkunQlT0x71ZTaQHvtLo4bVD6i'. The 'Authorization' tab is selected, showing 'Basic Auth' with 'Username' set to 'marktest' and 'Password' set to '*****'. The 'Body' tab is selected, showing a JSON response with the following content:

```
1 - {
2   "sub": "unccotest4",
3   "scope": "testScope",
4   "active": true,
5   "token_type": "Bearer",
6   "exp": 1510597178,
7   "client_id": "marktest",
8   "username": "unccotest4"
9 }
```

2805

2806 7.5 Testing the Application

2807 One last potential problem area in this SSO architecture is the back-end application, which must accept
2808 and validate access tokens. Troubleshooting methods there will depend on the design of the application.
2809 Building robust instrumentation and error reporting into RP applications will help identify problems. If
2810 the application validates JWT access tokens, then establishing and maintaining trust in the AS's signing
2811 certificate, including maintenance when the certificate is replaced, is essential to avoid validation
2812 problems. Clock synchronization between the AS and the RP is also important; a time difference of five
2813 minutes or more can cause validation errors as well.

2814 **Appendix A Abbreviations and Acronyms**

AD	Active Directory
API	Application Programming Interface
App ID	Application Identification
AppAuth	Application Authentication System
AS	Authorization Server
BCP	Best Current Practice
BIND	Berkeley Internet Name Domain
BLE	Bluetooth Low Energy
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
FQDN	Fully Qualified Domain Name
GB	Gigabyte
GHz	Gigahertz
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identification
IdP	Identity Provider
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
LPSD	Local Public Safety Department
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
PIN	Personal Identification Number
PKCE	Proof Key for Code Exchange
PSFR	Public Safety and First Responder
PSX	Public Safety Experience
QR	Quick Response
RAM	Random Access Memory
RFC	Request for Comments
RP	Relying Party
RPM	Red Hat Package Manager
SAML	Security Assertion Markup Language
SDK	Software Development Kit
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
TLS	Transport Layer Security
U2F	Universal Second Factor
UAF	Universal Authentication Framework
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
VLAN	Virtual Local Area Network
VPN	Virtual Private Network

2815 **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] Apple, "SFSafariViewController," 2019. [Online]. Available: <https://developer.apple.com/documentation/safariservices/sfsafariviewcontroller>. [Accessed 18 March 2019].
- [6] D. Waite, "Single Sign-on and iOS 11," Ping Identity, 8 August 2017. [Online]. Available: https://www.pingidentity.com/en/company/blog/2017/08/08/single_sign-on_and_ios_11.html. [Accessed 18 March 2019].
- [7] Apple, "ASWebAuthenticationSession," 2019. [Online]. Available: https://developer.apple.com/documentation/authenticationservices/aswebauthentication_session. [Accessed 18 March 2019].
- [8] OpenID Foundation, "openid/AppAuth-iOS," GitHub, 2019. [Online]. Available: <https://github.com/openid/AppAuth-iOS>. [Accessed 18 March 2019].
- [9] Google, "Google Chrome: Fast & Secure," Google Play, 2018. [Online]. Available: <https://play.google.com/store/apps/details?id=com.android.chrome>. [Accessed 25 February 2018].
- [10] Google, "Fast Identity Online Universal Second Factor API for Android," 18 December 2018. [Online]. Available: <https://developers.google.com/identity/fido/android/native-apps>. [Accessed 18 March 2019].

- [11] Google, "Google Authenticator," Google Play, [Online]. Available: <https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2>. [Accessed 25 February 2018].
- [12] J. Chong, "iPhone Support for YubiKey OTP via NFC," Yubico, 25 October 2017. [Online]. Available: <https://www.yubico.com/2017/10/iphone-support-yubikey-otp-via-nfc/>. [Accessed 18 March 2019].
- [13] J. Chong, "Yubico Extends Mobile SDK for iOS to Lightning," Yubico, 30 August 2018. [Online]. Available: <https://www.yubico.com/2018/08/yubico-extends-mobile-sdk-for-ios-to-lightning/>. [Accessed 18 March 2019].
- [14] J. Davis, "Release Notes for Safari Technology Preview 71," 5 December 2018. [Online]. Available: <https://webkit.org/blog/8517/release-notes-for-safari-technology-preview-71/>. [Accessed 1 April 2019].
- [15] 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].
- [16] 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].
- [17] Nok Nok Labs Inc., "Nok Nok™ Passport," Apple App Store, [Online]. Available: <https://itunes.apple.com/us/app/nok-nok-passport/id1050437340>. [Accessed 18 March 2019].
- [18] Motorola Solutions, "PSX App Suite," [Online]. Available: https://www.motorolasolutions.com/en_us/products/psx-app-suite.html. [Accessed 25 February 2018].
- [19] OpenID Foundation, "openid/AppAuth-Android," GitHub, [Online]. Available: <https://github.com/openid/AppAuth-Android>. [Accessed 25 February 2018].
- [20] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage," RFC 6750, DOI 10.17487/RFC6750, October 2012. [Online]. Available: <https://www.rfc-editor.org/info/rfc6750>. [Accessed 18 March 2019].
- [21] 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].

- [22] 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].
- [23] 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].
- [24] 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].
- [25] 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].
- [26] 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].
- [27] Nok Nok Labs, Inc., "Nok Nok Labs S3 Authentication Suite Solution Guide," v5.1.1, 2017.
- [28] Nok Nok Labs, Inc., "Nok Nok Authentication Server Administration Guide," v5.1.1, 2017.
- [29] Nok Nok Labs, Inc., "Nok Nok PingFederate Adapter Integration Guide," v1.0.1, 2017.
- [30] StrongKey, Inc., "PingFederate FIDO IdP Adapter Installation Guide," Revision 2, 2017.
- [31] J. Richer, Ed., "OAuth 2.0 Token Introspection," RFC 7662, October 2015. [Online]. Available: <https://tools.ietf.org/html/rfc7662>. [Accessed 25 February 2018].