Draft NISTIR 8336

Background on Identity Federation Technologies for the Public Safety Community

William Fisher
Mark Russell*
Sudhi Umarji
Karen Scarfone

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

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8336-draft

National Institute of Standards and Technology
U.S. Department of Commerce

Draft NISTIR 8336

Background on Identity Federation Technologies for the Public Safety Community

William Fisher
Applied Cybersecurity Division
Information Technology Laboratory

Mark Russell*
Sudhi Umarji
The MITRE Corporation
McLean, VA

Karen Scarfone
Scarfone Cybersecurity
Clifton, VA

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

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8336-draft

June 2021

U.S. Department of Commerce
Gina Raimondo, Secretary

National Institute of Standards and Technology
James K. Olthoff, Performing the Non-Exclusive Functions and Duties of the Under Secretary of Commerce for Standards and Technology & Director, National Institute of Standards and Technology

National Institute of Standards and Technology Interagency or Internal Report 8336
80 pages (June 2021)

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8336-draft

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

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. Many NIST cybersecurity publications, other than the ones noted above, are available at https://csrc.nist.gov/publications.

Public comment period: *June 9, 2021* through *July 26, 2021*

National Institute of Standards and Technology
Attn: Applied Cybersecurity Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 2000) Gaithersburg, MD 20899-2000
Email: psfr-nccoe@nist.gov

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

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in federal information systems.

Abstract

This report provides the public safety and first responder (PSFR) community with a basic primer on *identity federation*—a form of trust relationship and partnership involving the verification of a claimed identity. Identity federation technologies can help public safety organizations (PSOs) to share information with each other more easily while also protecting that data from unauthorized access. Identity federation technologies can also help PSOs transition services to the cloud and facilitate the use of mobile devices such as smartphones. The intent of this report is to aid the PSFR community in adopting identity federation technologies, with different portions of the report aimed at general audiences, technically capable readers, and federation technology implementers. This report was developed in joint partnership between the National Cybersecurity Center of Excellence (NCCoE) and the Public Safety Communications Research (PSCR) Division at NIST.

Keywords

identity, credential, and access management (ICAM); identity federation; OpenID Connect; public safety organization (PSO); Security Assertion Markup Language (SAML).

Acknowledgments

The authors of this report thank all who have contributed to its content and provided feedback.

Trademark Information

Call for Patent Claims

This public review includes a call for information on essential patent claims (claims whose use would be required for compliance with the guidance or requirements in this Information Technology Laboratory (ITL) draft publication). Such guidance and/or requirements may be directly stated in this ITL Publication or by reference to another publication. This call also includes disclosure, where known, of the existence of pending U.S. or foreign patent applications relating to this ITL draft publication and of any relevant unexpired U.S. or foreign patents.

ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in written or electronic form, either:

a) assurance in the form of a general disclaimer to the effect that such party does not hold and does not currently intend holding any essential patent claim(s); or

b) assurance that a license to such essential patent claim(s) will be made available to applicants desiring to utilize the license for the purpose of complying with the guidance or requirements in this ITL draft publication either:

   i. under reasonable terms and conditions that are demonstrably free of any unfair discrimination; or
   ii. without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination.

Such assurance shall indicate that the patent holder (or third party authorized to make assurances on its behalf) will include in any documents transferring ownership of patents subject to the assurance, provisions sufficient to ensure that the commitments in the assurance are binding on the transferee, and that the transferee will similarly include appropriate provisions in the event of future transfers with the goal of binding each successor-in-interest.

The assurance shall also indicate that it is intended to be binding on successors-in-interest regardless of whether such provisions are included in the relevant transfer documents.

Such statements should be addressed to: psfr-nccoe@nist.gov

Executive Summary

Public safety organizations (PSOs) face technology challenges that hinder their ability to accomplish their missions. A report from 2015 [1] explained one of these challenges:

> "In the explosion of technology supporting public mobility and ubiquitous connectivity, law enforcement, justice, and public safety agencies have been left behind: great difficulty still exists in making the connection to the last mile...the police officer, deputy sheriff, firefighter, and paramedic in a vehicle or in the field. These professionals—our colleagues—need immediate access to critical information from the wide variety of systems technology available (particularly portable computers, tablets, and smartphones) to make the best possible decisions and protect themselves and the public. Hand in hand with access challenges is the imperative to ensure robust internal controls on security, including factoring in today's 'Bring Your Own Device' (BYOD) environment."

Today most PSOs do not have immediate access to information shared by other agencies. A primary reason for that is the lack of interoperable identities and credentials for public safety and first responders (PSFRs). When an agency is responding to a request for sensitive information from an agency in a different jurisdiction, the lack of interoperability between the information systems makes it difficult to validate the identity of the person making the information request and authorize the access.

To address these challenges, all PSOs need to improve their identity, credential, and access management (ICAM) capabilities. In a 2019 workshop conducted by the National Institute of Standards and Technology (NIST), PSO leaders and subject matter experts defined the following vision statement for identity sharing in the PSFR community:

Getting the correct data to the correct people at the correct time with the correct protections and only if it is for the proper reason and in an efficient manner.

To help achieve this, many PSOs have expressed interest in adopting identity federation technologies. These technologies enable PSOs to take advantage of identity verification services that external service providers offer. Identity federation technologies can help PSOs to share information with each other more easily while also protecting that data from unauthorized access. Identity federation usage can also reduce overhead expenses for PSOs.

This report provides the PSFR community with a primer on identity federation, which should aid PSOs in understanding and adopting identity federation technologies. Different portions of the report are written for general audiences, technically capable readers, and federation technology implementers. The report recommends that the OpenID Connect 1.0 federated authentication protocol should be the default choice for any new identity federation technology implementations, and it provides considerable technical detail in the appendices on commonly used federation protocols for readers with that level of interest.

Table of Contents

List of Appendices

List of Figures

List of Tables

1 Introduction

The public safety and first responder (PSFR) community encompasses tens of thousands of national, state, local, and tribal/territorial public safety organizations (PSOs). They face an increasing need to rapidly share information with each other, but their existing information technology (IT) can't readily support this need, and they have limited budgets for IT spending. Before they share sensitive information with other PSOs, they also need to verify the identity of the requesting party. For example, in certain situations, a police department should only release specific categories of information to other organizations that are authorized to access that information.

PSOs need to improve their identity, credential, and access management (ICAM) capabilities so that they can share information with other PSOs. In a 2019 workshop conducted by the National Institute of Standards and Technology (NIST) National Cybersecurity Center of Excellence (NCCoE) and Public Safety Communications Research (PSCR) division, PSO leaders and subject matter experts defined the following vision statement for identity sharing in the PSFR community:

> **Note:** The NIST NCCoE, through its engagement with the NIST PSCR Lab, acts as an advisory resource to the PSFR community on cybersecurity, identity management, and related topics.

Getting the correct data to the correct people at the correct time with the correct protections and only if it is for the proper reason and in an efficient manner.

To help achieve this goal, many PSOs have expressed interest in adopting identity federation technologies.

1.1 Benefits of Identity Federation

Different organizations typically run their own IT systems, either locally or in the cloud. These systems might include Computer Aided Dispatch (CAD) and other public safety applications. Each organization maintains user accounts and passwords (or other authenticators) for its own users and manages their permissions using groups, roles, attributes, or other methods. Sometimes cross-organizational collaboration requires one agency to grant access to its IT systems and data to users from another organization. The simple approach is to treat users from the partner organization like the agency's own users by creating user accounts and passwords for them. This approach has several drawbacks both for the users and the organizations involved:

- Users now have an additional user account and password to manage. Requesting and obtaining access to the other agency's system takes time, which may impact operational efficiency.
- The organization granting access needs some way of validating the identity of users that it does not directly employ, including a way of determining the appropriate permissions for these individuals. The organization also has no direct knowledge of employee lifecycle events, like termination or changes to job duties, that require removal or modification of access. Addressing these issues requires cross-agency coordination, which increases management overhead for both organizations. Delays and inefficiencies in identity management increase the risk of unauthorized access to IT systems.

Using identity federation technologies, organizations can establish a trust relationship where user accounts in one organization are trusted by the other organization's systems. Users don't need new accounts or credentials, and the existing accounts, roles, and attributes managed by their organization can be used to control their access and privileges. The potential benefits of this approach include:

- Cost savings – identity and account management overhead is reduced for both organizations.
- Operational efficiency – individual users don't need to wait for access requests to be approved and accounts to be created by partner organizations.
- Improved security – by reducing the need for organizations to manage accounts for users outside the organization, identity federation reduces the risk of orphaned accounts and privileges.

> **Note:** Potential benefits of identity federation include cost savings, operational efficiency, and improved security.

Section 2.1 includes a more detailed discussion of the benefits of identity federation.

1.2 How to Use This Document

This report provides the PSFR community a basic primer on identity federation in order to aid PSOs in adopting identity federation technologies. Table 1 summarizes each part of the report and indicates which audiences are most likely to find each part of interest, based on the audiences' objectives:

- **General knowledge:** understand the core concepts of federation technologies. This material is appropriate for all readers, including high-level decision makers and other PSFR community members who may not have technical knowledge.
- **Technical information:** understand the technical requirements of implementing federation technologies. This material is intended for technically capable readers.
- **Technology implementation:** be prepared to implement federation technology solutions. This material provides additional technical details for readers who already have a basic understanding of the structure and syntax of JavaScript Object Notation (JSON) and Extensible Markup Language (XML).

Table 1. Report Contents Summary

Section/Appendix	General Knowledge	Technical Information	Technology Implementation
The **Executive Summary** summarizes the most important considerations for the PSFR community to understand.	✓	✓	✓
Section 1 introduces the report and describes its purpose and scope.	✓	✓	✓
Section 2 defines and explains basic identity federation concepts at a high level. This material will help PSFR community members prepare to discuss identity federation using a common vocabulary, and it describes the advantages to PSOs of adopting identity federation technologies.	✓	✓	✓
Section 3 provides an overview of identity federation technology concepts and an introduction to common identity federation technologies.		✓	✓
Section 4 takes a closer look at one identity federation protocol, the Security Assertion Markup Language (SAML) version 2.0.		✓	✓
Section 5 examines another identity federation protocol, OpenID Connect version 1.0.		✓	✓
Section 6 provides a conclusion for the report.		✓	✓
References lists all the references cited in the report.	✓	✓	✓
Appendix A provides additional information on SAML implementation that supplements Section 4.			✓
Appendix B gives an example of a SAML metadata document.			✓
Appendix C provides additional information on OpenID Connect implementation that supplements Section 5.			✓
Appendix D lists the acronyms and abbreviations used in the report.	✓	✓	✓

This report uses callout boxes to highlight certain types of information, as depicted in Figure 1. Callout boxes may contain new material that is not covered elsewhere in the report. A **Caution** box provides a warning of a potential issue with doing or not doing something. A **Definition** box provides the definition of a key term. A **Note** box gives additional general information on a topic. A **Tip** box offers advice that may be beneficial to the reader.

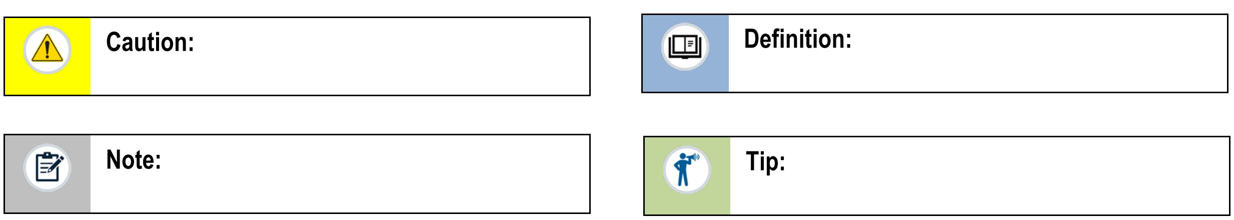

Figure 1. Callout Box Formats

2 Identity Federation Concepts

This section of the report explains basic identity federation concepts at a high level. The intent of this section is to help PSFR community members prepare to discuss identity federation using a common vocabulary, and to describe the advantages to PSOs of adopting identity federation technologies.

2.1 Basic Terminology

The term *federation* generally refers to a partnership where one partner trusts another to take care of a responsibility on its behalf. This publication examines *identity federation*, which is a form of federation involving the verification of a claimed identity. Identity federation builds on two concepts: [2]

- *Identity proofing*, which verifies that a person (a *subject*) is who they are claiming to be. During identity proofing, the person to be proofed is called an *applicant*. If proofing succeeds, the person is then called a *subscriber* and is issued a *credential* that associates them with an *authenticator* (such as a password). The credential provides a form of *digital identity*.
- *Digital authentication*, which verifies that a subject attempting to access a digital service is in control of one or more valid *authenticators* associated with that subject's digital identity.

> **Definition:** *Identity federation* is "a process that allows the conveyance of identity and authentication information across a set of networked systems." [3]

Generally speaking, in the US most PSFR organizations have not implemented identity federation. Although a local emergency medical services (EMS) department and a local police department (PD) may be well-integrated in terms of operations, this is achieved through human-to-human contact and PSFRs knowing who to call, rather than rapidly enabling access to necessary and critical information based on the digital identity and role of the PSFR.

Let's look at an example to illustrate what identity federation offers and what the alternatives are. Suppose that Paramedic John Doe, who was recently hired by his local EMS department, has been dispatched to provide medical assistance to someone at a residence. Ideally, Paramedic Doe should be warned before entering the premises if the local PD has a record of a violent or armed resident at the location. If the EMS and the PD have separate IT systems, there are three ways in which Paramedic Doe could get information concerning personal risk before entering the residence:

1. **Manually (no computer service)**. Without using any computer services, dispatch could contact someone in the PD (e.g., by radio, by phone) and ask them to look up the history of the address and tell them if there are any risks. Such a process could be time consuming and take away precious resources from the mission.
2. **Non-federated computer service.** Without federation, the PD could enroll all EMS employees, including Paramedic Doe, and issue each of them a digital identity and authenticators upon joining EMS. EMS employees like Paramedic Doe could use the PD-

issued authenticators to access the PD's application, which would alert them of risks at their assigned dispatch locations. In order for this to work, the PD would have to enroll and issue digital identities and authenticators for every firefighter, paramedic, and other PSFR members who might need such access. PSOs would have to take on the responsibility of identity governance and administration to manage digital identities and authenticators for users who are not members of their own organization.

3. **Federated computer service**. With federation, Paramedic Doe could undergo enrollment once at EMS and receive a single digital identity and authenticators that could be used to access applications not only with the EMS, but also at the PD and any other PSO participating in the federation. Figure 2 shows a simplified federated environment. Paramedic Doe, the subscriber, is on the left side. The subscriber wants to use their web browser (*user agent*) to access the PD's application. As part of federation, the application has a trust relationship with an *identity provider* (IdP), which takes responsibility for enrolling subscribers, issuing and managing their credentials, and directly authenticating them for the application by verifying their credentials. Because the PD's application relies on the IdP for these services, the application is termed a *relying party (RP)*. The use of federation enables Paramedic Doe to quickly access PD's systems and determine whether or not it's safe to enter the premises in question.

> **Definition:** An *identity provider* is a federation participant that issues and manages user credentials, authenticates users, and provides assertions to relying parties. [2]

> **Definition:** A *relying party* is a federation participant that accepts assertions from an identity provider. [2]

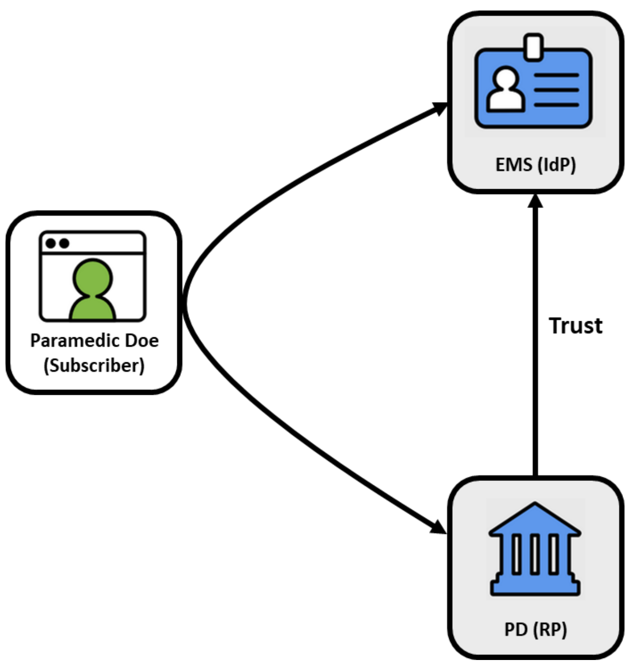

Figure 2. Federation Participants

2.2 Federation Protocols

Continuing the Figure 2 example, when the subscriber's user agent (Paramedic Doe's web browser) tries to authenticate the subscriber to the RP (the PD's application), the RP orchestrates interaction between the user agent and the IdP. The IdP takes care of the authentication on behalf of the RP, then provides information to the RP like the subscriber's identifier, authentication status, and type of authenticator used. These pieces of information are called *statements* or *claims*. The set of statements that the IdP provides to the RP regarding an authentication attempt is known as an *assertion*.

Definition: An *assertion* is a set of statements or claims about the user or an authentication event that an IdP provides to a RP. [2]

A *federation protocol* is a specification that defines what messages the participants in a federation scheme should send each other and how those messages should be structured, composed, protected, and processed. Message exchange specifics vary among federation protocols, but generally they involve the RP sending an authentication request to the IdP, followed by a series of interactions that end with the IdP sending an assertion to the RP. Errors may occur that cause the message exchange to end before an assertion is issued. Federation protocols typically allow for different *protocol flows*, or variations on the sequences or format of messages that make up a single protocol transaction.

Definition: A *federation protocol* is a specification that defines the structure, content, processing, and protection of messages between federation participants. [2]

As Figure 3 shows, federation protocols send messages through one of two paths: the front channel and the back channel. The *back channel* refers to the IdP and RP communicating directly with each other. The *front channel* refers to the IdP and RP communicating with each other indirectly through the subscriber's user agent. The user agent is not the originator or final recipient of any messages within the federation protocol; it is only a passive participant.

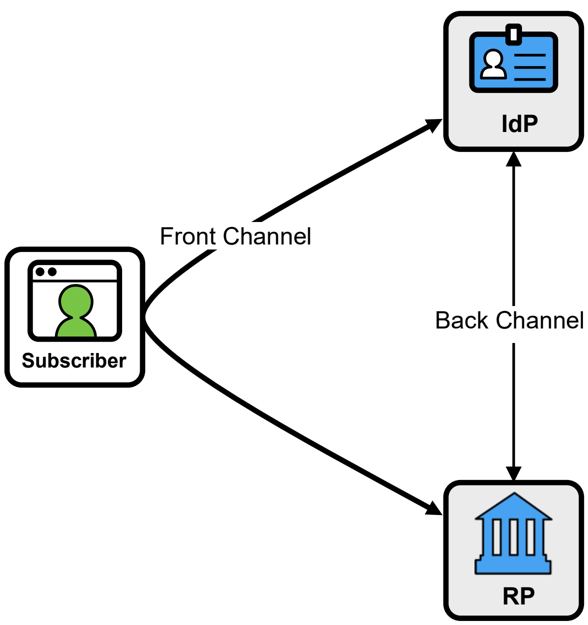

Figure 3. Front Channel and Back Channel

Assertions can contain different types of attributes. *Authentication attributes* provide information about the subscriber's authentication to the IdP—for example, when it occurred or what type of authenticator was used. *Subscriber attributes* provide information about the subscriber, such as identifiers or contact information (e.g., phone numbers or email addresses). Subscriber attributes also could include information about a subscriber's role or authorities, such as whether the subscriber is a sworn law enforcement officer.

> **Definition:** The *back channel* is a direct communications channel between the IdP and RP. The *front channel* is an indirect communications channel between the IdP and the RP that uses the user agent (typically a browser) to pass messages. [2]

The RP can use information in the IdP's assertion to decide whether or not to allow the subscriber to use the RP application. Each assertion contains *metadata*, which provides the RP with information about the assertion itself, such as which IdP issued it, which RP it was issued to, when it was issued, and when it expires. An assertion normally includes several metadata elements. Table 2 lists the assertion metadata elements that IdPs must include in assertions according to NIST's Digital Identity Guidelines [3].

Table 2. Assertion Metadata

Metadata Element	Description
Issuer	An identifier for the IdP that issued the assertion
Issuance	A timestamp indicating when the IdP issued the assertion
Audience	An identifier for the RP intended to use the assertion
Expiration	A timestamp indicating when the assertion expires and must no longer be accepted as valid by the RP
Identifier	A value uniquely identifying this assertion; used to prevent an attacker from reusing a prior assertion
Signature	Digital signature or message authentication code (MAC) for the entire assertion; used to verify the integrity of the assertion
Subject	An identifier for the subscriber whom the assertion is about
Authentication Time	A timestamp indicating when the IdP last authenticated the subscriber

2.3 Federation Participant Responsibilities

> **Definition:** Trust relationships between identity federation participants include administrative and/or legal aspects (the responsibilities and expectations of each organization) as well as technical aspects (federation protocol parameters and cryptographic keys).

An RP and an IdP must establish a trust relationship with each other before they can participate in federation. Trust relationships include technical aspects like agreeing on the details of the federation protocol to use, exchanging cryptographic keys, configuring service endpoint locations, and establishing lines of communication between the RP and IdP technical support teams to ensure that issues will be handled effectively. They also include administrative concerns like defining the expectations and responsibilities of each party. In some cases, written administrative and legal agreements may be required. Trust relationships among a community of organizations may be formalized in a *trust framework* (see Section 3.2).

Table 3 summarizes the typical responsibilities of IdPs and RPs in a federation trust relationship.

Table 3. Typical RP and IdP Responsibilities

RP Responsibilities	IdP Responsibilities

• Implementing the RP aspects of the federation protocol
• Expressing its authentication requirements to the IdP when initiating a federated authentication interaction
  ○ For example, an RP may be required by policies or regulations to strongly authenticate users (e.g., with multi-factor authentication) or to reauthenticate users before they perform highly sensitive actions
• Consuming the assertions issued by the IdP
  ○ Includes validating each assertion and extracting the user identifiers from it
• Maintaining any required profile or account for the subscriber in the RP app according to local requirements
	• Implementing the IdP aspects of the federation protocol
	• Authenticating users
	  ○ Either acting as a credential service provider (CSP), which issues credentials to subscribers, or leveraging another CSP (for example, accepting authenticators issued by other CSPs)
	• Maintaining information about subscribers, such as their identifiers, attributes, and authenticator bindings; this information is often kept in one or more user directories or databases
	• Acting as a verifier by requiring subscribers to demonstrate possession and control of an authenticator
	• Issuing assertions to the RP

NIST's Digital Identity Guidelines also describe the role of the Credential Service Provider (CSP), an entity that issues and manages authenticators and digital credentials for subscribers. IdPs also commonly perform the role of CSP, issuing credentials that subscribers can use to authenticate to the IdP in a federated login flow.

Some IdPs may also have trust relationships with other IdPs. For example, IdP A could authenticate a subscriber on behalf of IdP B and use a federation protocol to issue an assertion to IdP B. IdP B could then use information from the assertion to create its own assertion, which it would send to its RP through another federation protocol flow.

In most cases, an RP obtains subscriber attributes by asking the IdP for them and receiving the attributes in an assertion as part of the federation protocol. An RP can use subscriber attributes for various purposes, such as deciding which actions a particular subscriber should be authorized to do. However, in some environments, RPs obtain subscriber attributes outside of the authentication flow by sending attribute queries to a separate *attribute provider*. For example, an organization that provides training services such as active shooter response training might provide assertions that given individuals have completed their training. Figure 4 illustrates an RP app that uses an IdP to authenticate users and also obtains user attributes from an attribute provider (labeled "AP" in the figure).

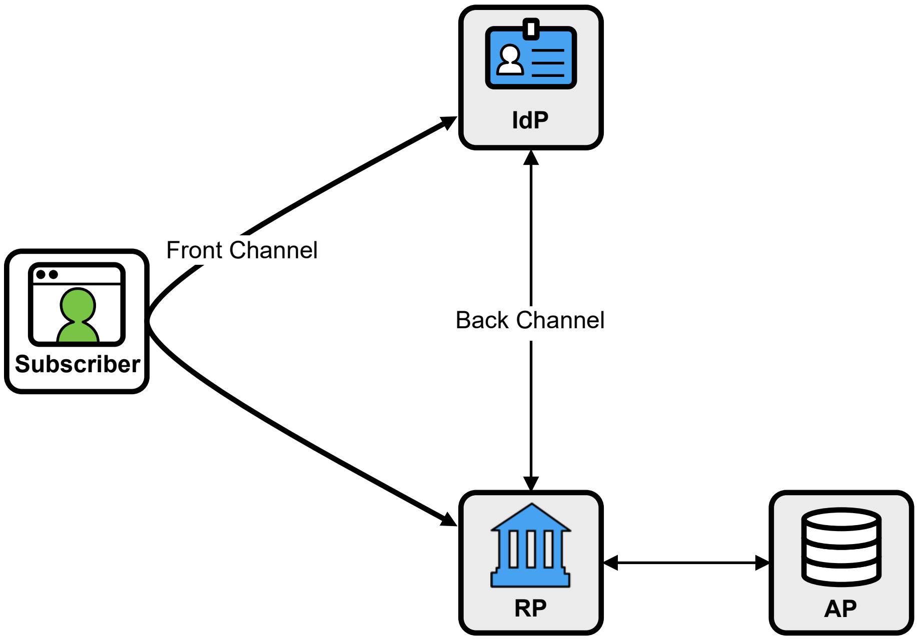

Figure 4. Federation with Attribute Provider

This report generally addresses the simpler case where attributes are obtained from the IdP and a separate attribute provider is not used.

2.4 Federation Benefits

Using identity federation can benefit both the subscriber and the RP.

- **Subscriber benefits**:
  - **Fewer credentials.** Since subscribers typically interact with a large number of applications and a comparatively small number of IdPs, identity federation reduces the number of credentials the subscriber must maintain and manage since unique credentials are not required by each RP.
  - **Fewer authentications**. When a user authenticates to an IdP using an authenticator, a session is created with a defined lifetime. If the user later attempts to access another RP while the session is still active, the user will be redirected to the IdP but will not be required to reauthenticate because of the established session. In most cases the IdP will not display a user interface and the user will seamlessly transition into the RP app, having already authenticated. This is a form of single sign-on (SSO) and provides a more convenient user experience than a non-federated environment where the user would explicitly authenticate to each RP.
- **RP benefits**:
  - **Efficiency**. The RP does not need to perform identity proofing, credential management, and authentication, which can be costly and time consuming.
  - **Flexibility**. As new authentication technologies and authenticator types are brought to market, the IdP is the only system that needs to be modified in order to allow the RPs to utilize them.

- **Separation of concerns**. RP application developers can focus on core application functionality without having to implement credential management and account recovery functions that require specialized expertise to implement correctly. These functions are implemented by the IdP, which is purpose-built to handle them.
- **Auditability and accountability.** The RP can use the assertions from the IdP to keep an audit log of accesses to its digital services, including who accessed them, when, and where. This can help enable the RP to conduct auditing to look for individuals who are making unauthorized queries, such as viewing someone's records for personal reasons.

2.5 IdP Discovery

An RP frequently needs to interact with multiple IdPs. Cloud service providers, for example, frequently use federation to authenticate their customers' users by redirecting each user to an IdP managed by the user's organization. When a user attempts to access the RP application, the application must determine which IdP should be used to authenticate that user. Another common scenario is an enterprise application that serves both internal users and external users from partner organizations. Internal users may be authenticated by the organization's own IdP, but users outside the organization might need to be redirected to the corresponding partner organization's IdP. Ensuring that the RP contacts the correct IdP for each user can be complicated and necessitate customizing an application.

Many strategies exist for RPs to select the appropriate IdP, a process called *IdP discovery*. Some common strategies for IdP discovery include:

- **Prompting the user to select an IdP** – This approach is most feasible when the RP uses a relatively small number of IdPs. The Office of Management and Budget (OMB) MAX website's login page, as seen in Figure 5,[1] uses this approach by providing a selection of government agency logos and names. From these, users pick their "home" agency icon, which redirects them to the corresponding IdP.

[1] Image source: https://login.max.gov/

Figure 5. OMB Max IdP Selection Interface

- **Asking the user for their user identifier** – Many Software as a Service (SaaS) providers prompt the user for an email address and use the domain portion (the part after the ‘@’) to identify the user’s IdP. This approach is feasible for any number of IdPs; however, it depends on users having and knowing a user identifier that will enable the RP to unambiguously determine the correct IdP for the user. Using email addresses, for example, relies on the assumption that all users within the same email domain can be authenticated by a single IdP, which may not be true in all cases.
- **Automatically identifying the IdP** – There are approaches that automatically identify the IdP without relying on input from the user. One example is having intermediate devices, like network gateways or proxies, insert headers into the users’ requests that signal to the RP which IdP should be used. Another example is having a mapping of client IP addresses to IdPs, but this is impractical in most cases.

3 Identity Federation Technical Concepts

This section explores several technical concepts involving identity federation technology. It also introduces and compares two common identity federation technologies: Security Assertion Markup Language (SAML) and OpenID Connect.

3.1 Federation Protocols

Table 4 compares several characteristics of the two most commonly implemented identity federation protocols, SAML 2.0 and OpenID Connect 1.0. Each protocol has advantages over the other, and it is likely that both will continue to be used for the next several years. However, in most cases where an organization is planning a new identity federation technology implementation and backwards compatibility with an existing SAML or SOAP (formerly an acronym for Simple Object Access Protocol) web service infrastructure is not required, OpenID Connect should be the default choice.

In many cases, PSOs will not have to choose between SAML and OpenID Connect. Most authentication software products and Identity as a Service (IDaaS) providers can support both protocols side by side. This is important since organizations may need to integrate with RP applications that support SAML for the foreseeable future.

Table 4. Comparison of Selected SAML 2.0 and OpenID Connect 1.0 Characteristics

	SAML 2.0	OpenID Connect 1.0
Underlying technologies	Older technologies, including: • Extensible Markup Language (XML) • SOAP	Newer technologies, including: • JavaScript Object Notation (JSON) • Representational State Transfer (REST) • OAuth 2.0
Ongoing development	There have been no updates to the core SAML specifications since 2005, except for errata corrections.	The core OpenID Connect specification was finalized in 2014. Numerous draft extension specifications and working groups are currently active.
Complexity	• The SAML specifications are complex. • SAML implementation involves several layers of constructs, including protocols, bindings, and profiles. • A developer using SAML must interpret and reconcile the requirements at each layer in terms of how they apply to a specific use case, like web SSO.	• The OpenID Connect specification is comparatively simple. • OpenID Connect focuses on a single use with fewer deployment options. • The OpenID Connect core specification is sufficient to implement the protocol in many cases.
Extensibility	• SAML supports a wide range of options. • SAML is adaptable to different transport protocols and environments.	• OpenID Connect only supports the web SSO use case over a single transport protocol, with stated assumptions about the environment in which the protocol operates.

	SAML 2.0	OpenID Connect 1.0
Security implementation	• Security measures like digital signatures and authentication are usually optional because the wider context and environment in which the protocol flow will occur is undefined. • The flexibility SAML provides in applying security measures has created issues such as signature wrapping attacks [4] that can bypass signature verification and cause a SAML RP to accept modified SAML assertions. • Assertions can optionally be encrypted using XML Encryption.	• With fewer options and a smaller set of security decisions for developers to make, OpenID Connect leaves less room for oversights and errors in implementation. • The JSON Web Signature (JWS) proposed standard used by OpenID Connect is much simpler and easier for developers to implement correctly than the XML Signature standards used in SAML. • Assertions can optionally be encrypted using JSON Web Encryption (JWE).
Mobile app support	Designed before the advent of the iPhone, SAML is not well suited to mobile apps, and integration is difficult.	OpenID Connect is routinely used in mobile apps; libraries are readily available to developers.

3.2 Trust Frameworks

Identity federation is a tool for providing authentication and identity services across partner organizations, with benefits to both users and application providers. For many use cases like social media and e-commerce, the only identity information required is the association of a user with a specific email address. For higher-assurance use cases, however, RPs may be bound by regulatory or legal requirements dictating that information can only be released to individuals meeting specific criteria. When an RP uses information in an assertion from an IdP to make authorization decisions, the RP needs some assurance that the information in the assertion is reliable, accurate, and timely. This requires knowing that the IdP exercises due diligence in managing user information and has the appropriate security and management controls in place to protect the integrity of its systems. Conversely, an organization that shares sensitive data with an RP system needs assurances that its data will be protected from compromise while held by that system and not released or shared inappropriately.

A *trust framework* is an agreement among participants in an identity federation ecosystem that specifies the rights and responsibilities of participants and the policies and procedures that govern participation in the federation [5]. Participants agree to be bound by the rules of the trust framework and may be audited for compliance. The policies of a trust framework might include requirements around the identity proofing of users, issuing and managing credentials, privacy and security, data handling, and interoperability with specific identity standards and profiles.

> **Definition:** A *trust framework* is an agreement among participants in an identity federation ecosystem that specifies the rights and responsibilities of participants and the policies and procedures that govern participation in the federation.

Trust frameworks can also support scalability as federations grow to include many participants. When two organizations agree to implement identity federation, they may institute a bilateral agreement covering the policy, security, and other considerations affecting trust. If a large number of organizations is involved in a federation, establishing bilateral agreements between each pair of participants becomes infeasible.

3.2.1 Existing Public Safety Trust Frameworks

The National Identity Exchange Federation (NIEF) is an example of a trust framework that serves the PSFR community. NIEF members include the Texas Department of Public Safety, the Federal Bureau of Investigation, Nlets (The International Justice and Public Safety Network), and other federal, state, and local public safety organizations. The NIEF website [6] publishes the NIEF trust framework policies, technical specifications, and governance framework. NIEF also publishes the NIEF Trust Fabric, a machine-readable, cryptographically signed file managed by the NIEF governance board containing federation parameters and keys for its members. The trust fabric enables NIEF members to dynamically establish trust relationships between their systems while providing assurance that all entities included in the trust fabric are NIEF members. Additional examples of trust frameworks can be found in NIST Interagency Report (IR) 8149, *Developing Trust Frameworks to Support Identity Federations*.

> **Definition:** A *trustmark* is a cryptographically signed document attesting to an organization's conformance to a defined standard.

NIEF also makes use of *trustmarks*, which are cryptographically signed documents that attest to an organization's conformance to a defined standard. As defined in the Trustmark Framework Technical Specification [7], a *trustmark issuer* issues a trustmark based on an evaluation of an organization's compliance with a set of requirements in a *trustmark definition*. The Trustmark Initiative has published numerous trustmark definitions based on established security policies, including NIST Special Publication (SP) 800-53 and the Criminal Justice Information Services (CJIS) Security Policy. Trustmarks can be "bound" to a specific federation participant through inclusion in federation metadata documents like the NIEF Trust Fabric. By including trustmark bindings in its signed trust fabric document, the NIEF provides assurance that the trustmark is associated with a specific system identified in the trust fabric. The trustmark itself can also be cryptographically verified to assure its authenticity and integrity.

3.3 Message Security

Section 2 introduced the concepts of front and back channels, the paths federation protocol messages are carried over. All federation protocols use the front channel because the IdP needs to interact directly with the subscriber's user agent in order to perform authentication, and this is only possible in the front channel. Some federation protocol flows use the front channel only, especially if the IdP and RP cannot directly connect to each other. Other federation protocol flows use both the front and back channels.

Because messages sent through the front channel are exposed to the subscriber's user agent and computing platform, there are important distinctions between the front and back channels in terms of security:

- Front channel messages are exposed to the user agent and could be read or manipulated by the subscriber or a software process running on the subscriber's client (e.g., malware). Though they may be protected in transit with Transport Layer Security (TLS) on both "legs" of the route, they are decrypted while being processed by the user agent.
- Message-level encryption can be used to protect message confidentiality in the front channel, provided the recipient's public key is available to the sender.

- The RP and IdP can authenticate to each other when communicating through the back channel using static secrets or public-key cryptography (e.g., mutually authenticated TLS). They cannot authenticate to each other at the connection layer when using the front channel. They can send signed messages through the front channel, which provides message-layer authentication and integrity protection. Since the connection layer is unauthenticated, these signed messages may still be subject to interception, replay, and hijacking by an unauthorized party.

3.4 Assertion Bindings and Assurance Levels

Assertion binding refers to a mechanism for associating an assertion with the authenticated subscriber who is authorized to present it to an RP. There are two types of assertion bindings [2]:

- *Bearer assertions* can be presented by any party and accepted as proof that the bearer of the assertion is the subscriber without further verification by the RP. As with a library card without a photo or other means of verifying the borrower, an unauthorized party who obtains a bearer assertion can present it to an RP and potentially impersonate the subscriber.

> **Definition:** *Bearer assertions* are presented to relying parties without any additional proof that the party presenting the assertion is the subject of the assertion.
>
> *Holder-of-key* assertions require the presenter of the assertion to prove possession of an associated cryptographic key.

- *Holder-of-key assertions* include a reference to a cryptographic key possessed by the subscriber. When presented with a holder-of-key assertion, the RP requires the presenter to prove possession of the key referenced in the assertion with a digital signature (e.g., by signing a cryptographic challenge). By verifying the signature, the RP can determine that the presenter of the assertion possesses the private or symmetric key referenced in the assertion. An unauthorized party who intercepts a holder-of-key assertion and presents it to an RP will be unable to meet the proof-of-possession requirement unless they have also compromised the corresponding key.

NIST SP 800-63-3 [3] defines three sets of assurance levels for aspects of digital identity:

- **Identity Assurance Levels (IALs)** apply to identity proofing.
- **Authenticator Assurance Levels (AALs)** apply to authenticators and authentication protocols.
- **Federation Assurance Levels (FALs)** apply to federation protocols. The three defined FAL values are shown in Table 5.

> **Definition:** A *Federation Assurance Level* "describes requirements for how assertions are constructed and secured for a given transaction." [2]

Table 5. Federation Assurance Levels

FAL	Assertion Binding Type	Requirements
FAL1	Bearer	Signed by IdP
FAL2	Bearer	Signed by IdP and encrypted to RP
FAL3	Holder-of-key	Signed by IdP and encrypted to RP

All three FALs (including FAL1) require the IdP to sign assertions, which is a critical security control in identity federation. The IdP's digital signature prevents attackers from creating their own assertions or modifying legitimate assertions.

FAL2 adds the requirement to encrypt the assertion with a key associated with the RP. This protects the confidentiality of the assertion and any personally identifiable information (PII) or other sensitive information it contains, and it mitigates the risk of assertions being replayed or redirected to RPs that are not the intended recipients, since only the intended RP can decrypt the assertion.

FAL3 includes the requirements of FAL2 plus the requirement to use holder-of-key assertions as described above, with the implied requirement that the subscriber must prove possession of the key when presenting the assertion to the RP.

> **Note:** The holder-of-key requirement included in FAL3 is extremely difficult to meet. The notion of holder-of-key has existed for over a decade, but actual implementations of holder-of-key are extremely rare. It has been commented that FAL3 was intended to be aspirational, as it is not widely achievable at present.

3.5 Federation and Direct Authentication

Most common identity federation use cases involve the authentication of users. Typically, this requires the user to authenticate directly to the IdP using an authenticator such as a password or cryptographic key. Federation standards like SAML and OpenID Connect do not specify how direct authentication is performed or what type of authenticator should be used. This provides the flexibility to introduce new authenticators and authentication schemes into identity federation implementations without the need to modify the federation standards themselves.

Federation protocols include mechanisms to allow the IdP to convey information about the direct authentication event to the RP in the assertion, such as the authenticator assurance level or the specific authenticator used. Sometimes the user may have an active session with the IdP from an earlier authentication and not need to authenticate again for a federated login to a RP application. IdPs can also convey the time when authentication actually occurred.

In some cases, the RP may need to specify direct authentication requirements to the IdP. The RP application may require multifactor authentication or authentication at a specific AAL, or the RP application may need to ensure that the IdP directly authenticates the user again (as opposed to allowing them to resume an existing session through SSO).

Some applications may require "step-up" authentication, an access control policy where access to sensitive functions or data within an application requires a higher AAL than general access to the application. For example, say a user accesses an RP application after having authenticated to the IdP with username and password. This low-assurance authenticator is adequate for some of the functions of the RP application. When the user attempts to access a sensitive function that requires AAL-2, the RP application redirects the user back to the IdP. A parameter in the authentication request to the IdP indicates that AAL-2 authentication is required, so the IdP prompts the user to authenticate with credentials that meet those requirements. Federation protocols provide the needed capabilities for RP applications to implement this form of step-up authentication.

3.6 Other Federation Security Considerations

Because of their role in user authentication, identity federation technologies should be considered critical components of an organization's security infrastructure. A compromised identity provider, for example, can be used to create arbitrary assertions and impersonate users to relying party systems. This potential risk became a reality in a recent, widespread cyberattack on numerous government and commercial organizations, as detailed in the National Security Agency (NSA) Cybersecurity Advisory report *Detecting Abuse of Authentication Systems* [8]. The report describes an attacker technique of compromising the cryptographic keys used to sign SAML assertions and using them to forge assertions, enabling them to impersonate legitimate users to applications and services. Agencies that run their own IdP services must ensure that any software vulnerabilities are promptly addressed and that cryptographic keys are protected from compromise. The NSA report includes specific recommendations for protecting IdPs. RP applications also must consider federation-specific security concerns. One example is the potential for a compromised or malicious partner IdP to enable the impersonation of internal or privileged users. This can generally be addressed by associating federated user identifiers with the IdP that issued their assertions rather than relying solely on the user identifier provided by the IdP.

Federation protocols can be complex and implementation decisions can have security impacts. Organizations deploying identity federation technology should reference the security guidance provided in the federation standards themselves and leverage community-defined standards profiles, such as the Federal Identity, Credential, and Access Management (FICAM) SAML Profile or the iGov or Financial-Grade API (FAPI) profiles of OpenID Connect. These profiles constrain implementations of the standards, typically by either mandating or prohibiting optional features and controls, to meet specific security and interoperability requirements. Agencies should also refer to Section 8 of NIST SP 800-63C for a more complete discussion of federation security concerns, threats, and mitigation strategies.

3.7 Implementation Considerations

When implementing identity federation technology, agencies should evaluate their requirements and plan out their architecture to ensure it meets their current and future needs. Some organizations may take a deliberate, strategic approach to federation capabilities, while others may need to quickly deploy identity federation capabilities to meet short-term information sharing needs. In either case, agencies should consider some basic factors in designing their solutions. Table 6 provides a list of considerations and questions to help guide the design of an identity federation deployment.

Table 6. Identity Federation Implementation Considerations

Areas to Consider	Notes
Federation roles	Does your organization need to act as an IdP, a RP, or both?
Protocols and flows	Which protocols and protocol flows do you need to implement? • What kinds of RP applications do you need to integrate with? SAML may be easier to integrate with SOAP-based web services and applications, whereas OpenID Connect may be preferable for REST-based applications or mobile apps. • Can the IdP and RP communicate directly with each other, or are there firewalls preventing direct connections? If direct connections are not possible, the implementation will be limited to flows that only use the front channel (e.g., the SAML Web Browser SSO profile or the OpenID Connect Implicit flow). Refer to NIST SP 800-63 and applicable security guidance for your chosen federation protocol to understand the implications of sending assertions in the front channel. • Apart from federated login, are other protocol flows needed (such as attribute query or single logout)?
Enterprise integration	What enterprise ICAM services will your identity federation systems need to integrate with? • An IdP system will need access to enterprise directory services to authenticate users and obtain their attributes. • RP applications may also benefit from integration with an enterprise authentication system that handles the federation protocol, rather than having each application implement RP functionality directly. • Both RP and IdP systems require common enterprise security functions like cryptographic key management and the ability to obtain trusted certificates.
On-premises, cloud, or hybrid deployment	Agencies have the option of installing and maintaining their own identity federation infrastructure either in a data center or in a cloud Infrastructure-as-a-Service (IaaS) hosting environment. In addition, authentication and identity federation services are available in a SaaS model in the form of Identity-as-a-Service (IDaaS) offerings. Traditional software products and IDaaS services often support both IdP and RP functionality. Agencies may also consider hybrid deployments, where some components of the identity federation solution (e.g., the identity provider interfaces) are cloud-hosted while others (e.g., enterprise directory and authentication services) remain on-premises. There are different benefits to these approaches, and numerous other factors beyond the scope of this document should be considered, including the technical capabilities and budget of the organization and its security, privacy, and authentication requirements such as multi-factor authentication (MFA). Agencies should consult NISTIR 8335, *Identity as a Service (IDaaS) for the Public Safety and First Responder Community* [9] for an analysis of IDaaS services for public agencies.
Existing trust frameworks	Can your organization benefit from joining an existing trust framework, such as NIEF? This may facilitate integration with other current federation members.
Authorization and attributes	For RP applications, how will you assign permissions, roles, and/or privileges to partner organization users? For IdPs, what information about your users do RPs need for authorization to their systems? Are there commonly understood attributes that can be used (e.g., Sworn Law Enforcement Officer)? Do all participants in the federation use common values and definitions for these attributes? Are attributes from third-party attribute providers needed?
Ongoing management	Periodic management and maintenance tasks should be planned, staffed, and accounted for. These might include: • Updates to partners' metadata – when federation participants rotate their signing or encryption keys (which should occur at regular intervals) or change URLs or other federation parameters, their counterparts must update their own systems' configurations to enable the federation trust relationship to continue to function. • Periodic evaluation of trust relationships – agencies should review existing trust relationships periodically to ensure they are still needed and appropriate.

Areas to Consider	Notes
Identity management	One advantage of identity federation is that it reduces administrative overhead by eliminating the need for relying parties to manage identities and accounts for federation partner users. However, some RP applications may continue to require the creation of accounts for federated users. Account creation may be manual (by administrator action), dynamically at authentication time (through the automatic creation of accounts for federated users), or out-of-band through an automated account synchronization process. These processes are typically application-specific, so RP applications owners should assess the application's identity management requirements and put processes in place to ensure that "orphaned" user accounts and privileges are removed from the system in an appropriate timeframe.

4 SAML 2.0

SAML 2.0 is a language for expressing assertions based on XML. As a markup language, SAML is agnostic to transports and protocols; SAML messages can be exchanged over a variety of mechanisms. SAML messages consist of different types of *requests* and *responses* typically pertaining to the need to authenticate and/or obtain information about a *subject*, which could be a person or a computer system. SAML's core functionality enables an *asserting party* to provide *assertions* about a subject to an *RP*.

The SAML 2.0 specifications define several aspects of SAML, including the following:

- **Assertions** (Section 4.1) – the structure and content of SAML assertions and the statements they contain
- **Metadata** (Section 4.2) – additional information about SAML participants
- **Protocols** (Section 4.3) – SAML request and response types for specific interactions, such as federated authentications or attribute queries
- **Bindings** (Section 4.4) – guidance on using SAML messages and protocols over specific transports, such as SOAP or Hypertext Transfer Protocol (HTTP) redirects
- **Profiles** (Section 4.5) – guidance on using SAML for specific use cases, such as web single sign-on (SSO)

In addition to maintaining the human-readable specification documents for SAML, the Organization for the Advancement of Structured Information Standards (OASIS) also provides XML schema definitions. They enable automated validation that SAML messages have the structure, data elements, and data formats required by the SAML specification.

Caution: This information about the SAML specifications provided in this section is up-to-date as of the time of writing, but standards may be updated at any time. Consult the OASIS SAML Wiki for the most current versions of the SAML specifications: https://wiki.oasis-open.org/security/FrontPage#SAML_V2.0_Standard

4.1 SAML Assertions

The SAML assertions and protocols specification [10], also referred to as *SAML Core*, defines SAML assertions.[2] An assertion is typically conveyed in a SAML response, and it contains a set of statements about the assertion subject. SAML Core defines three types of assertion statements:

- ***Authentication*** – The subject successfully authenticated to the IdP (along with associated information about the authentication event).
- ***Attribute*** – The given attributes are associated with the subject.
- ***Authorization Decision*** – The subject's request to access given resources should be granted or denied.

[2] OASIS also provides an XML schema defining SAML assertions associated with the "urn:oasis:names:tc:SAML:2.0:assertion" namespace.

SAML's assertion schema is also extensible so custom assertion statement types can be defined.

In XML terms, the assertion element type is a complex type that contains other mandatory and optional elements and attributes. "Attribute" is used here in the XML sense, meaning an attribute associated with an XML element that is contained in the element's opening tag. In the following example, the XML element "element1" has attribute "attribute1" and contains element "element2."

```
<element1 attribute1="true">
  <element2>value</element2>
</element1>
```

Table 7 lists the elements and attributes supported by the SAML assertion type. Element names are contained in angle brackets to distinguish them from attributes.

Table 7. Elements and Attributes of the SAML Assertion Type

Element / Attribute	Required / Optional	Description
Version	Required	The version of the SAML specification to which the assertion conforms (e.g., "2.0" for SAML 2.0 assertions).
ID	Required	A unique identifier for the assertion.
IssueInstant	Required	When the assertion was created, expressed in Coordinated Universal Time (UTC).
<Issuer>	Required	The IdP that made the assertion.
<ds:Signature>	Optional	A cryptographic signature for the assertion to protect the assertion's integrity. The signature must conform to the XML Signature standard [11]. There is no general requirement to sign assertions because a signature may be provided by an outer data layer, such as the SAML response containing the assertion or a signed SOAP envelope containing the SAML response. Certain use cases may require a signature to be used.
<Subject>	Optional	The user or computer system to which the assertion pertains.
<Conditions>	Optional	Logical conditions that the RP must evaluate before making use of the assertion. Examples: • The validity time period of the assertion • The audience to which the assertion is meant to be presented • A statement that the assertion is valid for one-time use
<Advice>	Optional	Additional information about the assertion that may assist in processing; unlike <Conditions>, <Advice> may be ignored if the RP does not understand it or does not wish to make use of it.
<Statement>	Optional (zero or more)	Assertion statements of the following types: • <AuthnStatement> - an authentication statement • <AttributeStatement> - an attribute statement • <AuthzDecisionStatement> - an authorization decision statement • <Statement> - a statement of a custom type that is defined in an extension schema

SAML assertions may optionally authenticate the issuer and/or may be encrypted using the XML encryption standard [12] to protect the confidentiality of the assertion.

4.2 SAML Metadata

The SAML metadata specification [13] defines a standard format for an XML document to identify SAML participants and provide information about their supported roles, endpoints, configuration, cryptographic keys, and other technical details. Systems participating in SAML exchanges in any capacity—IdPs, RPs, etc.—can express their configuration in metadata. SAML metadata documents and individual parts of them can be signed using the XML Signature standard [11].

Definition: *SAML metadata* is a standard XML format to identify and provide information about SAML participants and their configuration.

Metadata documents are often used to facilitate configuring trust relationships between SAML systems, and most implementations can at least partially automate the configuration of these connections by ingesting the required parameters from a partner system's metadata. The use of metadata documents in establishing trust relationships is not required; however, setting up such relationships without ingesting metadata requires a great deal of manual configuration.

Table 8 lists some of the key elements and attributes included in SAML metadata documents. Appendix B shows an example metadata document from the SAML metadata specification.

Table 8. Key Elements and Attributes in SAML Metadata

Element / Attribute	Description
<EntityDescriptor>	The root element describing a SAML entity. Includes the SAML Entity ID, role descriptors, and all other data elements pertaining to the entity.
<Organization>	Optional element identifying the organization responsible for the SAML system.
<RoleDescriptor>	Abstract type from which the specific role descriptors (such as <IDPSSODescriptor>) are derived. Includes the protocolSupportEnumeration attribute, which provides a set of Uniform Resource Identifiers (URIs) identifying the SAML protocols supported by the entity.
<KeyDescriptor>	Provides information about cryptographic keys used for XML signature or encryption. May optionally provide public keys or an indirect reference to them. Public keys may also be exchanged out of band and excluded from metadata.
SSODescriptorType	Abstract type from which the other SSO descriptor types are derived. Includes optional elements describing the entity's supported service endpoints – <ArtifactResolutionService>, <SingleLogoutService>, and <ManageNameIDService>. Also defines supported NameID.
<IDPSSODescriptor>	Extends SSODescriptorType with additional service endpoint definitions for the IdP role: <SingleSignOnservice>, <NameIDMappingService>, and <AssertionIDRequestService>. Also defines the attributes supported by the IdP. May specify that <AuthnRequests> must be signed.
<SPSSODescriptor>	Extends SSODescriptorType for the service provider (SP) role, including <AssertionConsumerService> and <AttributeConsumingService> service descriptors. Can specify whether the SP will sign <AuthnRequests> and request that assertions sent from IdPs be signed.
<AttributeConsuming Service>	Defines a service provided by an SP and the specific attributes that are requested or required to be provided by IdPs in response to <AuthnRequests>.

Element / Attribute	Description
<AttributeAuthority Descriptor>	Extends SSODescriptorType for systems that respond to AttributeQuery requests. Provides service endpoint descriptors <AttributeService> and <AssertionIDRequestService> and information about the attributes supported by the attribute authority.

4.3 SAML Protocols

SAML protocols typically consist of specific types of requests and corresponding responses, though in some cases an IdP may send a response without having first received a request. The design of XML request and response types uses the XML concept of inheritance, where general types are defined with basic attributes and features which are then extended by more specific types that inherit the features of the general classes and add elements required for their specific functions. For example, all SAML requests are based on the RequestAbstractType. Its elements and attributes, shown in Table 9, are common to all SAML requests. Specific types of SAML requests, such as AuthnRequests, extend the basic RequestAbstractType by adding the elements and attributes needed to describe a specific type of request.

Definition: *SAML protocols* define SAML requests and responses for RPs and IdPs to use for a specific function, like authenticating a user or obtaining attributes.

Table 9. Elements and Attributes of the SAML RequestAbstractType

Element / Attribute	Required / Optional	Description
ID	Required	A unique identifier for the request.
Version	Required	The version of the request; “2.0” for SAML 2.0.
IssueInstant	Required	The time the request was created in UTC.
Destination	Optional	A URI reference indicating where the request is to be sent; intended to prevent malicious forwarding of the request to other recipients.
Consent	Optional	Indicates whether consent of the subject was obtained; sample values include “Obtained,” “Prior,” and “Implicit.”
<saml:Issuer>	Optional	The identifier of the entity that generated the request.
<ds:signature>	Optional	A signature of the SAML request generated according to the XML signature specification.
<Extensions>	Optional	Custom extensions to the message format that are agreed upon by communicating parties.

A SAML response is encoded in a Response element, which has the attributes and elements listed in Table 10:

Table 10. Elements and Attributes of the SAML ResponseType

Element / Attribute	Required / Optional	Description
ID	Required	A unique identifier for the response.
InResponseTo	Optional	Identifier of the request corresponding to the response. Must be included if the response answers a request and must be omitted otherwise.
Version	Required	The version of the request; “2.0” for SAML 2.0.
IssueInstant	Required	The time the response was created in UTC.
Destination	Optional	A URI reference indicating where the response is to be sent; intended to prevent malicious forwarding of the response to other recipients.
Consent	Optional	Indicates whether consent of the subject was obtained; sample values include “Obtained,” “Prior,” and “Implicit.”
<saml:Issuer>	Optional	The identifier of the entity that generated the response.
<ds:signature>	Optional	A signature of the SAML response generated according to the XML signature specification.
<Extensions>	Optional	Custom extensions to the message format that are agreed upon by communicating parties.
<Status>	Required	A complex type that conveys information about the status of the request, including a status code and optional message and details elements. SAML Core defines several status codes for success and error conditions such as user authentication failure, invalid attribute name, SAML version mismatch, and numerous codes related to specific SAML protocols.
<Assertion>	Optional (zero or more)	An <Assertion> element as described in Section 4.1.
<EncryptedAssertion>	Optional (zero or more)	An <EncryptedAssertion> element as described in Appendix A.2.4.

SAML supports digital signatures on both SAML requests and responses, but they are optional because SAML can be deployed over a number of different transports and protocols. These transports and protocols may already provide authentication and integrity protection at a lower layer. Additional security requirements may be imposed by a given SAML binding or profile.

SAML Core defines the following SAML protocols. See Appendix A for more details about them.

- The **Authentication Request Protocol** implements the most common SAML use case, federated authentication. A requester (which is typically the RP) authenticates itself to the IdP and presents a SAML authentication request. The IdP authenticates the subject and returns a SAML response that contains an <AuthnStatement> element. The response may also include <AttributeStatement> elements or other statements about the subject.
- The **Assertion Query and Request Protocol** provides a means for RPs to request assertions from an IdP outside the context of an authentication flow. For example, a user may have authenticated directly to an application, but during that authenticated session

the application needs to obtain trusted user attributes from an authoritative source to make an authorization decision.

- SAML provides a mechanism for sending SAML requests and responses by reference rather than by value. In place of a SAML request or response element, the RP or IdP instead sends a small piece of data called an *artifact*. The artifact contains information enabling the recipient to determine which entity generated it, and the **Artifact Resolution Protocol** can be used to exchange the artifact for the full SAML request or response.
- The **Name Identifier Management Protocol** enables an IdP or an RP to notify its counterparts that a subject's name identifier has changed.
- The **Single Logout Protocol** defines a LogoutRequest message that can be sent by a session participant (an RP) or a session authority (an IdP). When a session authority initiates a logout (or receives a LogoutRequest from a session participant), it sends LogoutRequests to all other session participants to which it has provided assertions during the current session.

4.4 SAML Bindings

The request and response formats and protocols defined in SAML Core are agnostic to the transport protocol used to carry the messages. The SAML bindings specification [14] defines how SAML messages can be bound to common transport protocols like SOAP and HTTP in an interoperable way. Each binding is associated with a unique URI and identifies requirements for participant authentication, message integrity and confidentiality, potential error conditions, and security considerations specific to that binding.

Most SAML bindings are "composable" with each other, meaning a complete SAML message exchange can use multiple bindings. An RP might send a SAML request using the HTTP Redirect binding, and the IdP might respond using the HTTP POST or HTTP Artifact binding. The selection of bindings is constrained by both recipients' support for them (advertised in SAML metadata) and in some cases by support for optional features like RelayState data (see Appendix A.4.1 for a description of RelayState).

> **Definition:** *SAML bindings* specify how SAML messages are conveyed over transport protocols like SOAP and HTTP. This is not to be confused with the concept of "assertion bindings" discussed in Section 3.4.

SAML uses a combination of front and back channel bindings. The SAML specifications also use the terms *asynchronous* and *synchronous* to refer to the front and back channel, respectively.

The following are SAML bindings defined in SAML core. For more details about these, see Appendix A.

- **HTTP Redirect binding**: SAML messages are transported between a SAML requester and a SAML responder through the front channel as HTTP URL query parameters. The HTTP Redirect binding can be initiated by any SAML requester, including an SP application requesting user authentication through web SSO, an IdP sending a single logout request, or any other SAML actor initiating a message flow that supports the Redirect binding.

- **HTTP POST binding:** Like the HTTP Redirect binding, the HTTP POST binding uses the browser as an intermediary to pass messages between the RP and the IdP. Instead of submitting the SAML message and other parameters in the URL query string, the POST binding uses an HTML form to cause the browser to submit the parameters in the request body. This mitigates the message length concerns associated with the URL, since browsers and servers are designed to accommodate message bodies of arbitrary length.
- **HTTP Artifact binding:** This binding defines two methods for sending a SAML artifact in place of a SAML message to a recipient. The two methods are similar to the HTTP redirect and HTTP POST bindings. The sender can URL-encode the artifact and include it in a URL query string parameter named SAMLart in an HTTP redirect, or it can return an HTML form with a hidden SAMLart field containing the artifact.
- **SOAP binding**: SAML interactions over SOAP use a simple request-response model. The SAML requester sends a SAML request element as the sole contents of the SOAP body. The body may not contain more than one SAML request or any other XML elements outside of the SAML request. Similarly, the responder sends a SOAP message in reply that contains only a single SAML response in the SOAP body. The SOAP binding is synchronous.
- **PAOS binding**: PAOS is used between the client and a SAML requester (typically a Service Provider in the Enhanced Client or Proxy [ECP] profile). It enables the client to act as the intermediary in a SAML message exchange over SOAP between the SAML requester and a SAML responder.

Tip: The most commonly used bindings are HTTP Redirect and HTTP POST.

4.5 Standard SAML Profiles

Definition: *SAML profiles* specify how SAML protocols and bindings can be used to support a specific use case, like web browser SSO. These profiles are defined by OASIS and are distinct from the community profiles like FICAM and FAPI referenced in Section 3.6.

The SAML profiles specification [15] provides a set of profiles that tie together SAML protocols and bindings for specific use cases like SSO, and gives guidance on the use of attributes for specific types of attribute information or environments. A key aspect of profiles is imposing limitations on the broad optionality of the SAML specifications to enable interoperability within a specific scope or use case. Given the extreme range of options available in SAML, profiling is a necessity for interoperability among implementations.

Some of the profiles defined in the specification, like the Artifact Resolution Profile and the Assertion Query/Request Profile, do not add significant content beyond the corresponding protocol definitions, so they are not discussed here. The profiles of most interest for this report are as follows:

- The **web browser SSO profile** is the most commonly used SAML profile, supporting federated authentication and SSO for browser users. It uses the SAML Authentication Request protocol and supports the HTTP Redirect, HTTP POST, and HTTP Artifact bindings. In the context of SSO-related profiles, the relying party is referred to as a *service provider (SP)*. The IdP provides an SSO service endpoint, which receives and processes <AuthnRequest> messages. The SP provides an Assertion Consumer Service endpoint, which receives SAML response messages from the IdP. The IdP may determine the location of the SP's assertion consumer endpoint through its metadata, or the SP may specify the intended endpoint in the <AuthnRequest>.

- The **enhanced client or proxy (ECP) profile** targets web SSO use cases for clients other than web browsers. The intended use cases at the time the profile was developed included desktop thick-client applications and Wireless Application Protocol (WAP) proxies used by cellular network carriers to enable the pre-smartphone mobile devices of the day, which did not have full-featured web browsers, to access web content hosted on the internet. Today, the PAOS binding could be used to support devices with limited user interfaces like set-top boxes or smart televisions.
- The **single logout profile** specifies how the single logout protocol is used among IdPs and SPs to propagate logout events to multiple systems involved in a SAML federated login scheme. As described in Appendix A.3.5, IdPs and SPs each perform their own local session management once authentication (whether direct at the IdP or indirect at the SP) has succeeded and a user session is established. Single logout enables a user or an administrator to cause a logout event at either an IdP (a session authority) or an SP (a session participant) to propagate to other systems to which the user has been authenticated using SAML during the current session. IdPs may be both session authorities and session participants in cases where proxied SAML authentication is used to authenticate the user. In practical terms, SAML single logout is challenging to implement.

Tip: The SAML web browser SSO profile describes the most common use of SAML – authenticating users to web applications.

4.6 Summary of SAML Terminology

Table 11 summarizes important SAML terminology. Different terms may be used to refer to the participants in a SAML message exchange depending on context. Some terms describe actors at a high conceptual level, while others are used in reference to specific protocols or profiles, so multiple terms sometimes apply to an actor in a given message flow at different levels of abstraction. Terms like "relying party" and "service provider" that can be used interchangeably in some contexts but not others are frequently confused. Some of these terms, like "assertion," have general meanings beyond the context of SAML, but their SAML-specific definitions are included here.

Table 11. SAML Terminology

Term	Definition
Assertion	A piece of data produced by a SAML authority regarding an act of authentication performed on a subject, attribute information about the subject, or authorization data applying to the subject with respect to a specified resource.
Attribute	A distinct characteristic of a subject. Attributes are often represented as pairs of "attribute name" and "attribute value(s)."

Term	Definition
Asserting Party	Formally, the administrative domain that hosts one or more SAML authorities. Informally, an instance of a SAML authority.
Federated Identity	A principal's identity is said to be federated between a set of providers when there is an agreement between the providers on a set of identifiers and/or attributes to use to refer to the principal.
Identity Federation	The act of creating a federated identity on behalf of a principal.
Identity Provider	A kind of service provider that creates, maintains, and manages identity information for principals and provides principal authentication to other service providers within a federation, such as with web browser profiles.
Principal	A system entity whose identity can be authenticated.
Relying Party	A system entity that decides to take an action based on information from another system entity. For example, a SAML relying party depends on receiving assertions from an asserting party (a SAML authority) about a subject.
Requester	A system entity that utilizes the SAML protocol to request services from another system entity.
Responder	A system entity that utilizes the SAML protocol to respond to a request from another system entity.
SAML Authority	An abstract system entity in the SAML domain model that issues assertions.
Service Provider	A system entity that receives and accepts authentication assertions in conjunction with an SSO profile of SAML.
Session Authority	A role taken on by a system entity when it maintains state related to sessions, as in the SAML Single Logout profile.
Session Participant	A role taken on by a system entity when it participates in a session with a session authority, as in the SAML Single Logout profile.
Subject	A principal about which assertions are made.

An identity provider is also an asserting authority and a SAML authority, may be a session authority and a session participant (if it supports SAML single logout), and is at times a SAML requester and a SAML responder—but a SAML authority is not necessarily an identity provider. All relying parties are service providers, but not all service providers are relying parties. These subtle distinctions have frequently caused confusion. The SAML Glossary [16] provides additional definitions beyond those in Table 11.

5 OpenID Connect 1.0

OpenID Connect 1.0 is a federated authentication protocol standardized by the OpenID Foundation. OpenID Connect is not a revision of the older OpenID 2.0 standard, but a completely different protocol based on the OAuth 2.0 Authorization Framework. OAuth 2.0 is an adaptable framework for delegated authorization that is commonly used to authorize client requests to Representational State Transfer (REST) application programming interfaces (APIs). OpenID Connect is a profile of OAuth 2.0 tailored to provide federated authentication services.

> **Caution:** The information about the OpenID Connect specifications provided in this section is up-to-date as of the time of writing, but standards may be updated at any time. Consult the OpenID Foundation's website for the most current versions of the OpenID Connect specifications: https://openid.net/developers/specs/

5.1 OpenID Connect Terminology

OpenID Connect introduces different terms for federation participants than those used in SAML. An OpenID Connect IdP is called an *OpenID provider (OP)*; the relying party is an *OpenID client*, or simply a *client*. However, the terms *IdP* and *RP* are also commonly used to refer to OpenID Connect participants.

> **Definition:** *OpenID provider* is the term used for an identity provider in the OpenID Connect standard. *OpenID client* is the OpenID Connect term for a relying party.

OpenID Connect inherited ideas from SAML, and some of the authors of the original SAML specifications are also contributors to OpenID Connect. OpenID Connect is under active development as of this writing. Also, in addition to the Core working group, other OpenID working groups are developing draft specifications for specific industries and user communities including healthcare, finance, and mobile network operators.

5.2 OpenID Connect Assertions

The primary assertion format in OpenID Connect, defined in the OpenID Connect Core specification [17], is called an *ID token*. ID tokens are encoded as JSON Web Tokens (JWTs). The ID token is signed using JSON Web Signature (JWS) and may optionally be encrypted using JSON Web Encryption (JWE).

> **Definition:** An *ID token* is the assertion format used by OpenID providers.

OpenID Connect also defines the optional userinfo endpoint, an alternative mechanism for the OP to return claims to the client. When the userinfo endpoint is used, the OP issues an access token which the client can use to request user claims through a REST interface.

OpenID Connect Core defines a standard set of required and optional claims, shown in Table 12. The ID token can include additional claims to contain arbitrary attributes and other data about the user as needed for specific applications.

Table 12. Standard ID Token Claims for OpenID Connect

Claim Name	Required / Optional	Description
iss	Required	Identifier of the ID token issuer, formatted as an HTTP Secure (HTTPS) Uniform Resource Locator (URL).
sub	Required	Subject identifier - an identifier for the authenticated user; locally unique within the issuer and never reassigned.
aud	Required	Intended audience for the ID token. Contains the RP's client_id and may contain additional identifiers for other audiences.
exp	Required	Expiration time for the ID token; RPs must not accept expired tokens.
iat	Required	Time at which the ID token was issued.
auth_time	Conditional	Time at which the user authenticated to the IdP; required if a max_age request is made.
nonce	Conditional	String value used to associate a client session with an ID token, and to mitigate replay attacks. If the client submits a nonce parameter in the authentication request, the IdP must include it in the ID token.
acr	Optional	Authentication Context Class Reference; e.g., could be used to convey the AAL of the user's authentication to the IdP.
azp	Optional	Authorized party, the party to which the ID token was issued. Needed only when the aud value is different from the authorized party.

5.3 OpenID Clients

OAuth and OpenID Connect clients can be divided into two types: confidential and public clients. Table 13 compares the two types.

Table 13. Comparing Confidential and Public Clients

	Confidential Clients	Public Clients
Security Properties	Can protect secrets, like passwords or private keys	Lack secure storage to protect passwords or private keys
Examples	• Server-side web applications	• Javascript-based web applications that run locally in the web browser • Desktop "thick client" applications and native mobile apps, where individual instances of the client software run on end users' devices

A trust relationship between an OpenID client and an OpenID provider is established through a registration process. During client registration, client credentials (in the form of client secrets or public keys) are associated with confidential clients. Public clients do not use client credentials, since they lack any effective means of protecting them. For example, if a native mobile app available in the public app store included a client secret, anyone who downloaded the app could use software tools to extract the secret.

5.4 OpenID Connect Protocol and Authentication Flows

The OpenID Connect Protocol is a profile of OAuth 2.0, providing additional parameters and functions while also constraining the wide range of OAuth options to suit federated authentication use cases. An OP is also an OAuth 2.0 Authorization Server, and it may perform both OpenID Connect and OAuth functions in a single interaction.

> **Note:** OpenID Connect is related to OAuth 2.0. OpenID Connect is a profile of OAuth 2.0, meaning that it specifies how to use OAuth for a specific purpose (identity federation). OAuth is a general framework that serves many different purposes, most of which involve delegated authorization. OAuth can be used in conjunction with both SAML and OpenID Connect. See NIST SP 1800-13, *Mobile Application Single Sign-On*, for a detailed practice guide discussing the integration of OAuth with SAML and OpenID Connect for mobile apps.

Like OAuth 2.0, OpenID Connect supports different authentication flows. All flows begin with an authentication request from a client to the OP. The response_type parameter in the request identifies the specific flow requested by the client. OPs are not required to support all authentication flows and may reject requests for unsupported flows. All of the flows follow the same high-level process:

1. The client submits an authentication request to the OP.
2. The OP authenticates the user and optionally prompts the user to consent to the federated login and sharing their identifier and requested user profile information or other attributes.
3. The OP returns an ID token to the client.

The OpenID Connect *authentication request* is a specific type of OAuth authorization request and is sent to the OP's authorization endpoint. The request may be submitted as an HTTP GET (with parameters encoded in the URI query string) or POST (with parameters serialized as HTML form parameters). The parameters of an authentication request are shown in Table 14.

Table 14. OpenID Connect Authentication Request Parameters

Parameter	Required / Optional	Description
scope	Required	The OAuth 2.0 scope parameter. The "openid" scope value indicates that the request is an OpenID Connect authentication request. Other scope values may be included. Scopes can be used to request access to specific user attributes.
response_type	Required	Determines the authentication flow to be used.
client_id	Required	The identifier of the RP registered at the IdP.
redirect_uri	Required	The URI to which the authentication response should be sent; must match a URI value that has been pre-registered with the IdP.
state	Recommended	Value used to maintain RP state between the request and response, similar to SAML's RelayState value. Use and verification of this value mitigates Cross-Site Request Forgery (CSRF) attacks.
response_mode	Optional	A response delivery method that can be used to override the default response mode (for example, to request the response be delivered in the URL fragment instead of the query string).
nonce	Optional	String value used to associate a client session with an ID token, used to mitigate replay attacks. If the request includes the nonce parameter, the IdP will include a nonce claim with the identical value in the ID token.
display	Optional	Conveys a preference as to how the IdP displays its user interface (e.g., in a pop-up window, with a touch-friendly interface).

Parameter	Required / Optional	Description
prompt	Optional	Determines the behavior of the user interface at the IdP. The RP can instruct the IdP not to display any user interface, force the user to reauthenticate even if an active session already exists (similar to SAML's ForceAuthn), and other options.
max_age	Optional	Specifies a maximum time since the user was last actively authenticated to the IdP. If max_age has elapsed since the last authentication, the IdP must reauthenticate the user. The use of max_age also requires the IdP to include the auth_time claim in the ID token.
ui_locales	Optional	Specifies the user's language preferences.
id_token_hint	Optional	Provides an ID token previously issued by the IdP as a hint about the user's current or past authenticated session with the RP.
login_hint	Optional	Hint to the IdP about the identifier the user may want to use to authenticate. For example, if the RP prompts the user for an email address for IdP discovery, the RP can then pass the email address in the login_hint, and the IdP can pre-fill it in the username field of a login form to avoid prompting the user for it a second time.
acr_values	Optional	A set of requested authentication context class reference values indicating the RP's requirements for authentication methods to be used at the IdP. Could be used to require an authenticator with a specific AAL, like SAML's RequestedAuthnContext.
claims	Optional	Can be used to request specific claims (attributes) about the user. The claims parameter can be used to request claims that are not defined in the OpenID Connect specification and to request that specific claims be returned in the ID token or from the userinfo endpoint (described below).

> **Caution:** Some OpenID Connect request parameters, like *state* and *nonce*, have important security functions. Software developers should read applicable security guidance and ensure they use them properly to prevent attacks.

The following example shows how an authentication request can be sent in an HTTP Response from the RP to the user's browser through an HTTP redirect to the IdP, idp.example.com.

```
HTTP/1.1 302 Found
  Location: https://idp.example.com/authorize?
  response_type=code
  &scope=openid%20profile%20email
  &client_id=s6BhdRkqt3
  &state=af0ifjsldkj
  &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

OpenID Connect also supports login flows initiated by a party other than the RP. To enable this, the RP provides an optional login initiation endpoint with a parameter to indicate which IdP should be used for authentication. If the RP accepts a login initiation request, it submits an authentication request to the indicated IdP, and from there the authentication flow is the same as if it had been initiated by the RP.

The OpenID Connect specification defines the following protocol flows:

- **Authorization code flow**: Similar to the SAML web browser SSO flow when the artifact binding is used to deliver the response.

- **Implicit flow**: A flow intended for use by public client RPs; use of this flow is discouraged, and use of the authorization code flow is recommended instead.
- **Hybrid flow**: Can effectively enable the issuance of tokens separately to the front end and back end of an application.

6 Conclusion

Identity federation technologies could provide benefits to both users and application providers in the PSFR community. Users can gain the convenience of SSO and eliminate the need to manage unique credentials in multiple apps, and application providers can gain efficiencies by delegating authentication, authenticator management, and account recovery to an identity provider. Perhaps most important, the adoption of open federation standards can foster information sharing and collaboration across the PSFR community by enabling the trusted exchange of identity data and authentication services in an interoperable way.

This report recommends that the public safety community should look to OpenID Connect as the default choice for new federation implementations where SAML compatibility is not a requirement, due to the following considerations:

- The OpenID Connect specifications are simpler and easier for software developers to implement in a secure manner.
- OpenID Connect has been widely adopted by the commercial world, including cloud service providers and mobile app developers.
- The OpenID Connect specifications are undergoing continual development to meet new use cases and security requirements, whereas there has been little development activity of the SAML specifications in recent years.

The large existing base of SAML implementations and usage across the public safety community also must be acknowledged, with the implication that many PSOs will likely need to maintain SAML interoperability for several years. The community should still seek opportunities, where practical, to migrate to OpenID Connect.

1059 # References

[1] ICAM National Strategy Summit (2015) Identity, Credential, and Access Management (ICAM): Wireless Mobility in Law Enforcement, Justice, and Public Safety National Strategy Summit. (U.S. Department of Homeland Security, Washington, DC). https://www.cisa.gov/sites/default/files/publications/ICAM_Summit_Report.pdf

[2] Grassi PA, Richer JP, Squire SK, Fenton JL, Nadeau EM, Lefkovitz NB, Danker JM, Choong Y-Y, Greene KK, Theofanos MF (2017) Digital Identity Guidelines: Federation and Assertions. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-63C, Includes updates as of March 2, 2020. https://doi.org/10.6028/NIST.SP.800-63C

[3] Grassi PA, Garcia ME, Fenton JL (2017) Digital Identity Guidelines. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-63-3, Includes updates as of March 2, 2020. https://doi.org/10.6028/NIST.SP.800-63-3

[4] Somorovsky J, Mayer A, Schwenk J, Kampmann M, Jensen M (2012) On Breaking SAML: Be Whoever You Want to Be. *21st USENIX Security Symposium* (Bellevue, WA). https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final91.pdf

[5] Temoshok D, Abruzzi C (2018) Developing Trust Frameworks to Support Identity Federations. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) 8149. https://doi.org/10.6028/NIST.IR.8149

[6] National Identity Exchange Federation (2020) *Welcome to NIEF*. Available at https://nief.org/

[7] Trustmark Initiative (2021) *Trustmark Framework Technical Specification Version 1.4*. https://trustmarkinitiative.org/specifications/trustmark-framework/1.4/tfts-1.4.pdf

[8] National Security Agency (2020) *Detecting Abuse of Authentication Mechanisms.* Available at https://media.defense.gov/2020/Dec/17/2002554125/-1/-1/0/AUTHENTICATION_MECHANISMS_CSA_U_OO_198854_20.PDF

[9] Brown C, Umarji S, Russell M (to be published) Identity as a Service (IDaaS) for the Public Safety and First Responder Community. (National Institute of Standards and Technology, Gaithersburg, MD), Draft NIST Interagency or Internal Report (IR) 8335.

[10] Organization for the Advancement of Structured Information Standards (2005) *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. Available at http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[11] World Wide Web Consortium (W3C) (2013) *XML Signature Syntax and Processing Version 1.1*. Available at https://www.w3.org/TR/xmldsig-core

[12] World Wide Web Consortium (W3C) (2013) *XML Encryption Syntax and Processing Version 1.1*. Available at https://www.w3.org/TR/xmlenc-core1

[13] Organization for the Advancement of Structured Information Standards (2005) *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*. Available at http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf

[14] Organization for the Advancement of Structured Information Standards (2005) *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0.* Available at http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf

[15] Organization for the Advancement of Structured Information Standards (2005) *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0*. Available at http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf

[16] Organization for the Advancement of Structured Information Standards (2005) *Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0*. Available at http://docs.oasis-open.org/security/saml/v2.0/saml-glossary-2.0-os.pdf

[17] The OpenID Foundation (2014) *OpenID Connect Core 1.0 incorporating errata set 1*. Available at http://openid.net/specs/openid-connect-core-1_0.html

[18] OASIS Security Services Technical Committee (2020) *OASIS SAML Wiki*. Available at https://wiki.oasis-open.org/security/FrontPage

[19] Organization for the Advancement of Structured Information Standards (2006) *Security Assertion Markup Language (SAML) V2.0 Technical Overview, Working Draft 10*. Available at https://www.oasis-open.org/committees/download.php/20645/sstc-saml-tech-overview-2%200-draft-10.pdf

[20] OASIS Security Services Technical Committee (2013) *SAML 2.1*. Available at https://wiki.oasis-open.org/security/SAML21

[21] Salesforce.com (2020) *Example SAML Assertions*. Available at https://help.salesforce.com/articleView?id=sf.sso_saml_assertion_examples.htm&type=5

[22] The OpenID Foundation (2020) *Specifications*. Available at https://openid.net/developers/specs

[23] Sakimura N, Bradley J, Agarwal N (2015) Proof Key for Code Exchange by OAuth Public Clients. (Internet Engineering Task Force [IETF]), IETF Request for Comments (RFC) 7636. https://doi.org/10.17487/RFC7636

Appendix A—Additional Information on SAML Implementation

This appendix provides additional information on SAML implementation that supplements the contents of Section 4. This information is intended for readers who are familiar with XML syntax and conventions and who need more detailed information than what Section 4 provides.

A.1 SAML Specifications

SAML version 2.0 is defined in a set of standards maintained by the OASIS Security Services Technical Committee. Table 15 lists the primary SAML specifications. These specifications were approved as an OASIS standard in 2005. Updates in the form of errata have subsequently been published by OASIS, and updated working drafts are available from the SAML Wiki [18]. The Committee also produced the "Security Assertion Markup Language (SAML) V2.0 Technical Overview," [19] which describes the use cases, concepts, and architecture of SAML and provides context for the individual specifications.

Table 15. SAML Specifications

Document Title	URL
Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0	http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf
Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0	http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf
Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0	http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf
Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0	http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf
Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0	http://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf
Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0	http://docs.oasis-open.org/security/saml/v2.0/saml-conformance-2.0-os.pdf
Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0	http://docs.oasis-open.org/security/saml/v2.0/saml-sec-consider-2.0-os.pdf

SAML 2.0 is a non-backwards-compatible update to the SAML 1.0 and 1.1 specifications. The OASIS SAML Wiki refers to a proposed version 2.1 of the SAML specifications. The SAML 2.1 page [20] has not been edited since 2013, and all identified work items show a status of "not yet started." Though some additional profiles have been introduced in recent years, activity on the core SAML 2.0 specifications since 2005 has been limited to correcting identified errors.

A.2 Assertions

A.2.1 Subject Element

The following example from the SAML Technical Overview [19] shows an assertion with Issuer, Subject, Conditions, and AuthnStatement elements in which a user identified by the email address "jdoe@example.com" is asserted to have authenticated to the www.example.com IdP with a password sent over a protected transport.

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  Version="2.0"
  IssueInstant="2005-01-31T12:00:00Z">
  <saml:Issuer Format=urn:oasis:names:SAML:2.0:nameid-format:entity>
    http://www.example.com
  </saml:Issuer>
  <saml:Subject>
    <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress">
      j.doe@example.com
    </saml:NameID>
  </saml:Subject>
  <saml:Conditions
    NotBefore="2005-01-31T12:00:00Z"
    NotOnOrAfter="2005-01-31T12:10:00Z">
  </saml:Conditions>
  <saml:AuthnStatement AuthnInstant="2005-01-31T12:00:00Z"
SessionIndex="67775277772">
    <saml:AuthnContext>
      <saml:AuthnContextClassRef>
        urn:oasis:names:tc:SAML:2.0:ac:classes:
PasswordProtectedTransport
      </saml:AuthnContextClassRef>
    </saml:AuthnContext>
  </saml:AuthnStatement>
</saml:Assertion>
```

If a Subject element is included in the assertion, then all the Statement elements refer to that Subject. Subject may be omitted in cases where other elements (such as AttributeStatements) are used to identify the Subject. The Subject typically contains a NameID element using a predefined identifier format such as email address, X.509 subject name, or Windows domain qualified name. SAML also supports two forms of pseudonymous identifiers: *persistent*, meaning that the same identifier will be used in future SAML responses pertaining to the same subject, and *transient*, meaning that different transient identifiers will be used in subsequent transactions for the same subject. Pseudonyms support user privacy by reducing the ability of RPs to correlate user activities across different domains.

A.2.2 SubjectConfirmation Element

The Subject may also contain a SubjectConfirmation element that can provide a means for the RP to verify that the assertion is being presented by the intended Subject. The SAML Profiles specification defines three SubjectConfirmation methods:

- **Holder of Key** – indicates that the Subject is in possession of a cryptographic key. The RP can verify that the presenter of the assertion is the Subject through a cryptographic challenge. Information about the key is provided in the SubjectConfirmationData element.
- **Sender Vouches** – indicates that no additional information is available about the context of the assertion. The SubjectConfirmationData element may contain additional information that the RP can use to confirm the Subject.

- **Bearer** – indicates that the party presenting the assertion is the Subject. SubjectConfirmationData may include additional constraints such as a timeframe in which the assertion must be presented or the intended recipient.

A.2.3 AttributeStatement

The example below from the SAML Technical Overview [19] shows an AttributeStatement containing three Attributes. It demonstrates the use of the SAML "uri" and "basic" name formats and a custom name format defined by the "smithco" issuer. The first two attribute values are strings, but the third uses the custom smithco value type. This demonstrates the ability to associate attribute names and data types with XML namespaces and schemas. This can make XML messages extremely verbose, but it conveys information about the specific meanings of names and values in a particular context and avoids the potential ambiguity of the same attribute names being used differently by different issuers or in different contexts.

```
<saml:AttributeStatement>
  <saml:Attribute
    xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
    Name="urn:oid:2.5.4.42"
    FriendlyName="givenName">
    <saml:AttributeValue xsi:type="xs:string" x500:Encoding="LDAP">John
    </saml:AttributeValue>
  </saml:Attribute>
  <saml:Attribute
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
Name="LastName">
    <saml:AttributeValue xsi:type="xs:string">Doe</saml:AttributeValue>
  </saml:Attribute>
  <saml:Attribute NameFormat="http://smithco.com/attr-formats"
    Name="CreditLimit">xmlns:smithco="http://www.smithco.com/smithco-
schema.xsd"
    <saml:AttributeValue xsi:type="smithco:type">
      <smithco:amount currency="USD">500.00</smithco:amount>
    </saml:AttributeValue>
  </saml:Attribute>
</saml:AttributeStatement>
```

A.2.4 Encrypted Assertions

Encrypted SAML assertions use the EncryptedAssertion element, which contains an EncryptedData element consisting of the assertion encrypted as per the XML Encryption specification and zero or more EncryptedKey elements containing wrapped keys to enable decryption of the data. In the example below, taken from Salesforce's SSO Implementation Guide [21], the CipherData inside the EncryptedKey element contains a symmetric key that has been encrypted using the RP's public key and the RSA Encryption Scheme-Public Key Cryptography Standards #1 version 1.5 (RSAES-PKCS1-v1_5) algorithm. Using its private key, the RP can decrypt the symmetric key and use it with the AES-128 algorithm to decrypt the CipherData element of the EncryptedAssertion. The plaintext value should be an Assertion element as described above. The Base64-encoded CipherData values below have been truncated for readability.

```
<saml:EncryptedAssertion
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
  <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
    Id="Encrypted_DATA_ID"
    Type="http://www.w3.org/2001/04/xmlenc#Element">
    <xenc:EncryptionMethod Algorithm=
      "http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
    <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:RetrievalMethod URI="#Encrypted_KEY_ID"
        Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey"/>
    </ds:KeyInfo>
    <xenc:CipherData >
      <xenc:CipherValue>Nk4W4mx...</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
  <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
    Id="Encrypted_KEY_ID">
    <xenc:EncryptionMethod Algorithm=
      "http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
    <xenc:CipherData>
      <xenc:CipherValue>PzA5X...</xenc:CipherValue>
    </xenc:CipherData>
    <xenc:ReferenceList>
      <xenc:DataReference URI="#Encrypted_DATA_ID"/>
    </xenc:ReferenceList>
  </xenc:EncryptedKey>
</saml:EncryptedAssertion>
```

A.2.5 AuthzDecisionStatement

The AuthzDecisionStatement type is less commonly used than authentication and attribute statements; it can be used to notify an RP of authorization decisions. An AuthzDecisionStatement has a Resource attribute, indicating URLs or other resource identifiers that uniquely identify the application resources to which the decision pertains, and a Decision attribute with a value of “Permit,” “Deny,” or “Indeterminate.” The AuthzDecisionStatement contains one or more “Action” elements, which could be used to indicate the permitted HTTP verbs or simpler concepts like “read” or “write,” and optionally one or more Evidence elements that identify the assertions (by direct inclusion or by reference) used to make the authorization decision.

The SAML Core notes that the AuthzDecisionStatement feature is frozen in SAML 2.0 and no future development is planned. The note points potential users to the Extensible Access Control Markup Language (XACML) as a potential substitute.

A.3 Protocols

A.3.1 Authentication Request Protocol

The authentication request protocol implements the most common SAML use case, federated authentication. A requester (which is typically the RP) authenticates itself to the IdP and presents a SAML authentication request; the IdP authenticates the subject and returns a SAML response containing an AuthnStatement. The response may also include AttributeStatements or other statements about the subject. The authentication request protocol also introduces the concept of a *presenter*, the entity that actually conveys the authentication request to the IdP. In the web SSO use case, the relying party is the requester and the subject (or user) is the presenter.

Authentication requests use the AuthnRequest XML type, which extends the RequestAbstractType, meaning that it shares all of the required and optional attributes and elements listed in Table 9 and adds its own unique attributes and elements as shown in Table 16.

Table 16. Elements and Attributes of the SAML AuthnRequest

Element / Attribute	Required / Optional	Description
<saml:Subject>	Optional	If included, the Subject element indicates the requested Subject of the assertion. This would typically be used in cases where a claim of a specific identity has already been made (i.e., the subject has been identified) and authentication of the Subject's identity is needed. If Subject is omitted, the presenter of the request is assumed to be the requested subject (as in the common web SSO case). The Subject element may include a SubjectConfirmation element indicating requirements for how the presenter of an assertion can be confirmed to be the associated Subject.
<NameIDPolicy>	Optional	Specifies requirements for the type of subject name identifier to be asserted (e.g., email address, transient, persistent).
<saml:Conditions>	Optional	Describes the conditions the requester expects to apply to the assertion(s) that will be returned by the IdP (for example, validity period).
<RequestedAuthnContext>	Optional	Can specify authentication context requirements such as authenticating the user at a particular assurance level. This element includes one or more AuthnContextClassRef Elements, which are URI references to specific context classes or declarations, and an optional Comparison attribute specifying whether the IdP must use one of the specified classes or if they are references for comparison ("minimum," "maximum," or "better").
<Scoping>	Optional	Specifies a set of IdPs that the RP will trust to authenticate the user. Scoping is not typically used in the web SSO context.
ForceAuthn	Optional	A Boolean value. If true, it indicates that the IdP must authenticate the user and must not rely on an existing security context (e.g., an active session maintained by a cookie).
IsPassive	Optional	A Boolean flag that when true requires that the IdP not display a user interface or otherwise visibly take control of the browser session. If the user does not have an active session at the IdP, or if ForceAuthn is also true, the IdP must authenticate the user through a method that does not display a user interface, such as Kerberos authentication.
AssertionConsumerService Index	Optional	An index referencing a pre-defined location (e.g., in the RP metadata) to which the IdP must submit the response. This is an alternative to providing an explicit URL and binding with AssertionConsumerServiceURL and ProtocolBinding.
AssertionConsumerService URL	Optional	Provides a URL to which the SAML response should be sent; used in conjunction with ProtocolBinding as an alternative to AssertionConsumerServiceIndex.

Element / Attribute	Required / Optional	Description
ProtocolBinding	Optional	Specifies the SAML binding to be used at the AssertionConsumerServiceURL.
AttributeConsumingService Index	Optional	Identifies a set of assertions requested to be returned by the IdP (for example, to request specific user attributes in the SAML response). The index could point to a defined set of attributes in the requester's metadata.
ProviderName	Optional	A human-readable name that the IdP can display to the user to identify the requester.

The response to an AuthnRequest is a SAML response as described in Section 4.3. The method the IdP uses to authenticate the subject (password, X.509 client certificate, etc.) is not dictated by the SAML specification, although details about this authentication mechanism may be included in an assertion in the SAML response.

The optional parameters of the AuthnRequest enable a wide range of functionality. A minimal AuthnRequest would simply request the IdP to authenticate a user, but the RP can also optionally request a certain class of authenticator (which could be used to require a specific SP 800-63 AAL) to request specific user attributes, or to require that the user be reauthenticated rather than relying on an existing session at the IdP. Combining ForceAuthn with a high-assurance authentication class through RequestedAuthnContext, the RP could implement a step-up authentication flow for users who previously authenticated at a lower AAL.

The SAML authentication request protocol also permits the IdP to proxy the authentication request to a different IdP if needed to authenticate the presenter. The RP can restrict this proxying behavior through the Scoping element's ProxyCount attribute, which limits the number of proxies that can be used. Proxied authentication can be performed using SAML or a different mechanism like OpenID Connect. When SAML is used, the proxying IdP creates its own SAML AuthnRequest to the destination IdP, receives a response, and then creates its own response to send to the requester. Proxying is illustrated in Figure 6.

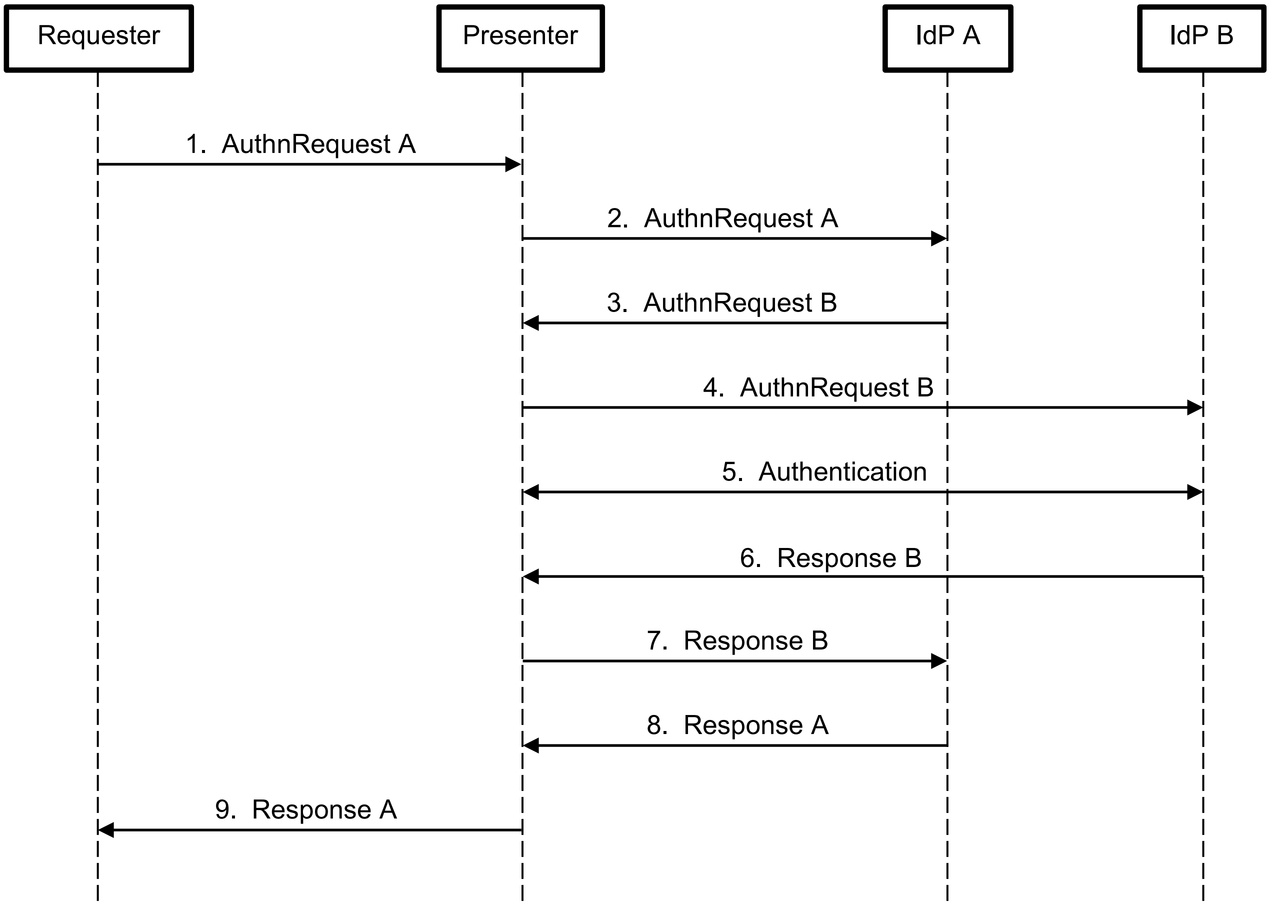

Figure 6. SAML Proxied Authentication

The sequence of steps is as follows:

1. A SAML requester submits a request (AuthnRequest A) using the presenter (which is typically a web browser in the web SSO use case).
2. The presenter passes AuthnRequest A to IdP A.
3. Because IdP A cannot directly authenticate the user, it creates a new request (AuthnRequest B) and submits it to IdP B, using the presenter as an intermediary.
4. The presenter passes AuthnRequest B to IdP B.
5. The presenter authenticates to IdP B by some supported mechanism.
6. IdP B returns response B to IdP A (again, through the presenter).
7. The presenter passes Response B to IdP A.
8. IdP A validates the response and creates its own response (Response A), which it returns to the requester through the presenter.

AuthnRequest B must be issued in accordance with the restrictions of AuthnRequest A; if AuthnRequest A specifies a RequestedAuthnContext, AuthnRequest B must request an equivalent or stricter authentication context, and the value of ProxyCount must not be exceeded. IdP A may include any relevant attribute statements from the response received from IdP B in its own response to the requester. Attribute values may be changed as needed (for example, to meet the NameID Format requirements of the original request). Response A also must include an AuthenticatingAuthority element in the AuthnContext element referencing the IdP to which the request was proxied.

Although some of the examples given in this appendix refer to the web SSO use case, the SAML authentication request protocol is agnostic to the underlying transport. HTTP is the most commonly used transport, but SOAP or any other messaging protocol—even the Simple Mail Transfer Protocol (SMTP)—could be used to convey SAML authentication requests and responses.

A.3.2 Assertion Query and Request Protocol

The assertion query and request protocol provides a means for RPs to request assertions from an IdP outside the context of an authentication flow. A user may have authenticated directly to an application, but during that authenticated session the application needs to obtain trusted user attributes from an authoritative source to make an authorization decision. SAML Core defines XML elements that are included in a SAML request to make the corresponding types of queries.

An AssertionIDRequest element can be used to request an assertion by providing the unique ID of the assertion in an AssertionIDRef element. It is assumed that an assertion has been previously generated by the IdP and the requester knows its ID attribute. This protocol can enable a client to pass a SAML assertion to a server by reference rather than including it in an application message. If a client has already obtained a SAML assertion from an IdP and needs to make a request to another system and provide the assertion as input to an authorization decision, the client can specify the Assertion ID in its request and the system receiving the request can obtain the original assertion from the IdP using the AssertionIDRequest. This type of interaction would typically occur in a web services context, where an application is interacting with other back-end systems.

The other types of queries defined by the protocol typically request information about a given subject identified in a SubjectQuery element. The following query types are supported:

- **AuthnQuery** – a request for assertions containing authentication statements for the given subject. The query may contain a RequestedAuthnContext element to filter the responses to those satisfying authentication context requirements. The IdP does not attempt to authenticate the subject before responding to the query; it simply returns any existing authentication statements based on prior authentication events.
- **AttributeQuery** – a request for attribute statements about the subject. The query may contain Attribute elements to request specific attributes, and they in turn may contain AttributeValue elements indicating that the response should only include attribute statements that have the specified values. If no Attribute statements are included, a default set of attributes is returned based on policy that has been established out-of-band by the participants.

- **AuthzDecisionQuery** – a request for an assertion containing an AuthzDecisionStatement based on the subject and other details of a system action for which authorization needs to be decided. The request must identify a resource to which access is requested and may specify an action requested to be taken against that resource and evidence (e.g., a set of SAML assertions) that should be used as input to the authorization decision. AuthzDecisionQuery is "frozen" in SAML 2.0, meaning that no further development is expected on this feature, and implementers are recommended to use XACML as a potential replacement.

The response to an AssertionIDRequest or a SAML query is a standard SAML response including one or more assertions with statements appropriate to the request content.

A.3.3 Artifact Resolution Protocol

SAML artifacts provide a mechanism for sending SAML requests and responses by reference rather than by value. In place of a SAML request or response element, the RP or IdP instead sends a small piece of data called an *artifact*. The artifact contains information enabling the recipient to determine which entity generated it, and the artifact resolution protocol can be used to exchange the artifact for the full SAML request or response.

Artifacts are used to avoid sending SAML messages over a transport where the size or sensitivity of the message is a concern. For example, SAML messages sent as HTTP request parameters can make for very long URL query strings that may be problematic in some environments, and they may be exposed to an end-user's browser or written to HTTP server logs. Using SAML artifacts mitigates these concerns, since only the artifact is sent through the front channel; the actual SAML messages are sent directly between the IdP and RP. This also reduces the need for message-level integrity protection with digital signatures, though many implementations still use signed messages with the artifact protocol.

The artifact resolution protocol defines an ArtifactResolve element that can be included in a SAML request with a specific artifact. The response to an ArtifactResolve request includes an ArtifactResponse element containing the original SAML request or response referenced by the artifact. Artifacts are restricted to one-time use and have a limited lifetime; if an artifact is reused, the responder must not return the original SAML message.

The artifact resolution protocol provides a means for a recipient to use an artifact to obtain the SAML message it references. The separate question of how the artifact is sent to the recipient is defined by the HTTP Artifact binding discussed in Appendix A.4.3.

A.3.4 Name Identifier Management Protocol

In a SAML environment, generally RPs and IdPs both maintain information about principals in the form of user profiles, databases, and directories. In some cases the RP and IdP may use the same name identifier to refer to a subject, but in others the subject may have different persistent identifiers in both systems. In either case, there is a need to maintain a mapping of identifiers to user profiles in the two systems and to properly handle name identifier changes.

The SAML name identifier management protocol enables an IdP or RP to notify its counterparts that a subject's name identifier has changed. A ManageNameIDRequest message includes a

NameID (or EncryptedID) containing the existing name identifier, and either a NewID, NewEncryptedID, or Terminate element. If an IdP sends a request containing a NewID or NewEncryptedID, this indicates that the NameID element of future assertions pertaining to that subject will contain that new ID, and the RP should make any required updates to associate the new ID with the user profile that was associated with the original ID. If the request contains a Terminate element, this generally means the IdP no longer has a relationship with the subject, and in any event the IdP will not issue future assertions for that subject. RP-submitted name identifier management protocol requests impact the SPProvidedID attribute of the NameID element, which is used to indicate a local identifier used at the RP to identify the subject and notify the IdP that either a new SPProvidedID should be used to refer to the subject, or that the identifier is no longer used at the RP.

The recipient of the name ID management request sends a ManageNameIDResponse, which is a basic SAML response containing status information but no assertions or statements.

A.3.5 Single Logout Protocol

In a SAML environment, users may have sessions established with an IdP and with multiple RPs. IdPs will generally establish a session upon successful user authentication, which enables SSO since interactive authentication will not be required (unless specifically requested by the RP) when the user attempts to access additional RPs. RPs likewise typically establish sessions upon receiving a SAML response from an IdP indicating successful authentication. Sessions are managed through HTTP cookies set by each site with which the user’s browser interacts and subject to the same-origin security policy enforced by the browser. This means that in most cases the session cookies associated with the IdP and RPs are set and managed by each participant. There is no browser-provided mechanism for any one participant to track or control sessions associated with the others.

In the context of single logout, an IdP is referred to as a *session authority* and RPs are *session participants*. The SAML single logout protocol defines a LogoutRequest message that can be sent by a session participant or a session authority. When a session authority initiates a logout (or receives a LogoutRequest from a session participant), it sends LogoutRequests to all other session participants to which it has provided assertions during the current session. An optional SessionIndex parameter in the request can be used to identify a specific session at the session authority with which participant sessions are associated. If a SessionIndex is specified, only participant sessions associated with that index should be terminated. This could accommodate use cases where only a subset of the subject’s sessions, perhaps those associated with a specific client device, should be terminated.

See Appendix A.5.3 for more details about single logout.

A.4 Bindings

A.4.1 HTTP Redirect Binding

In the HTTP Redirect binding, SAML messages are transported between a SAML requester and a SAML responder through the front channel as HTTP URL query parameters. The HTTP Redirect binding can be initiated by any SAML requester including an SP application requesting user authentication through web SSO, an IdP sending a single logout request, or any other

SAML actor initiating a message flow that supports the Redirect binding. For SAML authentication requests, the browser can display a user interface for authentication at the IdP and may facilitate access to smart cards or other cryptographic credentials. The flow of interactions in the Redirect binding is shown in Figure 7.

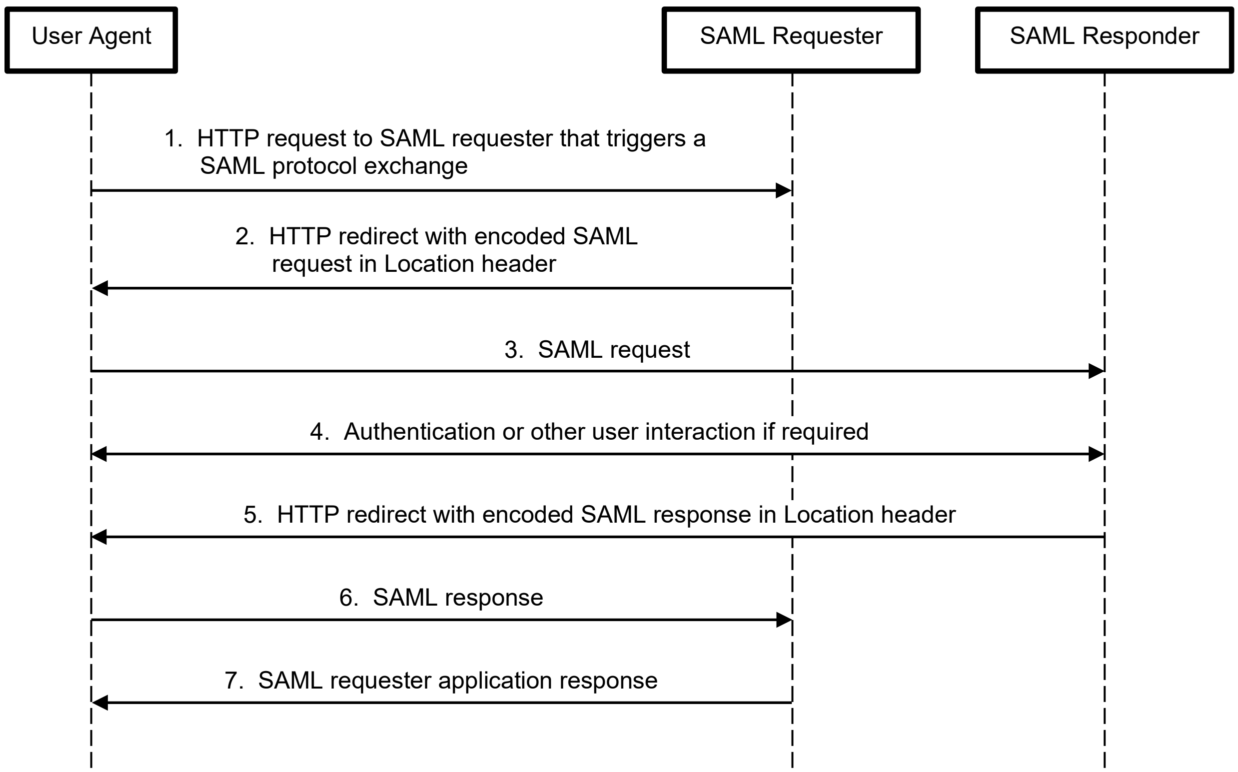

Figure 7. HTTP Redirect Binding Message Flow

The steps are as follows:

1. The user's browser sends a request to the SAML requester that triggers a SAML protocol exchange. Examples include attempting to access a protected resource without an active session triggering an authentication request, or a logout button click triggering a single logout request.

2. The requester creates a SAML request and a URL that points to an appropriate endpoint at the responder and includes the SAML request as an encoded query parameter. The requester returns an HTTP redirect response to the browser with the constructed URL in the Location header.

3. The browser follows the URL in the Location header, effectively submitting the SAML request to the responder.

4. The responder evaluates the SAML request and performs any required user interaction.

5. The responder creates a SAML response and a URL that points to a SAML endpoint on the requester and includes the SAML response as an encoded query parameter. As in step 2, the URL is returned in an HTTP redirect response.

6. The browser follows the redirect, transmitting the SAML response to the requester.
7. The requester validates the SAML response, takes any required action, and returns a response to the browser.

Although the HTTP standard does not define a maximum URL length, in practice web servers, proxies, and browsers may limit the maximum size, which renders the redirect binding unsuitable for very large SAML messages. The HTTP POST or Artifact bindings, described later in this appendix, can be used for messages too large to be conveyed by the redirect binding.

The requester may send state information such as the URL the user originally requested in a parameter called RelayState. If RelayState is sent with the request, the responder is required to return the same RelayState value with the SAML response. The RelayState value is limited to 80 bytes, so some implementations send a reference to state information stored by the requester in the RelayState value.

Encoding must be applied to SAML messages to enable them to be included in valid URLs. The redirect binding defines one encoding method called DEFLATE. As part of the DEFLATE encoding, any signature on the SAML request or response object itself must be removed. Embedded signatures within the message, such as signed assertion objects, are not removed but their use with the redirect binding is discouraged since they greatly increase message length. The SAML message is compressed, base64-encoded, and URL-encoded, then added to the URL query string with the parameter name SAMLRequest or SAMLResponse. If RelayState is used, it is URL-encoded and added to the query string with the name RelayState.

If the message is to be signed, the signature is calculated over a concatenation of the request or response, RelayState if present, and signature algorithm, and the base64-encoded signature and algorithm are also appended as query parameters. The SAML Bindings specification [14] provides the following SAML request as an example:

```
<samlp:LogoutRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="d2b7c388cec36fa7c39c28fd298644a8"
    IssueInstant="2004-01-21T19:00:49Z" Version="2.0">
    <Issuer>https://IdentityProvider.com/SAML</Issuer>
    <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent">
      005a06e0-ad82-110d-a556-004005b13a2b
    </NameID>
    <samlp:SessionIndex>1</samlp:SessionIndex>
</samlp:LogoutRequest>
```

Here is how this request would be transmitted with the redirect binding. The URL query parameters in the Location header are highlighted for readability:

```
HTTP/1.1 302 Object Moved
Date: 21 Jan 2004 07:00:49 GMT
Location:
https://ServiceProvider.com/SAML/SLO/Browser?SAMLRequest=fVFdS8MwFH0f7D
%2BUvGdNsq62oSsIQyhMESc%2B%2BJYlmRbWpObeyvz3puv2IMjyFM7HPedyK1DdsZdb%2F
%2BEHfLFfgwVMTt3RgTwzazIEJ72CFqRTnQWJWu7uH7dSLJjsg0ev%2FZFMlttiBWADtt6R
%2BSyJr9msiRH7O70sCm31Mj%2Bo%2BC%2B1KA5GlEWeZaogSQMw2MYBKodrIhjLKONU8Fd
```

```
eSsZkVr6T5M0GiHMjvWCknqZXZ2OoPxF7kGnaGOuwxZ%2Fn4L9bY8NC%2By4du1XpRXnxPc
XizSZ58KFTeHujEWkNPZylsh9bAMYYUjO2Uiy3jCpTCMo5M1StVjmN9SO150sl9lU6RV2Dp
0vsLIy7NM7YU82r9B90PrvCf85W%2FwL8zSVQzAEAAA%3D%3D&RelayState=0043bfc1bc
45110dae17004005b13a2b&SigAlg=http%3A%2F%2Fwww.w3.org%2F200%2F09%2Fxmld
sig%23rsa-sha1&Signature=NOTAREALSIGNATUREBUTTHEREALONEWOULDGOHERE
Content-Type: text/html; charset=iso-8859-1
```

Upon receiving this request, the browser will submit a GET request for the URL in the Location header, submitting the SAML message and associated parameters to the recipient.

A.4.2 HTTP POST Binding

Like the HTTP Redirect binding, the HTTP POST binding uses the browser as an intermediary to pass messages between the RP and the IdP. Instead of submitting the SAML message and other parameters in the URL query string, the POST binding uses an HTML form to cause the browser to submit the parameters in the request body. This mitigates the message length concerns associated with the URL, since browsers and servers are designed to accommodate message bodies of arbitrary length.

The message flow for the HTTP POST binding is similar to the HTTP Redirect binding flow shown in Figure 7. Instead of encoding the SAML message into a redirect response, in the POST binding the sender returns a normal success status (code 200) and an Extensible Hypertext Markup Language (XHTML) page containing a form. The form contains a hidden field called SAMLRequest or SAMLResponse which holds the base64-encoded SAML message. RelayState data can be included in a separate hidden form field if needed. The form's action attribute is the URL of the appropriate SAML endpoint at the recipient for handling the specific request or response type, and its method is POST. The XHTML page can also include JavaScript to automatically submit the form without user action; from the user experience standpoint, the transition seems no different from a redirect. The browser submits the encoded form data in the request body to the recipient.

In an example from the SAML Bindings specification [14], the following SAML message

```
<samlp:LogoutResponse
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="b0730d21b628110d8b7e004005b13a2b"
    InResponseTo="d2b7c388cec36fa7c39c28fd298644a8"
    IssueInstant="2004-01-21T19:00:49Z" Version="2.0">
    <Issuer>https://ServiceProvider.com/SAML</Issuer>
    <samlp:Status>
      <samlp:StatusCodeValue="urn:oasis:names:tc:SAML:2.0:status:Succes
s"/>
    </samlp:Status>
</samlp:LogoutResponse>
```

is encoded into the following HTTP response and XTML page. The "onload" attribute of the body element causes the form to automatically submit once the page has loaded:

```
HTTP/1.1 200 OK
Date: 21 Jan 2004 07:00:49 GMT
Content-Type: text/html; charset=iso-8859-1
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<body onload="document.forms[0].submit()">

<noscript>
<p>
<strong>Note:</strong> Since your browser does not support
JavaScript,you must press the Continue button once to proceed.</p>
</noscript>

<form action=https://IdentityProvider.com/SAML/SLO/Response
    method="post">
<div>
<input type="hidden" name="RelayState"
value="0043bfc1bc45110dae17004005b13a2b"/>
<input type="hidden" name="SAMLResponse"
value="PHNhbWxwOkxvZ291dFJlc3BvbnNlIHhtbG5zOnNhbWxwPSJ1cm46b2FzaXM6bmFt
ZXM6dGM6U0FNTDoyLjA6cHJvdG9jb2wiIHhtbG5zPSJ1cm46b2FzaXM6bmFtZXM6dGM6U0F
NTDoyLjA6YXNzZXJ0aW9uIg0KICAgIElEPSJiMDczMGQyMWI2MjgxMTBkOGI3ZTAwNDAwNW
IxM2EyYiIgSW5SZXNwb25zZVRvPSJkMmI3YzM4OGNlYzM2ZmE3YzM5YzI4ZmQyOTg2NDRhO
CINCiAgICBJc3N1ZUluc3RhbnQ9IjIwMDQtMDEtMjFUMTk6MDA6NDlaIiBWZXJzaW9uPSIy
LjAiPg0KICAgIDxJc3N1ZXI+aHR0cHM6Ly9TZXJ2aWNlUHJvdmlkZXIuY29tL1NBTUw8L0l
zc3Vlcj4NCiAgICA8c2FtbHA6U3RhdHVzPg0KICAgICAgICA8c2FtbHA6U3RhdHVzQ29kZS
BWYWx1ZT0idXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6Mi4wOnN0YXR1czpTdWNjZXNzIi8+D
QogICAgPC9zYW1scDpTdGF0dXM+DQo8L3NhbWxwOkxvZ291dFJlc3BvbnNlPg=="/>
</div>
<noscript>
<div>
<input type="submit" value="Continue"/>
</div>
</noscript>
</form>
</body>
</html>
```

This results in a browser request similar to the following:

```
POST /SAML/SLO/Response HTTP/1.1
Host: IdentityProvider.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:70.0)
Gecko/20100101 Firefox/70.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://IdentityProvider.com
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 691

RelayState=0043bfc1bc45110dae17004005b13a2b&SAMLResponse=PHNhbWxwOkxvZ2
91dFJlc3BvbnNlIHhtbG5zOnNhbWxwPSJ1cm46b2FzaXM6bmFtZXM6dGM6U0FNTDoyLjA6c
HJvdG9jb2wiIHhtbG5zPSJ1cm46b2FzaXM6bmFtZXM6dGM6U0FNTDoyLjA6YXNzZXJ0aW9u
Ig0KICAgIElEPSJiMDczMGQyMWI2MjgxMTBkOGI3ZTAwNDAwNWIxM2EyYiIgSW5SZXNwb25
zZVRvPSJkMmI3YzM4OGNlYzM2ZmE3YzM5YzI4ZmQyOTg2NDRhOCINCiAgICBJc3N1ZUluc3
```

```
RhbnQ9IjIwMDQtMDEtMjFUMTk6MDA6NDlaIiBWZXJzaW9uPSIyLjAiPg0KICAgIDxJc3N1Z
XI%2BaHR0cHM6Ly9TZXJ2aWNlUHJvdmlkZXIuY29tL1NBTUw8L0lzc3Vlcj4NCiAgICA8c2
FtbHA6U3RhdHVzPg0KICAgICAgICA8c2FtbHA6U3RhdHVzQ29kZSBWYWx1ZT0idXJuOm9hc
2lzOm5hbWVzOnRjOlNBTUw6Mi4wOnN0YXR1czpTdWNjZXNzIi8%2BDQogICAgPC9zYW1scD
pTdGF0dXM%2BDQo8L3NhbWxwOkxvZ291dFJlc3BvbnNlPg%3D%3D
```


A.4.3 HTTP Artifact Binding

The HTTP Artifact binding defines two methods for sending a SAML artifact in place of a SAML message to a recipient. The two methods are similar to the HTTP Redirect and HTTP POST bindings. The sender can URL-encode the artifact and include it in a URL query string parameter named SAMLart in an HTTP redirect, or it can return an HTML form with a hidden SAMLart field containing the artifact. RelayState data can be sent with the artifact in the same way as in the HTTP Redirect and POST bindings.

The HTTP Artifact binding also defines the format of artifacts. Artifacts must begin with a two-byte TypeCode and two-byte EndpointIndex. The TypeCode references an artifact type definition explaining how to interpret the remaining data; the EndpointIndex references a specific endpoint of the sender's artifact resolution service to which the artifact can be sent to obtain the referenced SAML message. This would typically reference an endpoint specified in the sender's SAML metadata. Arbitrary data can follow these four bytes, and the artifact is composed of the base64-encoded concatenation of the TypeCode, EndpointIndex, and the remaining data.

The binding also defines a specific artifact type with code 0x0004, where the data following the TypeCode and EndpointIndex consists of a 20-byte SourceID and 20-byte MessageHandle. The SourceID is a hash of the sender's SAML Entity ID (typically a URL that uniquely identifies each participant in a SAML environment), and the message handle is a pseudorandom value. The recipient can use the SourceID to identify the issuer of the artifact. Other artifact types can be defined, although no others are known to be in wide use. TypeCodes 1-3 are legacy codes associated with SAML 1.0 and 1.1.

The HTTP Artifact binding represents one half of a complete SAML message exchange using artifacts; the other component is the artifact resolution protocol discussed in Appendix A.3.3. Figure 8 shows a complete message exchange with a SAML request sent using the artifact URL encoding and the response sent using the artifact form encoding.

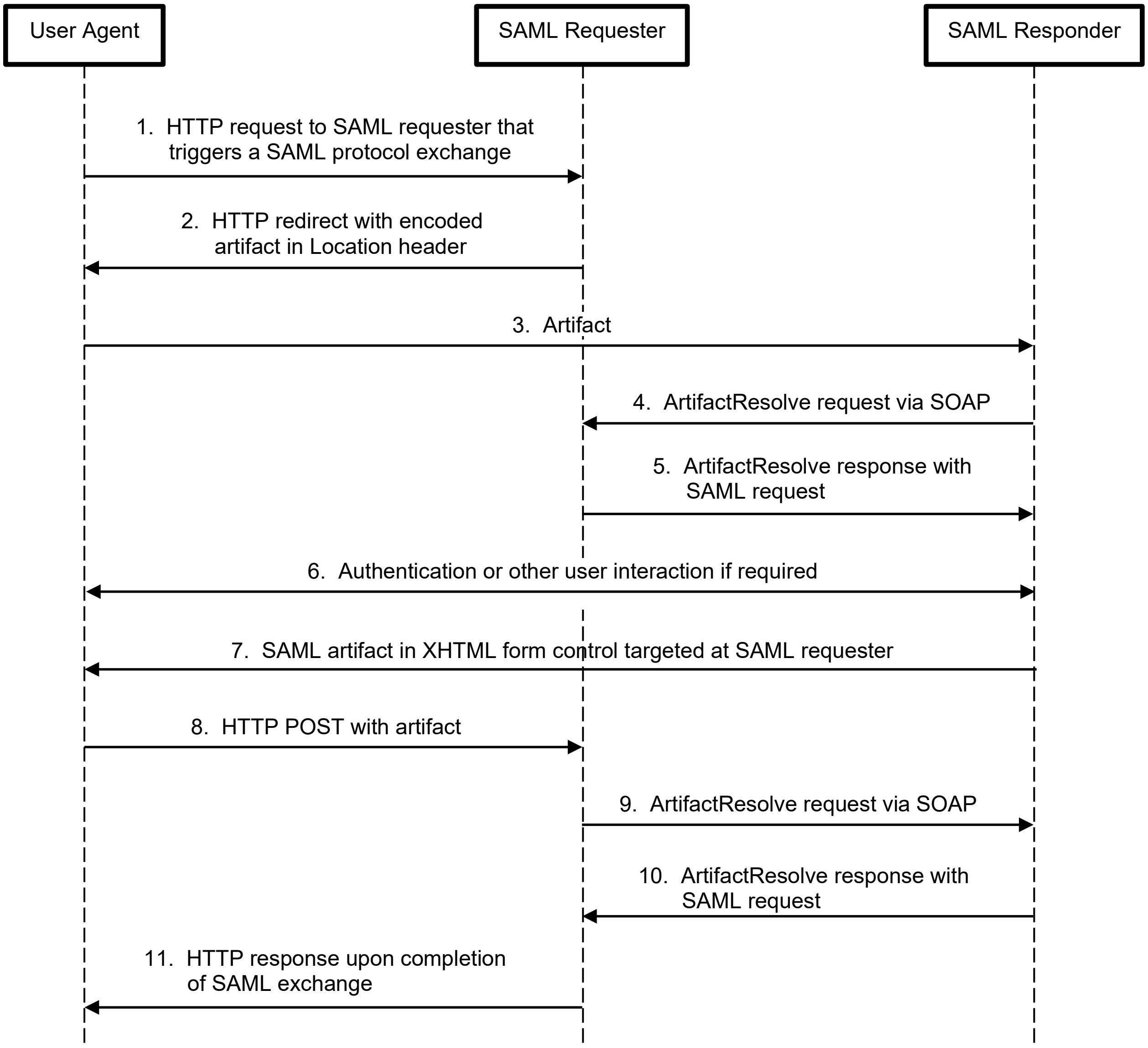

Figure 8. SAML HTTP Artifact Message Exchange

Whereas the HTTP Redirect and HTTP POST bindings use only asynchronous bindings, an artifact message exchange requires both asynchronous and synchronous bindings (since the artifact resolution protocol has no asynchronous bindings). One consequence of this is that there must be direct connectivity between the SAML requester and SAML responder.

A.4.4 SOAP Binding

SOAP is an XML-based, extensible messaging framework that defines an XML message envelope with separate sections for message headers carrying control information and a message body containing actual data. Like SAML, SOAP is transport protocol-agnostic, but it is typically sent over HTTP.

SAML interactions over SOAP use a simple request-response model. The SAML requester sends a SAML request element as the sole contents of the SOAP body. The body may not contain more than one SAML request or any other XML elements outside of the SAML request. Similarly, the responder sends a SOAP message in reply that contains only a single SAML response in the SOAP body. Error handling depends on where the error occurs. If the responder encounters a SOAP error or a general error that prevents SAML processing, it must return a SOAP fault. If an error occurs within the processing of the SAML request—for example, if the user fails to authenticate or there is a problem with fulfilling the specific SAML request—the responder must return HTTP status 200 (“OK”) and include a SAML response in the SOAP body containing a Status element that reflects the SAML error condition. This maintains a clear separation between the SOAP transport processing and the SAML message processing.

The following example from the SAML Bindings specification [14] shows a SAML request sent via SOAP over HTTP.

```
POST /SamlService HTTP/1.1
Host: www.example.com
Content-Type: text/xml
Content-Length: nnn
SOAPAction: http://www.oasis-open.org/committees/security

<SOAP-ENV:Envelope
    xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/”>
    <SOAP-ENV:Body>
      <samlp:AttributeQuery xmlns:samlp:=”...”
          xmlns:saml=”...” xmlns:ds=”...” ID=”_6c3a4f8b9c2d”
          Version=”2.0” IssueInstant=”2004-03-27T08:41:00Z”>
          <ds:Signature> ... </ds:Signature>
          <saml:Subject>
          ...
          </saml:Subject>
      </samlp:AttributeQuery>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The corresponding response would be similar, beginning with HTTP response headers and then containing a SOAP message with the SAML response in the body.

A.4.5 Reverse SOAP (PAOS) Binding

The SAML PAOS binding was created to support the ECP profile. PAOS is used between the client and a SAML requester (typically a service provider in the ECP profile) and enables the client to act as the intermediary in a SAML message exchange over SOAP between the SAML requester and a SAML responder.

In a PAOS message exchange, the client sends a request to the SAML requester that includes HTTP headers indicating that the client can support the PAOS binding. The SAML requester returns an HTTP response with a SOAP envelope containing a SAML request in the message body. Typically, the client then submits the SAML request to a SAML responder using the SOAP binding and receives a SAML response in a SOAP envelope; this interaction does not depend on the PAOS binding. The client then uses the PAOS binding to submit the response back to the SAML requester by including the SOAP envelope in the body of an HTTP request.

A.5 Profiles

A.5.1 Web Browser SSO Profile

Two message flows are defined in the web browser SSO profile. In SP-initiated web SSO, the flow begins with the user's browser attempting to access a resource at the SP that requires authentication via SAML, and the SP redirecting the user to the IdP. In IdP-initiated web SSO, the flow begins with the user interacting with the IdP and being redirected to the SP.

One instance of the SP-initiated web browser SSO message flow is illustrated in Figure 9. In this example, the AuthnRequest is delivered using the HTTP Redirect binding and the response is delivered using the POST binding. Any combination of the Redirect, POST, and Artifact bindings can be used to transmit the request and the response.

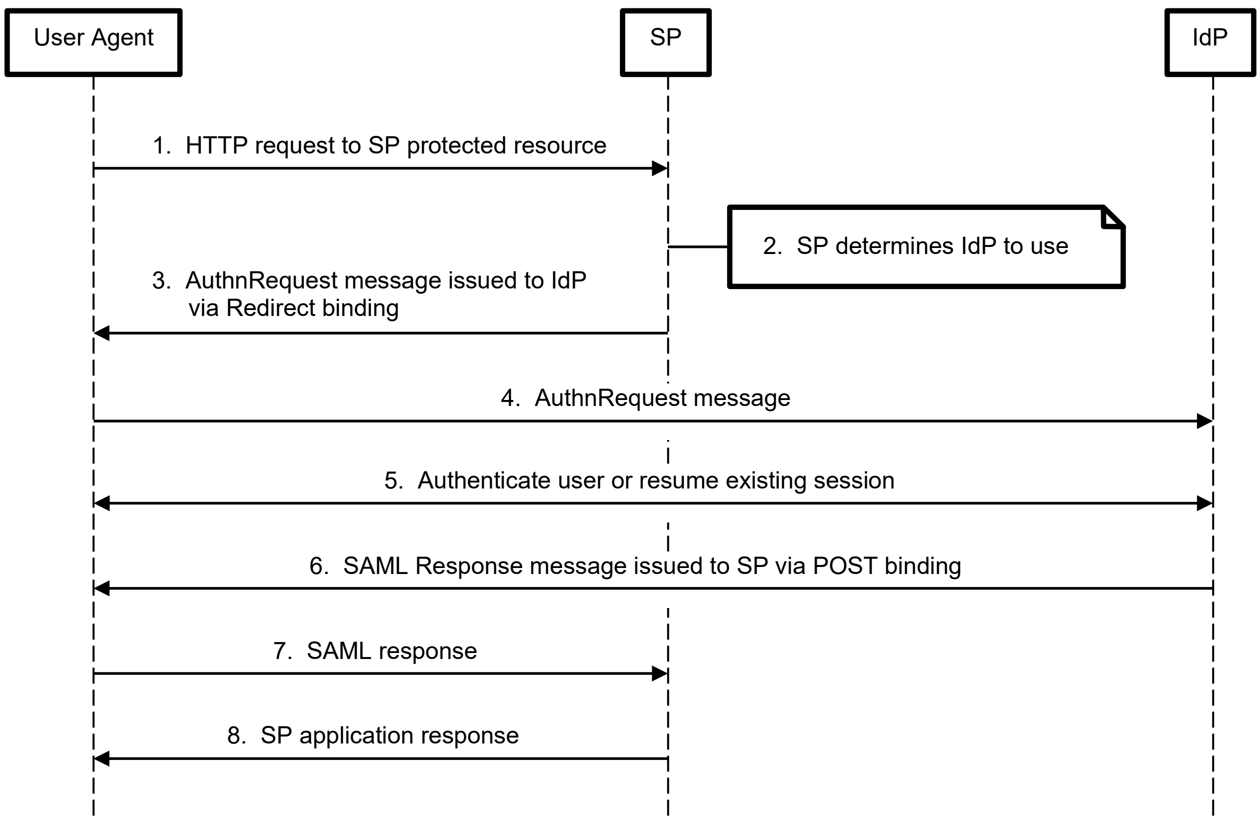

Figure 9. Web Browser SSO SP-Initiated Message Flow with Redirect and POST Bindings

The detailed steps are as follows:

1. The browser submits a request to the SP that requires authentication.
2. The SP performs IdP discovery to identify the IdP to which the user should be redirected.
3. The SP submits a SAML AuthnRequest to the appropriate IdP through the browser using the Redirect binding by sending an HTTP redirect message with the AuthnRequest encoded into the URL passed in the Location header.
4. The browser submits the AuthnRequest to the IdP through an HTTP GET in response to the redirect.

5. The IdP receives the AuthnRequest and performs any required validation such as checking the request signature if applicable. If the user does not have an existing session or if the user's session does not meet the requirements of the AuthnRequest (e.g., if the user was authenticated at a lower AAL than the SP has requested or if explicit authentication is requested using the ForceAuthn attribute), the IdP authenticates the user.
6. The IdP creates a SAML response including an AuthnStatement, subject identifier, authentication context information, and other elements as specified in Section 4.1. If the IdP supports the Single Logout profile, the AuthnStatement must include a SessionIndex attribute (see Appendix A.5.2). The IdP responds to the browser with an XHTML document including a form carrying the encoded SAML response as per the HTTP POST binding.
7. The browser submits the form data including the response to the RP's assertion consumer service in an HTTP POST message. This typically occurs without user interaction through JavaScript included in the XHTML page.
8. The RP validates the SAML response, extracts the subject identifier and any other required attributes, and establishes an application session for the user. The RP's response to the browser is undefined by the SAML specifications and is typically application-specific content.

Figure 10 shows another variation on the RP-initiated flow where the AuthnRequest is sent using the POST binding and the response is sent using the Artifact binding. The artifact is sent through the front channel in place of the response, and the RP makes an additional back-channel ArtifactResolve request to obtain the response.

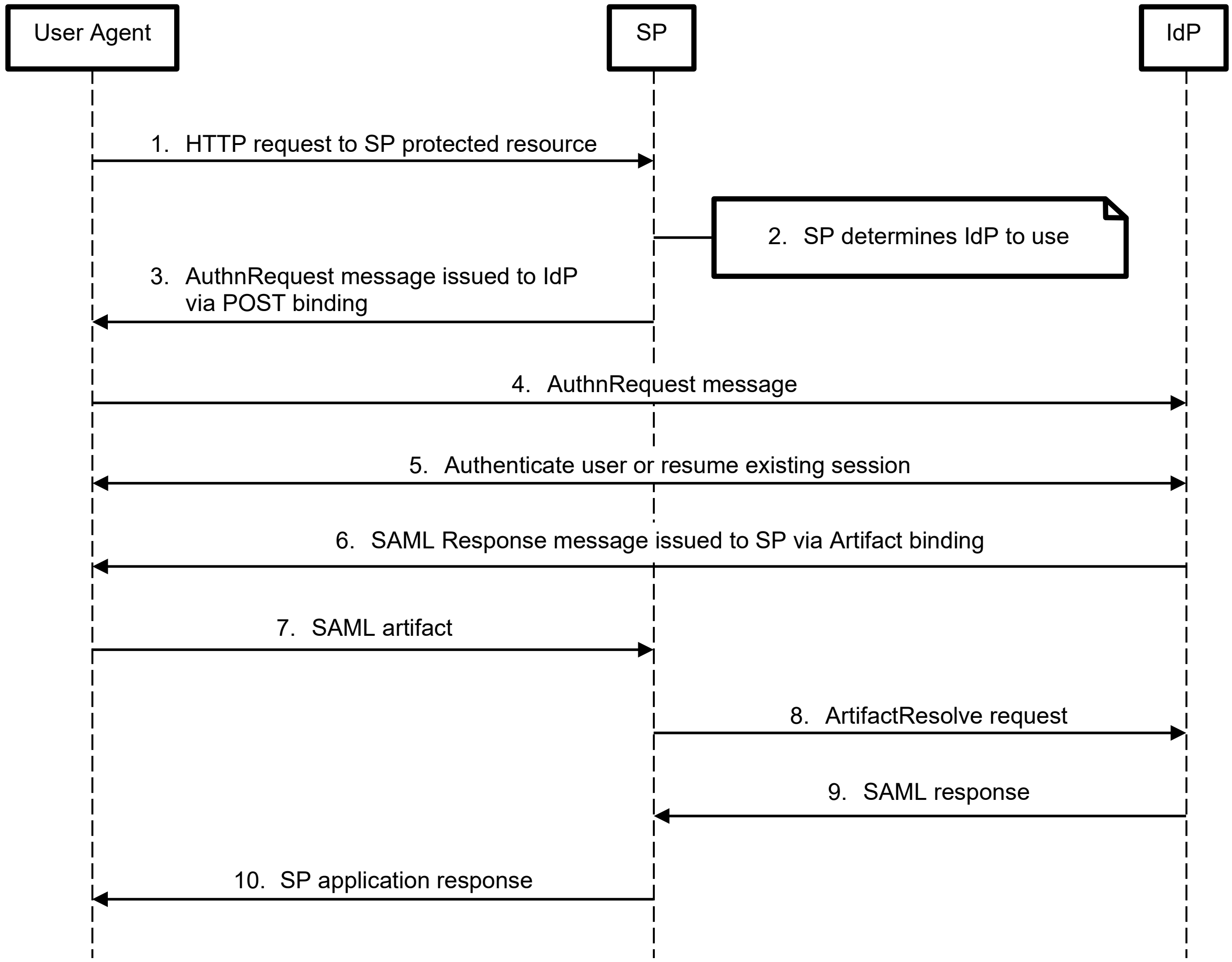

Figure 10. Web Browser SSO SP-Initiated Message Flow with POST and Artifact Bindings

Figure 11 shows the IdP-initiated web browser SSO message flow. In this flow, the user interacts with the IdP before submitting any request to the RP. A typical use case for the IdP-initiated flow is a portal that users log into in order to access multiple SP applications. The user submits a request to the IdP to interact with the SP. The IdP creates a SAML response addressed to the SP's assertion consumer service and submits it through the POST binding via the browser (the Redirect and Artifact bindings can also be used). The response is unsolicited since the SP has not sent an AuthnRequest; the response does not have an InResponseTo attribute, which would typically contain the ID of the corresponding request.

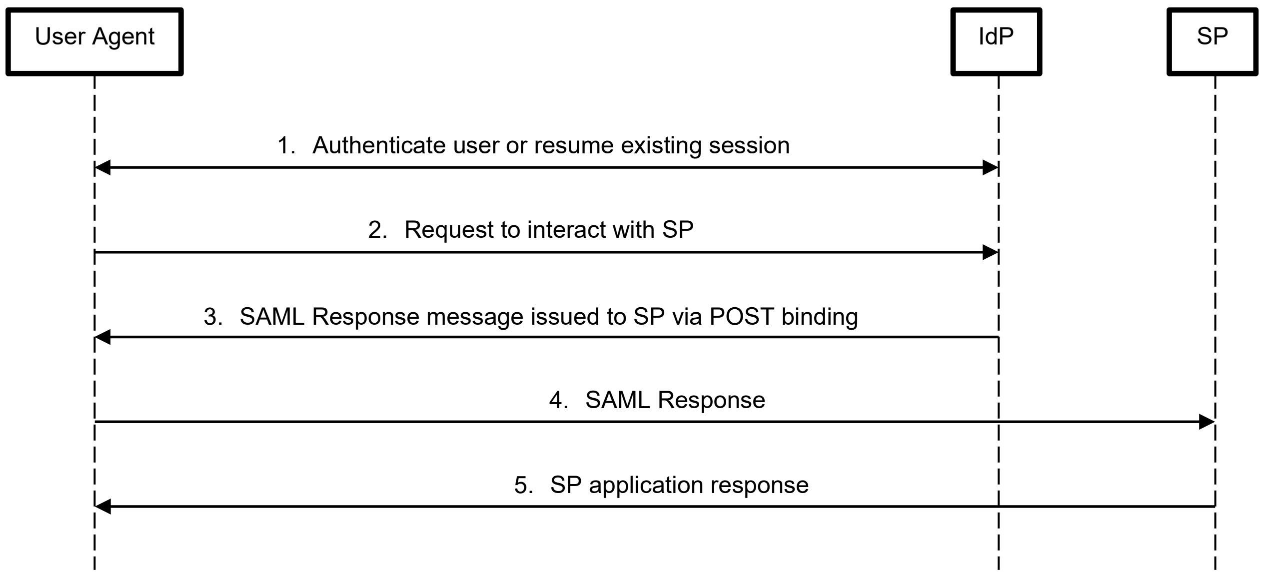

Figure 11. Web Browser SSO IdP-Initiated Message Flow with POST Binding

A.5.2 Enhanced Client or Proxy (ECP) Profile

In the ECP profile message flow, a client attempts to access an SP resource over HTTP but does not have an active session. The SP sends an HTTP response whose body includes a SOAP envelope that in turn contains a SAML request in the SOAP body. The client then submits the SAML request to the IdP using the SOAP binding and receives a SOAP response containing the SAML response. The client then submits an HTTP request containing the SOAP response in the message body back to the RP, which processes the SAML response and returns an HTTP response. The contents of the final response are not specified by the PAOS binding, but they would typically be the RP application's response to the original HTTP request or an HTTP error if the SAML response was not accepted.

Essentially, the ECP acts as an intermediary to pass SOAP messages between the RP and the IdP. It is assumed that the ECP is pre-configured to use a specific IdP. The ECP profile is not widely used or supported in existing software, so it is not discussed at length here.

A.5.3 Single Logout Profile

The single logout profile supports sending LogoutRequest and LogoutResponse messages over the SOAP, HTTP Redirect, POST, or Artifact bindings. The single logout message flow is shown in Figure 12. The detailed steps are as follows:

1. A session participant initiates the Single Logout flow by sending a LogoutRequest.
2. When an IdP receives a LogoutRequest or initiates Single Logout itself, it terminates the affected user session and identifies any additional session participants that should be notified. LogoutRequests sent by session participants must include a SessionIndex parameter. This value is originally sent by the IdP to the SP in its response to the AuthnRequest, and it can be used by the IdP to identify additional session participants that should be involved in the Single Logout flow.

3. The IdP attempts to send LogoutRequests to all session participants involved in the current session using any combination of bindings supported by the participants.
4. Individual session participants process the LogoutRequest by terminating the user's session.
5. Individual session participants send a LogoutResponse to the IdP indicating their success or failure in processing the request.
6. If the request was initiated by a session participant, once the IdP has either received responses from all session participants or encountered errors in contacting them, it sends a LogoutResponse to the participant that initiated the request. The IdP's LogoutResponse messages indicates success or failure in terminating the user's session at the IdP. If not all session participants returned successful LogoutResponses, the IdP's LogoutResponse can include a second-level status code indicating that a partial logout has occurred.

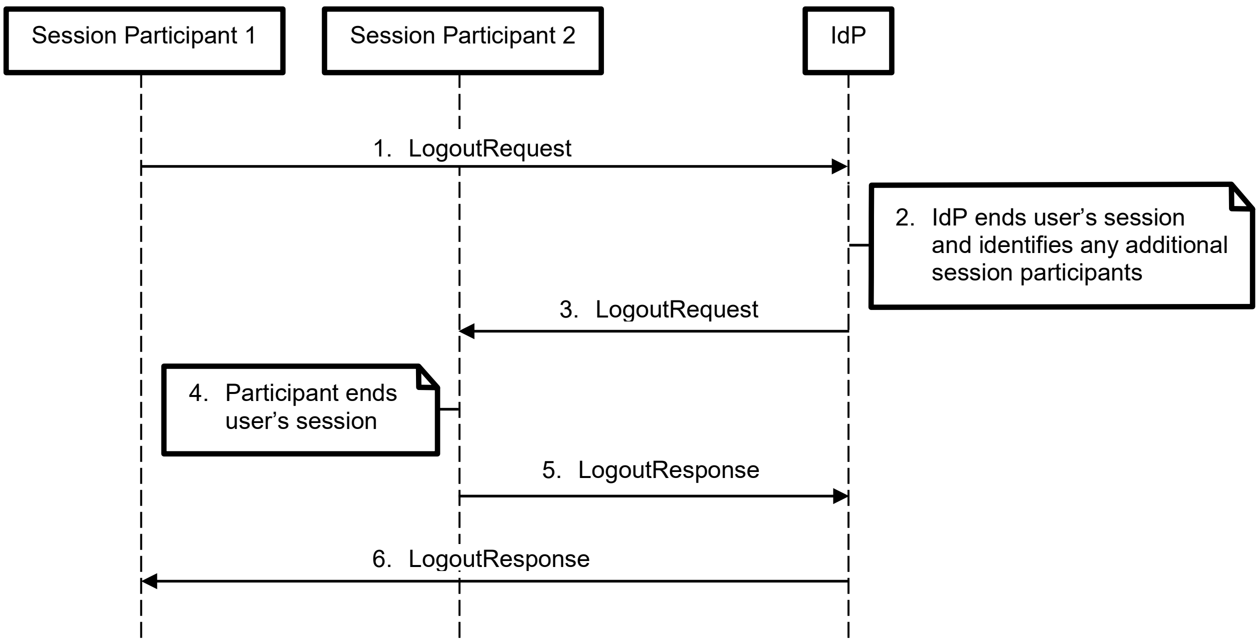

Figure 12. Single Logout Profile Message Flow

Although Figure 12 shows a Single Logout flow initiated by a session participant, Single Logout may also be initiated by the IdP, in which case the above flow would begin at Step 2, and Step 6 would not occur.

The single logout profile supports both front-channel and back-channel bindings, but it recommends using a front-channel binding when sending a LogoutRequest from a session participant to an IdP to maximize the likelihood of the IdP being able to contact all session participants. The rationale for this guidance is that some session participants may only support front-channel bindings and if the initial LogoutRequest is submitted via the back-channel SOAP binding, the IdP has no interaction with the user's browser and the front channel cannot be used to send LogoutRequests to additional session participants. However, the front channel also has the drawback that it requires the user to wait for a series of redirects to complete as each session participant is contacted sequentially. If the browser appears to become unresponsive while the user waits for a logout to complete, many users may browse to a different page or close the browser, interrupting the single logout process.

Appendix B—Sample SAML Metadata Document

This example from the SAML metadata specification shows the metadata document for a system that performs the IdP and attribute authority roles. The Signature element value shown here is a placeholder for an actual XML signature value.

```
<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   entityID="https://IdentityProvider.com/SAML">
   <ds:Signature>...</ds:Signature>
   <IDPSSODescriptor WantAuthnRequestsSigned="true"
      protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
      <KeyDescriptor use="signing">
         <ds:KeyInfo>
            <ds:KeyName>IdentityProvider.com SSO Key</ds:KeyName>
         </ds:KeyInfo>
      </KeyDescriptor>
      <ArtifactResolutionService isDefault="true" index="0"
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
        Location="https://IdentityProvider.com/SAML/Artifact"/>
      <SingleLogoutService
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
        Location="https://IdentityProvider.com/SAML/SLO/SOAP"/>
      <SingleLogoutService
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
        Location="https://IdentityProvider.com/SAML/SLO/Browser"
        ResponseLocation="https://IdentityProvider.com/SAML/SLO/Response"/>
      <NameIDFormat>
        urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
      </NameIDFormat>
      <NameIDFormat>
        urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
      </NameIDFormat>
      <NameIDFormat>
        urn:oasis:names:tc:SAML:2.0:nameid-format:transient
      </NameIDFormat>
      <SingleSignOnService
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
        Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
      <SingleSignOnService
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
        Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
      <saml:Attribute
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
        Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
        FriendlyName="eduPersonPrincipalName">
      </saml:Attribute>
      <saml:Attribute
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
        Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
        FriendlyName="eduPersonAffiliation">
        <saml:AttributeValue>member</saml:AttributeValue>
        <saml:AttributeValue>student</saml:AttributeValue>
        <saml:AttributeValue>faculty</saml:AttributeValue>
        <saml:AttributeValue>employee</saml:AttributeValue>
        <saml:AttributeValue>staff</saml:AttributeValue>
```

```
      </saml:Attribute>
   </IDPSSODescriptor>
   <AttributeAuthorityDescriptor
      protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
      <KeyDescriptor use="signing">
        <ds:KeyInfo>
            <ds:KeyName>IdentityProvider.com AA Key</ds:KeyName>
        </ds:KeyInfo>
      </KeyDescriptor>
      <AttributeService
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
        Location="https://IdentityProvider.com/SAML/AA/SOAP"/>
      <AssertionIDRequestService
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
        Location="https://IdentityProvider.com/SAML/AA/URI"/>
      <NameIDFormat>
        urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
      </NameIDFormat>
      <NameIDFormat>
        urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
      </NameIDFormat>
      <NameIDFormat>
        urn:oasis:names:tc:SAML:2.0:nameid-format:transient
      </NameIDFormat>
      <saml:Attribute
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
        Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
        FriendlyName="eduPersonPrincipalName">
      </saml:Attribute>
      <saml:Attribute
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
        Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
        FriendlyName="eduPersonAffiliation">
        <saml:AttributeValue>member</saml:AttributeValue>
        <saml:AttributeValue>student</saml:AttributeValue>
        <saml:AttributeValue>faculty</saml:AttributeValue>
        <saml:AttributeValue>employee</saml:AttributeValue>
        <saml:AttributeValue>staff</saml:AttributeValue>
      </saml:Attribute>
   </AttributeAuthorityDescriptor>
   <Organization>
      <OrganizationName xml:lang="en">
        Identity Providers R US
      </OrganizationName>
      <OrganizationDisplayName xml:lang="en">
        Identity Providers R US, a Division of Lerxst Corp.
      </OrganizationDisplayName>
      <OrganizationURL xml:lang="en">
        https://IdentityProvider.com
      </OrganizationURL>
   </Organization>
</EntityDescriptor>
```

Appendix C—Additional Information on OpenID Connect Implementation

This appendix provides additional information on OpenID Connect implementation that supplements the contents of Section 5. This information is intended for readers who are already familiar with JSON syntax and conventions and who need more detailed information than what Section 5 provides.

C.1 Specifications

Final and draft OpenID Connect specifications are published on the OpenID Foundation's website [22]. Table 17 lists the OpenID Connect specifications maintained by the core OpenID Connect working group.

Table 17. OpenID Connect Core Working Group Specifications

Document Title	Status	URL
OpenID Connect Core 1.0	Final	http://openid.net/specs/openid-connect-core-1_0.html
OpenID Connect Discovery 1.0	Final	http://openid.net/specs/openid-connect-discovery-1_0.html
OpenID Connect Dynamic Client Registration 1.0	Final	http://openid.net/specs/openid-connect-registration-1_0.html
OAuth 2.0 Multiple Response Type Encoding Practices	Final	http://openid.net/specs/oauth-v2-multiple-response-types-1_0.html
OAuth 2.0 Form Post Response Mode	Final	http://openid.net/specs/oauth-v2-form-post-response-mode-1_0.html
OpenID 2.0 to OpenID Connect Migration 1.0	Final	http://openid.net/specs/openid-connect-migration-1_0.html
OpenID Connect Session Management 1.0	Implementer's Draft	http://openid.net/specs/openid-connect-session-1_0.html
OpenID Connect Front-Channel Logout 1.0	Implementer's Draft	http://openid.net/specs/openid-connect-frontchannel-1_0.html
OpenID Connect Back-Channel Logout 1.0	Implementer's Draft	http://openid.net/specs/openid-connect-backchannel-1_0.html
OpenID Connect Federation 1.0	Implementer's Draft	http://openid.net/specs/openid-connect-federation-1_0.html

C.2 Assertions

To create an ID token, the IdP encodes a set of claims in a JSON object. OpenID Connect Core provides the following example of a JSON object containing claims to be included in an ID token:

```
{
 "iss": "http://server.example.com",
 "sub": "248289761001",
 "aud": "s6BhdRkqt3",
 "nonce": "n-0S6_WzA2Mj",
 "exp": 1311281970,
 "iat": 1311280970,
 "name": "Jane Doe",
```

```
    "given_name": "Jane",
    "family_name": "Doe",
    "gender": "female",
    "birthdate": "0000-10-31",
    "email": "janedoe@example.com",
    "picture": "http://example.com/janedoe/me.jpg"
   }
```


A JWT is then created using the JSON object as the payload. A signed JWT is structured in three sections—a header, the payload or content of the JWT, and the signature. The JWT header is itself a JSON object that references the key and algorithm used to sign the JWT, as in the following example:

```
{"kid":"1e9gdk7","alg":"RS256"}
```

The “kid” claim contains a reference to the signing key. The key itself would either be shared out-of-band or made available through another means such as a JSON Web Key Set (JWKS) URL. The “alg” claim identifies the signing algorithm, which in this case is the Rivest, Shamir, and Adelman (RSA)-256 algorithm.

To construct the JWT, the IdP encodes the header and payload with the BASE64URL encoding. The resulting encoded strings are concatenated together, separated by a period, and the signature is calculated over the resulting string. The final form of the JWT is the concatenation of the header, payload, and signature, all BASE64URL encoded, separated by periods. The JWT created using the previous examples is shown below. To make the different sections easier to identify, the header and signature are shown in red text.


```
eyJraWQiOiIxZTlnZGs3IiwiYWxnIjoiUlMyNTYifQ.ewogImlzcyI6ICJodHRwOi8vc2Vy
dmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5NzYxMDAxIiwKICJhdWQiOiAiczZ
CaGRSa3F0MyIsCiAibm9uY2UiOiAibi0wUzZfV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOT
cwLAogImlhdCI6IDEzMTEyODA5NzAsCiAibmFtZSI6ICJKYW5lIERvZSIsCiAiZ2l2ZW5fb
mFtZSI6ICJKYW5lIiwKICJmYW1pbHlfbmFtZSI6ICJEb2UiLAogImdlbmRlciI6ICJmZW1h
bGUiLAogImJpcnRoZGF0ZSI6ICIwMDAwLTEwLTMxIiwKICJlbWFpbCI6ICJqYW5lZG9lQGV
4YW1wbGUuY29tIiwKICJwaWN0dXJlIjogImh0dHA6Ly9leGFtcGxlLmNvbS9qYW5lZG9lL2
1lLmpwZyIKfQ.rHQjEmBqn9Jre0OLykYNnspA10Qql2rvx4FsD00jwlB0Sym4NzpgvPKsDj
n_wMkHxcp6CilPcoKrWHcipR2iAjzLvDNAReF97zoJqq880ZD1bwY82JDauCXELVR9O6_B0
w3K-E7yM2macAAgNCUwtik6SjoSUZRcf-O5lygIyLENx882p6MtmwaL1hd6qn5RZOQ0TLrO
Yu0532g9Exxcm-ChymrB4xLykpDj3lUivJt63eEGGN6DH5K6o33TcxkIjNrCD4XB1CKKumZ
vCedgHHF3IAK4dVEDSUoGlH9z4pP_eWYNXvqQOjGs-rDaQzUHl6cQQWNiDpWOl_lxXjQEvQ
```


C.3 Protocols

C.3.1 Authorization Code Flow

The OpenID Connect authorization code flow is shown in Figure 13. Though the messaging and transport protocols used are different, the flow is similar to the SAML web browser SSO flow when the artifact binding is used to deliver the response, as shown in Figure 10.

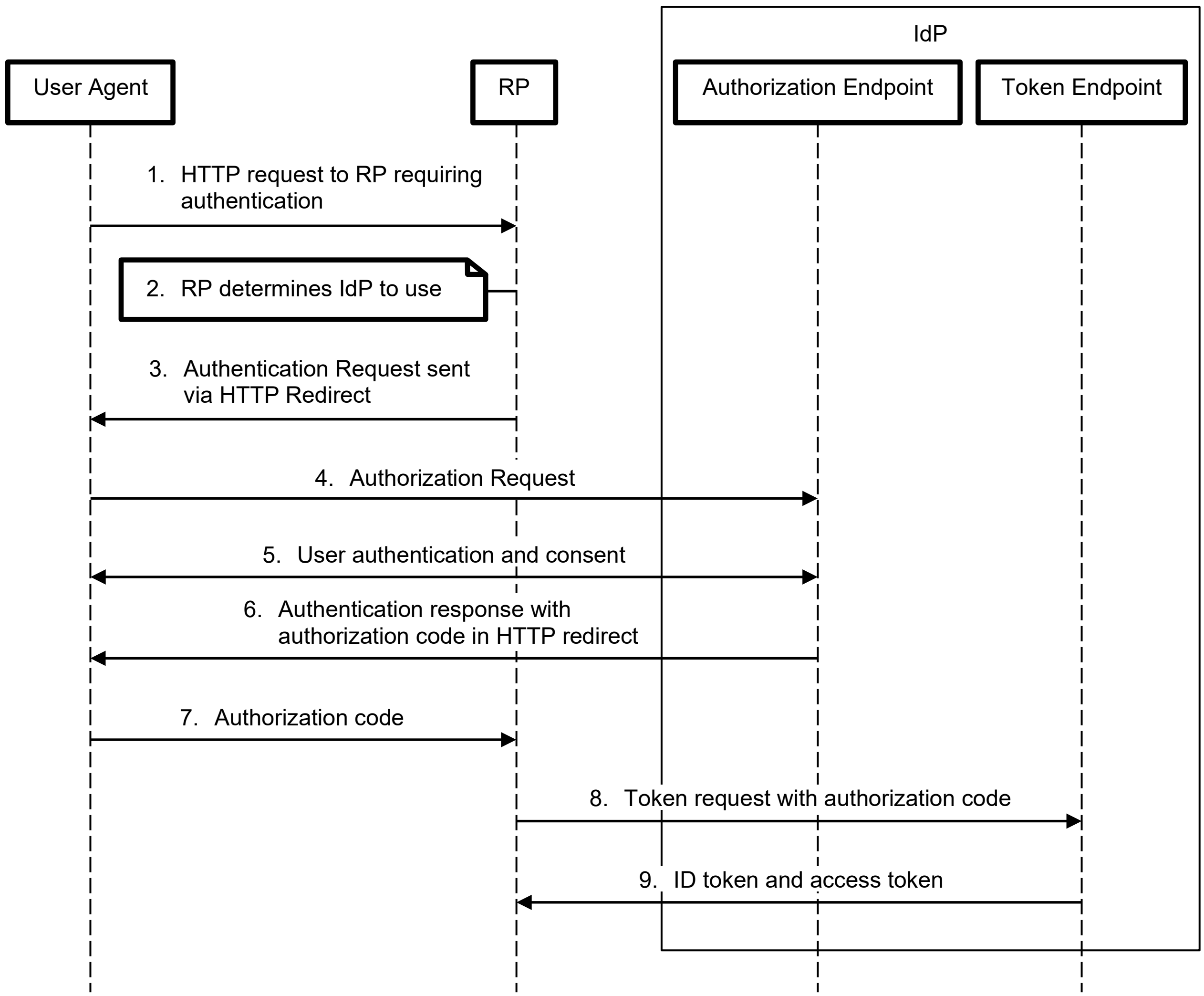

Figure 13. OpenID Connect Authorization Code Flow

The steps are as follows:

1. The browser sends an HTTP request to the RP that requires authentication.
2. The RP determines which IdP to use to authenticate the user through IdP discovery, as discussed in Section 2.5.
3. The RP sends an HTTP redirect response to the browser with a URL that points to the IdP's authorization endpoint and contains an encoded authentication request. The request's response_type parameter is set to "code," which triggers the authorization code flow.
4. The browser submits the authentication request to the authorization endpoint.
5. If necessary, the IdP prompts the user for authentication and for consent to authenticate and provide any requested identifiers and attributes to the RP.
6. The IdP sends an HTTP redirect to the browser with a URL that points to the RP's redirect_uri and contains an authorization code. The authorization code is a short-lived opaque value that references the authentication transaction, similar to a SAML artifact.

7. The browser submits the authorization code to the RP's redirect_uri.
8. The RP submits a token request to the IdP's token endpoint with the authorization code as a parameter. If the client is a confidential client, it authenticates itself to the IdP as part of this request.
9. The IdP returns the ID token to the RP along with an access token and optionally a refresh token to enable access to the userinfo endpoint (see Appendix C.3.4).

The RP can validate the ID token by checking the signature and validity period, checking the "aud" (audience) claim to ensure the token was sent to the intended RP, validating "nonce" and "state" values, etc. If the token is valid, the RP can extract the "sub" value and other attributes and initiate or create a local session for the authenticated user.

C.3.2 Implicit Flow

The implicit flow is intended for use by public client RPs. The flow is shown in Figure 14.

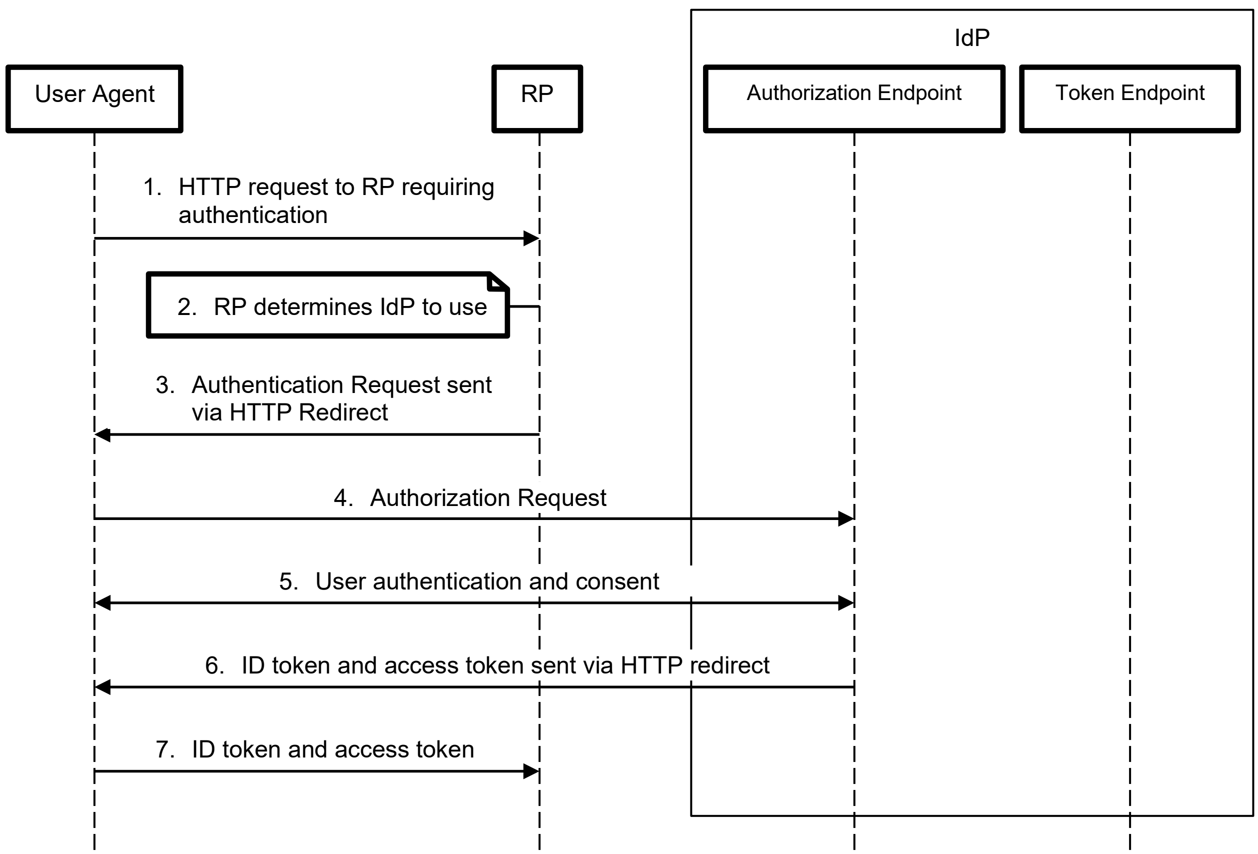

Figure 14. OpenID Connect Implicit Flow

The implicit flow begins the same way as the authorization code flow. In the authentication request sent in step 3, the response_type is either "id_token" or "id_token token"—either of these values will trigger the implicit flow. If "token" is included in the response_type parameter, an access token for use at the userinfo endpoint will be returned in addition to the ID token.

In step 6, the IdP returns the ID token (and access token, if requested) directly to the RP instead of using an authorization code. The IdP's token endpoint is not used in the implicit flow, and the RP does not authenticate itself to the IdP (which public clients cannot do in any case).

The implicit flow is the only OpenID Connect flow where the ID token is transmitted through the front channel, increasing the likelihood of interception of the ID and access tokens by an unauthorized party.

> **Caution:** Best practice guidance has shifted to discourage the implicit flow in both OAuth and OpenID Connect for public clients in favor of using the authorization code flow. Although public clients cannot authenticate themselves to the IdP's token endpoint, they can use other security measures like Proof Key for Code Exchange (PKCE) [23]. PKCE does not authenticate the client, but it does provide assurance that the authorization code can only be redeemed by the same client that initiated the authentication request.
>
> *Token Binding*, another proposed standard to protect OAuth and OpenID Connect protocol flows against man-in-the-middle and token export or replay attacks, has not gained industry adoption and is unlikely to be supported in commonly used web browsers or client software.

C.3.3 Hybrid Flow

There are three variations on the hybrid flow; in each case the IdP returns one or more tokens in both the front and back channels. Three different values can be used for the response_type parameter in the authentication request to trigger the different versions of the hybrid flow and dictate what objects are returned in the front channel:

- code id_token
- code token
- code id_token token

The message sequence of the hybrid flow is similar to the authorization code flow shown in Figure 13, except that in step 6 the IdP's authorization endpoint would return an ID token and/or an access token in addition to the authorization code.

The hybrid flow can effectively enable the issuance of tokens separately to the front end and back end of an application. Consider a web application built using a reactive framework where the front end running in the user's browser interacts with a back-end API but also has independent client-side functionality. Using the "code token" response_type parameter, the front end would obtain an access token and the back end could use the authorization code to obtain its own separate access token. The two tokens could have different scopes of access associated with them, authorizing the front end to make a limited set of API calls. The access token issued to the back end would be delivered through the back channel and not exposed to the front end and could have a wider scope of authorizations. This scenario would typically occur in a situation where the IdP is also acting as an OAuth authorization server providing access to other APIs.

C.3.4 Userinfo Endpoint

OpenID providers host an additional endpoint called userinfo that provides a REST interface to obtain claims about the user. RPs must present a valid access token issued by the IdP through one of the authentication flows described above to authorize userinfo requests. The userinfo response is sent within the body of the IdP's HTTP response and may consist of a JSON object (equivalent to the JSON payload of an ID token) or a JWT that is signed and/or encrypted. OpenID Connect Core provides the following sample userinfo response in JSON format:


```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "sub": "248289761001",
  "name": "Jane Doe",
  "given_name": "Jane",
  "family_name": "Doe",
  "preferred_username": "j.doe",
  "email": "janedoe@example.com",
  "picture": "http://example.com/janedoe/me.jpg"
}
```


The userinfo endpoint is functionally similar to the SAML attribute query protocol. OpenID Connect does not dictate that the claims returned from userinfo be the same set of claims in the ID token. Some implementations include a minimal number of claims in the ID token and provide more information via userinfo. Clients can use the optional claims request parameter to request that certain claims be made available in the ID token or from the userinfo endpoint. Userinfo might also be used to verify that a claim previously received in an ID token is still valid and has not changed.

Appendix D—Acronyms and Abbreviations

AAL	Authenticator Assurance Level
AES	Advanced Encryption Standard
AP	Attribute Provider
API	Application Programming Interface
BYOD	Bring Your Own Device
CAD	Computer Aided Dispatch
CJIS	Criminal Justice Information Services
CSP	Credential Service Provider
CSRF	Cross-Site Request Forgery
ECP	Enhanced Client or Proxy
EMS	Emergency Medical Services
FAL	Federation Assurance Level
FAPI	Financial-Grade Application Programming Interface
FICAM	Federal Identity, Credential, and Access Management
FOIA	Freedom of Information Act
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IaaS	Infrastructure as a Service
IAL	Identity Assurance Level
ICAM	Identity, Credential, and Access Management
IDaaS	Identity as a Service
IdP	Identity Provider
IETF	Internet Engineering Task Force
IP	Internet Protocol
IR	Interagency or Internal Report
IT	Information Technology
ITL	Information Technology Laboratory
JSON	JavaScript Object Notation
JWE	JSON Web Encryption
JWKS	JSON Web Key Set
JWS	JSON Web Signature
JWT	JSON Web Token

MAC	Message Authentication Code
MFA	Multi-Factor Authentication
NCCoE	National Cybersecurity Center of Excellence
NIEF	National Identity Exchange Federation
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OASIS	Organization for the Advancement of Structured Information Standards
OMB	Office of Management and Budget
OP	OpenID Provider
PD	Police Department
PII	Personally Identifiable Information
PKCE	Proof Key for Code Exchange
PKCS	Public Key Cryptography Standards
PSCR	Public Safety Communications Research
PSFR	Public Safety and First Responder
PSO	Public Safety Organization
REST	Representational State Transfer
RFC	Request for Comments
RP	Relying Party
RSA	Rivest, Shamir, and Adelman
RSAES	RSA Encryption Scheme
SaaS	Software as a Service
SAML	Security Assertion Markup Language
SMTP	Simple Mail Transfer Protocol
SP	Service Provider, Special Publication
SSO	Single Sign-On
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
XACML	Extensible Access Control Markup Language

XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language