Building a Well Managed Cloud Application

Okta Inc.
301 Brannan Street
San Francisco, CA 94107

info@okta.com
1-888-722-7871

Contents

Introduction

Okta is the market leading on-demand identity and access management service that enables enterprises to accelerate the secure adoption of their cloud and web based applications.

Okta supports integration with a variety of single sign-on options across the hundreds of applications in the Okta Application Network, including delegated authentication, standard-based protocols such as Security Assertion Markup Language (SAML), or even proprietary vendor specific protocols.

For those customers with Microsoft Active Directory (AD) deployments, Okta also provides a simple, wizard driven process to integrate with AD as the authoritative identity store. The integration allows user synchronization, application provisioning and de-provisioning to extend beyond the on-premise environment controlled by the domain controller. User access to Okta itself can be delegated to the on premise instance of AD. All of this is done via the Okta Active Directory agent that can be easily installed and configured without additional network or firewall configuration.

To both improve the depth of integration with our current Independent Software Vendor (ISV) partners and to provide guidance to future ISV partners, we have developed this whitepaper to better articulate the range of integration options that should be considered by ISVs. The end goal of both this guidance and our collaboration with partners is to improve the ability of our joint customers to more broadly adopt, deploy and use cloud based applications.

Working with Okta

If you are like most rapidly growing SaaS ISVs, your engineering resources are focused on creating features for your customers that help differentiate the business functionality of your product in the market. You may not have the engineering expertise or the desire to grow it, or tackle the range of user authentication and user management integration challenges that customers are asking you to solve as your application is adopted more broadly across their organization.

Partnering with Okta can help you address these challenges.

By working with Okta, you can:

- Increase user adoption and overall usability across your installed base by simplifying user access and change management.
- Integrate into on premise, corporate directories, such as Microsoft Active Directory.
- Take advantage of best-of-breed authentication and provisioning capabilities without distracting your developers from your core product functionality.
- Reduce friction in sales cycles by partnering with IT around service deployments, user management, and security as well as integration with behind-the-firewall environments.

A Well Managed Cloud Application

When considering what it means to be a well-managed application, ISVs should think about end-users, IT, Business Administrators, and Business leaders.

Ease of use is essential for users when accessing any resources or applications. IT has to strike a balance between ease of use and security. A well-defined identity infrastructure is important. But just as important is the ability for each application to integrate with this infrastructure and with other applications. End-users do not want to remember different passwords for different systems. The ability to single sign-on with a single set of credentials eliminates password fatigue and reduces extra overhead of unnecessary help desk calls for password resets.

Customers also care about usage, productivity, and ROI. They want to know their users are getting access to the application when they need it, that the application is improving their competitiveness, and that they are getting a great return on their investment.

Externalizing user management capabilities via APIs enables IT administrators to efficiently set up application accounts for their end-users. It also improves security by allowing customers to centrally manage application access and disable access in a timely fashion.

Capturing application usage and account information helps customers to understand how the applications are being used across the board. Providing auditable and detailed reporting on user activities and permissions is important in satisfying customer compliance initiatives.

Okta strongly encourages ISVs to implement these features to help improve manageability of your application. These are also some of the key criteria identified by many customers and they are reflected in our Okta scale. The more criteria satisfied by your application, the higher the Okta ranking will be for your application. Customers can use the scale when they are evaluating new cloud applications they want to adopt, and ISVs can use the scale to help ensure their applications continue to meet market demand. For more detailed information about the Okta Scale, visit www.okta.com/why-okta/okta-scale.html.

Whether or not your customer is working with Okta, these features will guarantee better end-user and IT adoption by allowing tighter integration with your customers' existing IT infrastructure—a major criterion to deployment success of any cloud application.

Single Sign-On and Authentication

User Authentication or single sign-on is a simple concept and is also one of the most challenging problems to solve. Most applications start with a standalone model where users and their credentials are created, stored and managed within the application itself. Each application the enterprise adds to its portfolio also adds to these identity islands. Dealing with multiple applications becomes a hassle – both for the end-users and for administrators fielding all the issues from password resets to account lockouts.

For customers, especially those with a reliable authentication mechanism already in place, the solution is to centralize access. Rather than authenticating a user with another set of credentials, many customers prefer the vendor to provide the option to integrate with an existing identity provider of choice to authenticate and authorize a user. There are several approaches.

Federated Single Sign-On

Federated Single Sign-On or Identity Federation is a common approach when the customer wants an application to rely on an existing identity provider. In a federated scenario, rather than authenticating a user within your application, your application establishes a trust relationship with the customer's chosen identity provider and allows any user authenticated through the identity provider to have access to your application. For example, if your application relies on Yahoo as the identity provider, any user with a Yahoo account who has been authenticated through Yahoo will have access to your application. This identity provider may be another trusted application, a single sign-on solution or any other system that is aware of all the users and has the ability to authenticate and maintain an authenticated session.

Federation with Security Assertion Markup Language (SAML)

SAML has been the most widely used standard for implementing federated single sign-on. In the SAML terminology, the identity provider authenticates the users for the relying parties. In this case, your application is the relying party integrated with the customer's identity provider of choice. Okta supports SAML integration with many application vendors. The following is a guideline to help you with your SAML implementation including best practices in helping customers deploy a SAML integration between your application and their identity providers.

1. Implementing SAML: As the service provider, you will be responsible for issuing the initial SAML request to the identity provider. You also need the ability to receive and parse the SAML assertion coming back from the identity provider. An understanding of the SAML protocol is obviously required.

   Oasis provides a lot of material on SAML. A good technical overview is available here: Security Assertion Markup Language (SAML) V2.0 Technical Overview. These types of tutorials are the best place to start. You do not need to read through the entire SAML 2.0 specification. As a reference, here is the specification published on OASIS: http://www.oasis-open.org/specs/#samlv2.0.

   From an implementation standpoint, many open source libraries are available: http://saml.xml.org/wiki/saml-open-source-implementations provides a list of some of the commonly used libraries including OpenSAML.

   See "Appendix" for a sample SAML request and a SAML assertion.

2. Testing Your Implementation: Once you have completed your implementation, you need to test your application with a SAML-aware identity provider. If you do not have an identity provider for testing, Okta can help you with your testing and validation. Just as we integrate with many of the vendors through SAML, Okta will act as the identity provider where users are being authenticated.

3. Self administration and set-up: A successful SAML implementation does not automatically translate to customer success. A federated solution must be easy to integrate from your customer standpoint. A well-defined integration will save you and your customer valuable time and effort. Fortunately with SAML, it's possible to configure your application as a relying party with just a few key pieces of information from the identity provider. The best approach is to provide a self- administration console for customers to enter this information.

In most SAML integrations, the following information is typically needed from the identity provider:

- An issuer ID: the entity ID of the identity provider used in the assertion allowing your application to identify the issuer.
- An authentication certificate: issued by the identity Provider.
- User ID Type: determines which element is used by the SAML assertion to identify the user. This can be the username in your application or some other external ID.
- User ID Location: determines the location in the assertion where the user ID is identified which can be in the
- <Subject> statement or in the <Attribute> section of the assertion. For the latter, the attribute name must also be specified.
- Login URL: specifies the Identity Provider URL where your application needs to obtain the SAML assertion.
- Logout URL: specifies the URL where your application will direct the user upon a logout from your application.

Note: The parameters above are based on a SAML 2.0 implementation. SAML 1.1 differs slightly.

Here is a self-administration example for SAML configuration from Google Apps.

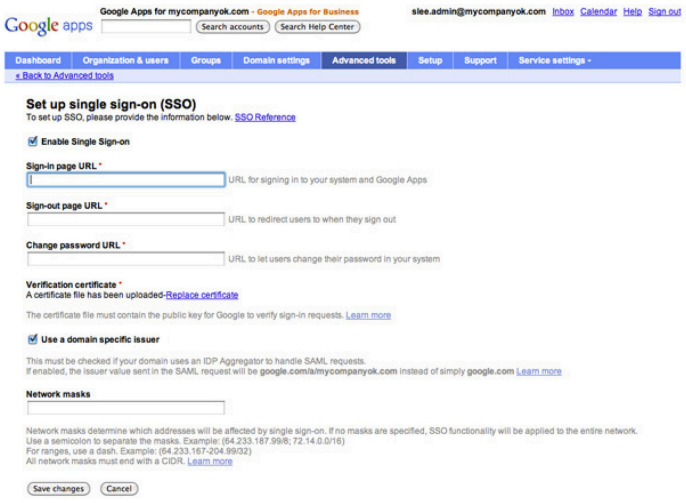

Figure 1: Google Apps SAML Configuration

In Google Apps, the SAML configuration is accessible by the administrator under the domain management section. As you can see, the parameters are clearly listed and can be easily configured. Information from the identity provider such as certificate related information and the assertion format is required. The customer will be responsible to gather this information from the chosen identity provider.

When using Okta as an identity provider, we provide vendor-specific integration guidelines to make this easier and error-proof for customers. Okta will show customers the exact information needed for your SAML configuration. A few cut and pastes and your application is federated with Okta!

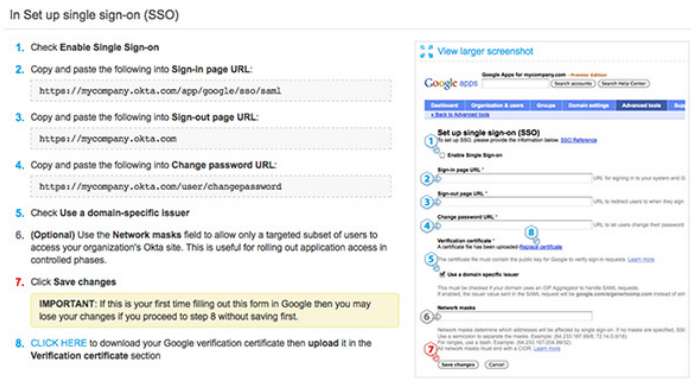

Figure 2: Google Apps SAML Configuration with Okta

4. Partial rollout: A shift from local authentication to SAML integration is considered a major change from an IT perspective especially when most of your customers are dealing with live production instances. To minimize risk and to allow feedback from end-users, most IT administrators prefer the option to perform a partial rollout with a selected group of users before rolling out to the entire user base. You may implement this based on existing groupings of users within your product. At a minimum, you should allow individual users to be selected. A back door login URL for emergency admin access bypassing the SSO settings is also extremely useful to avoid locking-out the entire instance during troubleshooting.

5. Partnering with Okta: For Okta customers using your application, end-users can easily integrate your application with Okta via SAML integration with Okta being the identity provider.

Delegated Authentication

Another way to centralize authentication is through delegated authentication. In a delegated authentication scenario, a user is still being authenticated in your application. Instead of using local credentials in your application, customers would like to leverage user credentials residing in an existing identity provider. This is slightly different from the federated use case. In federation, your application grants access to a user based on an authenticated account with the identity provider.

In delegated authentication, the user is authenticating to your application and only your application. The validation of user credentials is simply delegated or outsourced to an existing identity provider.

In a typical implementation, the vendor would send a request to the identity provider containing the user and relevant credentials to be authenticated by the identity provider. The format of the message should be determined by your application and made available to identity providers in order to implement the appropriate integration. Sample ASP scripts or WSDL (Web Service Definition Language) are some of the ways to communicate this information. Identity providers will then implement according to the templates and identify a URL endpoint exposing the delegated authentication functionality. This URL endpoint should be configurable through a self-administration console and will be used for redirection when a user logs in.

Here is an example of configuring delegated authentication for Zendesk:

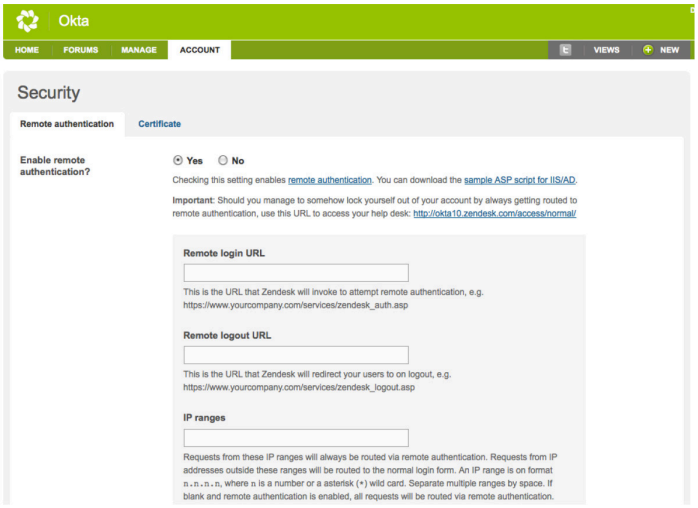

Figure 3: Zendesk Remote Authentication Configuration

Zendesk provides a remote authentication configuration page allowing customers to easily populate identity provider related information. A sample ASP template is provided to help implement the desired feature in the identity provider. In addition, Zendesk offers an IP range option to enable the remote authentication feature for limited IPs only.

See "Appendix" for a sample WSDL for delegated authentication.

When integrating with Okta as the delegated authentication provider, we provide step-by-step instructions, walking customers through the entire setup.

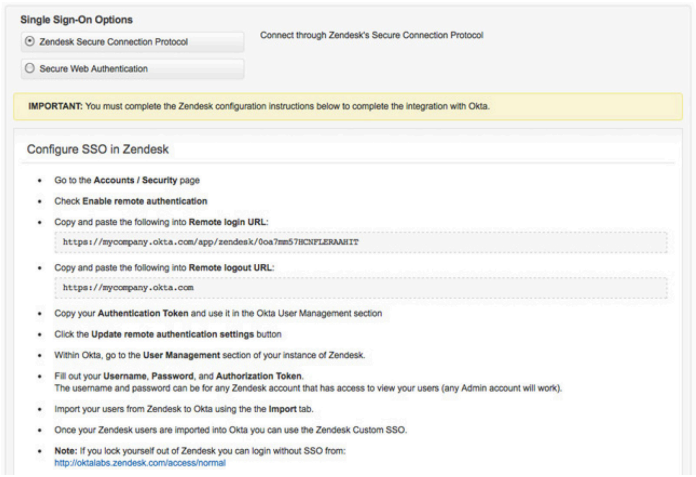

Figure 4: Zendesk Remote Authentication Configuration with Okta

Similar to setting up SAML, the vendor should provide the ability to support partial rollout and provide back door access for troubleshooting.

Support for both Federation and Delegated Authentication

A customer may choose between a federated single sign-on approach and a delegated authentication approach to integrate your application. However, the two are not mutually exclusive and there are cases where a combination of both is needed. Most federated single sign-on solutions today are tailored for web-based applications as they rely on the browser to act as the liaison or the agent between your application and the identity provider. This restriction limits these federated solutions when handling authentication through mobile clients and other non-web-based clients such as traditional thick clients. Delegated authentication provides an alternative by allowing logins from these non-browser-based clients to leverage the same credentials from a single identity provider.

Directory Integration

For many customers, a corporate directory (LDAP) serves as the authoritative source of identities and it is often used as the identity store behind a SAML identity provider or a single sign-on solution. Many on-premise applications support LDAP integration out-of-the-box. For cloud vendors, many customers will request for such integration, particularly those with Microsoft Active Directory.

Directory integration is useful in a couple of ways. It can be used for delegated authentication. For example, customer may want users to login to your application using their Microsoft Active Directory credentials. For Okta customers, they can set up SAML integration with Okta as the Identity Provider for your application, or set up delegated authentication from your application to Okta.

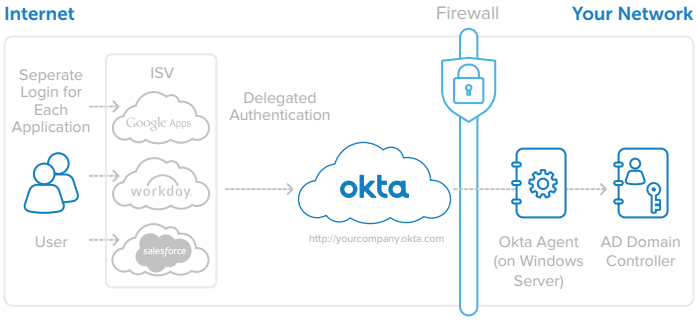

Figure 5: Delegated Authentication from ISV to Okta

In both cases, Okta itself can be configured with the customer's Microsoft Active Directory for delegated authentication. So whether your application is configured for SAML or for delegated authentication, users will be logging in using their Active Directory Credentials.

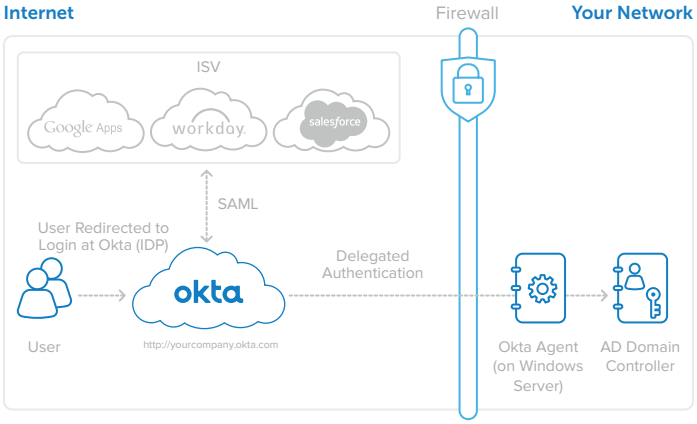

Figure 6: SAML between ISV and Okta

Directory integration also plays a crucial role in the identity life cycle—from account provisioning, and de-provisioning , to access provisioning. Application accounts may need to be created based on directory users. User attributes in the directory may need to be synchronized with your application user profile. Group memberships in the directory may be used to define authorization policies in your application. The next section will cover more about user and access management. In general, to be a well-managed cloud application, directory integration is a prerequisite.

Okta has implemented a robust integration with Microsoft Active Directory to support both delegated authentication as well as user and group management in an easy-to-manage and highly available manner. For any Okta customers, your well-managed application can rely on Active Directory for authentication, user provisioning, user profile and

User and Access Management—the Identity Lifecycle

Whether you are familiar with the term "identity lifecycle" or not, your application is dealing with it.

- How is a user account created in your application?
- How is a user profile updated?
- How to limit or grant user access in your application?
- How are users synchronized with the customer identity source?
- How to help your customer align access policies across different applications?
- How to deactivate an account?

Most vendors provide administrative consoles to perform these tasks manually. For many customers, these operations are often tied to the overall identity lifecycle of each user. For example:

- A HR system creates the initial footprint of an employee, which should then trigger a new domain user to be created in Active Directory.
- A VOIP telephone account should be created automatically for all new employees.
- Everyone in the sales division should have an account in the corporate CRM application with basic privileges assigned.

When an employee leaves the company, their access to the various applications should be revoked.

An Identity lifecycle defines the different phases a user goes through including onboarding, user profile changes, job changes and exit. These changes are typically triggered through an HR system or a directory such as Microsoft Active Directory. With Okta and its user and group management capabilities, these changes can be recognized and used to trigger appropriate actions in the integrated applications. By exposing the necessary user and access management APIs, your well-managed application can rely on an external system like Okta to manage and monitor users and access within your application. At a minimum, the APIs should support the following:

1. Account Creation: Typically, this involves the creation of a user profile and setting of various user attributes such as user login name, first name, last name, email, phone numbers, etc. An API to determine if an account already exists is a useful accompaniment.
2. User Profile Management: An application user profile may need to be updated during its lifetime. As an application, you want to avoid stale user profile information as much as possible. In some cases, such as an email change, the application may not function properly without the update. Your user management API should allow a client to set and retrieve profile information from the application.
3. Deactivation/Reactivation: When a user should no longer have access to the application, the account should be de-activated or disabled. Such an event is typically triggered by an external identity lifecycle event. The employee may be changing his job role or leaving the company and no longer require access to the application. From a compliance standpoint, account deactivation is crucial to disallow access. Your user management API should support deactivation—and also reactivation since there are cases where reactivation is necessary.
4. Authorization Management: Whether your application uses group memberships or other types of authorization methods, they must be associated with users to take effect. Exposing APIs for group and authorization policy management allows authorization to be driven and managed by an external system. Your customer may want to assign all the sales employees a particular set of application privileges—or have the membership of a Microsoft Active Directory security group synchronized with an application group within your application.

Having your well-managed application integrated with Okta, customers now have centralized user and access management across all the well-managed applications integrated within Okta—allowing them to efficiently provision accounts and access to users. Equally important is the ability to de-provision access in a timely fashion in case of an employee's departure or a change of job responsibilities.

With the entire audit trail of user activities and application access maintained within Okta, your customer can easily report on the current and historical view of this data to answer some key questions.

- Who has an account in the CRM application?
- What permissions does a user have in that application?
- Who had access to the application six months ago?
- Who accessed the application and when?

Okta helps satisfy this key audit and compliance requirement for your application, and for customers.

Beyond User and Access Management

Okta's initial service is built on a secure, reliable and extensible on-demand, multi-tenant cloud services platform. That platform will be the foundation for a growing set of core Okta and partner services that extend beyond the current identity and access management functionality. Our initial Okta Scale is also focused on measuring the ability of an application to meet the identity and access management requirements of enterprise customers. Over time, when we think of a well-managed application, we are thinking beyond user management, authentication and authorization.

Below are a few examples of what we have seen supported across various vendors. We would encourage you to consider these and other new customer challenges as you think about maturing the manageability of your services.

1. Centrally manage application licenses and subscription: From a vendor standpoint, there may be a single contract license and subscription per customer. For customers, however, they are dealing with many cloud vendors. Having all the licensing and subscription information in one place allows them to reference the information while monitoring the application. For example, allowing customers to quickly see the number of seats used or remaining.
2. Monitoring and reporting of important application system events: Certain application system events highlighting user access and user activity may be useful for customers from a monitoring and audit standpoint. Having APIs exposing the event information allows customers to fetch this data for the purpose of reporting, auditing, or analysis. Cloud vendors such as Google Apps and Salesforce. com expose APIs for audit purpose. Events related to System status or changes that may impact a customer's experience should also be made available through APIs.
3. Obtaining updates and product news: Providing operational updates of your application through feeds is another way to provide additional insight to customers. In cases where the information is not customer specific, such as site outage, updates or product news, many vendors exposed these through web feeds such as RSS and Twitter. For customers—both end-users and administrators, these feeds open up another communication channel they can optionally subscribe to.
4. Reporting and analyzing of Application Usage: Depending on the type of application, application usage may be important for customers to evaluate the overall usage by users. For example, a VoIP vendor can expose an API to allow usage information of each user to be fetched.
5. Centrally managing notifications across applications for end-users and administrators: Many applications send notifications to users and administrators through email. Exposing these notifications through API gives the ability for your customer to manage these notifications in other ways. For example, a notification may trigger an SMS sent out to the end-user through a corporate-wide SMS system. A corporate user portal can pick up notifications across different applications and expose a consolidated task list to the end- user when he arrives at his home page first thing in the morning.

Exposing this additional information allows Okta to provide this information to our joint customers. Customers can now choose how they want to use this information. But the greatest value is that customers can now gain a much greater insight of its entire cloud network through a pre-integrated, centralized framework delivered through Okta. As the Okta scale evolves, these will almost certainly become a part of the rating criteria along with other new and evolving customer requirements.

Conclusion

Addressing these customer requirements and providing a well-managed application for your customers will be crucial to your success as a vendor and to the success of cloud computing in general. It gives customers the ability to centralize authentication, and user and access management of your application. From a business standpoint, it allows you to externalize some of the most important and yet complex identity management requirements. This gives customers more flexibility in terms of integration options. More importantly, it saves you the hassle to tackle these issues, allowing you to focus on your core business.

At Okta, we believe in the value of a well-managed application. We have created the Okta scale to identify the important requirements for ISVs and to provide a benchmark for customers to better evaluate their applications. When integrated with Okta, our joint customers automatically benefit from our rich support in application provisioning, single sign-on, access management, and reporting—bringing the best out of your application and allowing customers to centrally manage your application and other cloud applications in Okta—a win-win situation.

Appendix

Case Study—Salesforce.com

Saleforce.com provides a comprehensive set of integration features including many of the ones described in this whitepaper.

SAML and Delegated Authentication

For every Salesforce.com org, whether it is a production org, a sandbox org or a developer edition org, both SAML and delegated authentication are supported. A customer can easily access the configuration in the setup area of the product available to users with system administration privilege.

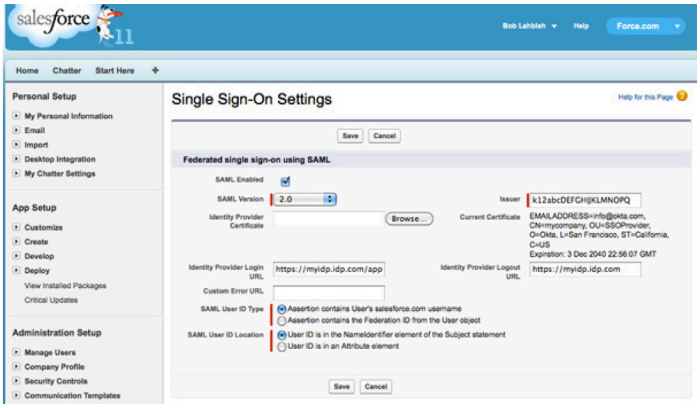

Figure 7: Salesforce.com SSO configuration

As shown in the above screenshot used for SAML configuration, Salesforce.com has provided a user-friendly interface making it extremely intuitive for customers to set up federated single sign-on on their own. In addition, the Help page also provides clear guidance and instructions about what parameters are needed and how to obtain them from the identity provider.

For delegated authentication, the customer must contact customer support to enable the feature. Once enabled, the configuration becomes available on this page with a single identity provider URL to be populated to specify the web service end point. A Delegated Authentication WSDL is available for download for the identity provider implementation.

Rolling out a single sign-on change is a daunting task for IT—especially when dealing with a large population of end- users. Salesforce.com allows you to selectively roll out these changes based on user profiles. For example, you can test out your delegated authentication setup with a small group of users sharing a common user profile and enable delegated authentication for that user profile only. This way, IT can test the solution with a small group of end-users before implementing a wider rollout.

User & Group Management

On the user management front, Salesforce.com supports most of the basic account management features including account creation, deactivation, reactivation, password reset and user profile updates. It also allows profiles and roles to be specified during user creation and to be modified later on. In essence, most of the operations allowed on the "Manage Users" page are available through the API.

Integration with the API is extremely simple and secured. A username with administrative privilege must be used in conjunction with a security token generated by Salesforce.com when authenticating through the API.

Lastly, all the APIs are well documented on Salesforce.com website. For example, here is the documentation on the APIs around a User object: Salesforce API Help

The integration steps have been implemented with the customer in mind. The configuration steps are mostly self- service driven without the need for customers to interact with the vendor at all. Understanding that applications do not function in a silo-ed fashion but in and amongst other applications and infrastructure components, these features have enabled Salesforce.com to integrate with their customers' IT infrastructure—allowing secured access, efficient management and simple integration with their Salesforce.com instances, positioning Salesfore.com as a well-managed cloud application.

Sample SAML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:AuthnRequest xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
AssertionConsumerServiceURL="http://www.sampleprovider.com/sso"
Destination="http://mycompany.okta.com/app/sampleprovider/saml"
ForceAuthn="false"
ID="_d1c7bb2ed91fc1982f11660497ff2dc0"
IsPassive="false" IssueInstant="2010-07-04T13:37:25.155Z"
ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Version="2.0">
<saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">http://www.
sampleprovider.com/sso</saml2:Issuer>
<saml2p:NameIDPolicy AllowCreate="false"
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
SPNameQualifier="http://www.sampleprovider.com/sso"/>
</saml2p:AuthnRequest>
```

Sample SAML Assertion

```
<samlp:Response
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
ID="r657673434904598469567" InResponseTo="r34943460856p567809567"
Version="2.0"
IssueInstant="2011-01-05T09:22:05Z"
Destination="http://www.sampleprovider.com/sso/response">
<saml:Issuer>https://www.okta.com/saml2</saml:Issuer>
<samlp:Status>
<samlp:StatusCode
Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
</samlp:Status>
<saml:Assertion
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
ID="r657673460856p567809567"
Version="2.0"
IssueInstant="2011-01-05T09:22:05Z">
<saml:Issuer>https://www.okta.com/saml2</saml:Issuer>
<ds:Signature
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
```

```
<saml:Subject>
<saml:NameID
Format=”urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress”>
test@okta.com
</saml:NameID>
<saml:SubjectConfirmation
Method=”urn:oasis:names:tc:SAML:2.0:cm:bearer”>
<saml:SubjectConfirmationData
InResponseTo=”r2187236345-04560-4596-0465”
Recipient=”http://www.sampleprovider.com/sso/response”
NotOnOrAfter=”2011-01-05T09:27:05Z”/>
</saml:SubjectConfirmation>
</saml:Subject>
<saml:Conditions
NotBefore=”2011-01-05T09:17:05Z”
NotOnOrAfter=”2011-01-05T09:27:05Z”>
<saml:AudienceRestriction>
<saml:Audience>http://www.sampleprovider.com/sso</saml:Audience>
</saml:AudienceRestriction>
</saml:Conditions>
<saml:AuthnStatement
AuthnInstant=”2011-01-05T09:22:00Z”
SessionIndex=”r4398458956912456234690 31”>
<saml:AuthnContext>
<saml:AuthnContextClassRef>
urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
</saml:AuthnContextClassRef>
</saml:AuthnContext>
</saml:AuthnStatement>
</saml:Assertion>
</samlp:Response>
```

Sample WSDL for Delegated Authentication Implementation

```
<!-- Sample Delegated Authentcation WSDL -->
<definitions targetNamespace=”urn:authentication.api.mysampleapp.com”>
<types>
<schema elementFormDefault=”qualified” targetNamespace=”urn:authentication.api. mysampleapp.com”>
<complexType name=”Authenticate”>
<sequence>
<element name=”username” type=”xsd:string”/>
<element name=”password” type=”xsd:string”/>
<element name=”sourceIp” type=”xsd:string”/>
<any namespace=”##targetNamespace” maxOccurs=”unbounded” minOccurs=”0”
processContents=”lax”/>
</sequence>
</complexType>
<complexType name=”AuthenticateResult”>
<sequence>
<element name=”Authenticated” type=”xsd:boolean”/>
</sequence>
</complexType>
<element name=”Authenticate” type=”tns:Authenticate”/>
<element name=”AuthenticateResult” type=”tns:AuthenticateResult”/>
</schema>
</types>
<message name=”AuthenticateRequest”>
<part element=”tns:Authenticate” name=”parameters”/>
</message>
<message name=”AuthenticateResponse”>
<part element=”tns:AuthenticateResult” name=”parameters”/>
</message>
<!-- Soap PortType -->
<portType name=”AuthenticationPortType”>
<operation name=”Authenticate”>
<input message=”tns:AuthenticateRequest”/>
<output message=”tns:AuthenticateResponse”/>
</operation>
```

```
</portType>
<!-- Soap Binding -->
<binding name=”AuthenticationBinding” type=”tns:AuthenticationPortType”>
<soap:binding style=”document” transport=”http://schemas.xmlsoap.org/soap/http”/>
<operation name=”Authenticate”>
<soap:operation soapAction=””/>
<input>
<soap:body parts=”parameters” use=”literal”/>
</input>
<output>
<soap:body use=”literal”/>
</output>
</operation>
</binding>
<!-- Soap Service Endpoint for MySampleApp to call the service -->
<service name=”MySampleAppAuthenticationService”>
<documentation>Authentication Service for MySampleApp</documentation>
<port binding=”tns:AuthenticationBinding” name=”AuthenticationService”>
<soap:address location=”http://localhost/”/>
</port>
</service>
</definitions>
```

User Management APIs

API Functionality	Description
Create User	Ability to create a new user using basic set of user attributes - first name, last name, email, etc.
Update User	Ability to subsequently update user profile. External changes on the user may require updates such as email update, status update (e.g. disable a user).
Get All Users	Ability to get a list of all the users (accounts) in your application. A client may wish to easily obtain a full list of accounts. In some cases, an import of users and their user profile details is needed. You should consider both use cases. Get User Status Depending on your application, they may be different types of status.
Set User Status	Ability to update user status is crucial. Again, the basic active/disabled should be supported. If there are other application specific statuses that should be exposed, make sure you include a way f or the client to fetch the list of values since they vary from application to application.
Get Permissions	Ability to set any permission related attributes or relationships for a user. Depending on your application, these could be attributes within the user profile. There might be groups, roles or profiles (or a combination of these) that need to be associated or granted to a user. The idea is to allow the API client to obtain a good picture of individual user access to your application.
Update Password	Ability to reset user passwords. This allows a client who may be centrally managing end-user passwords to synchronize passwords into your application. If your application has a configurable password policy, the policy should also be retrievable through some API—allowing your client to validate passwords accordingly if needed. Where possible, allow hashed values as your client may not have access to the clear text value. Let your client know the type of hash supported through documentation.

General API recommendations

1. Use query-based API for searching where possible. Often times, the parameters available to the client may not map to the unique identifier(s) in your API. Allowing flexible search criteria is key. Also, allow client to specify what to return. For example, an API to fetch a user attribute (like first name, or manager's name) should not return all the attributes available to the user and leave it up to the client to parse through the information. It is both tedious and inefficient.
2. Proper error handling is important. The returned error should provide sufficient details for the client to figure out what the problem is. For example, include short but precise text explanation as part of the error. If error codes are being used, make sure the meaning of each code is clearly documented.
3. A key feature often missing is API level logging on the application side. This is extremely useful when it comes to debugging issues during integration development. Without this, developer can only rely on returned errors if available or changes in the application itself to see if the desired behavior has been achieved. An API log showing clearly which API is being used along with the parameters that are passing through can help diagnose problems much quicker.
4. Think through each use case from the point of view of the API client. In most cases, the operations supported by these APIs are things that can be done in the product GUI involving one or more end users. Sometimes an operation may require multiple screens to capture the input. Certain sequential operations may require time delays in between or pending on another action to be executed. For example, workflows may be required due to end user approvals. Think of how an API client should cope with these complex scenarios. As part of the design, you may need to explore the option of a different workflow or business flow in the product to support your clients where it makes sense.
5. Good Documentation is essential. This is the main channel of communication between you and the client developer and should include code documentation, code samples, setup information, and error details.

Becoming an Okta ISV partner

Join hundreds of ISVs in the Okta Application Network. Let Okta help you transform your application into a coordinated and integrated part of our joint customers' IT infrastructure. To learn more, visit www.okta.com/ partners/isv-partners.html.About Okta

Okta is the foundation for secure connections between people and technology. By harnessing the power of the cloud, Okta allows people to access applications on any device at any time, while still enforcing strong security policies. It integrates directly with an organization's existing directories and identity systems, as well as 4,000+ applications. Because Okta runs on an integrated platform, organizations can implement the service quickly at large scale and low total cost. More than 2,500 customers, including Adobe, Allergan, Chiquita, LinkedIn, MGM Resorts International and Western Union, trust Okta to help their organizations work faster, boost revenue and stay secure.

For more information, visit us at www.okta.com or follow us on www.okta.com/blog.

About Okta

Okta is the foundation for secure connections between people and technology. By harnessing the power of the cloud, Okta allows people to access applications on any device at any time, while still enforcing strong security policies. It integrates directly with an organization's existing directories and identity systems, as well as 4,000+ applications. Because Okta runs on an integrated platform, organizations can implement the service quickly at large scale and low total cost. More than 2,500 customers, including Adobe, Allergan, Chiquita, LinkedIn, MGM Resorts International and Western Union, trust Okta to help their organizations work faster, boost revenue and stay secure.

For more information, visit us at www.okta.com or follow us on www.okta.com/blog.