|  |  |
| --- | --- |
| Implementers' Draft |  |
|  | Simple Auth Working Group |
|  | August 18, 2009 |

# Simple Auth

### Abstract

Simple Auth is a delegated authorization protocol which allows applications (Consumers) to easily access REST web service APIs at a Service Provider by presenting a short lived Access Token. Consumers acquire Access Tokens by presenting Token Issuers with long lived credentials. Token Issuers may accept multiple types of credentials and may be independant of the Service Provider. Token Issuers will typically perform the heavier authorization processes and return a short lived Access Token that is simple for the Service Provider to verify.

The Token Issuer may interact with a user directly to obtain authorization for the Consumer to access the Service Provider on behalf of the user. The Token Issuer will provide the Consumer with long lived credentials that can be presented to obtain an Access Token. This enables a Consumer to access a Service Provider on behalf of a user without having direct access to the user's credentials.

### Table of Contents

1. Overview
2. Simple Auth Dance
   1. Obtaining and Using Long Lived Credentials
      1. User and Browser
         1. Consumer Directs the User to the Token Issuer
         2. Token Issuer Directs the User back to the Consumer
         3. Requesting the Access Token
      2. Username and Password
      3. Account Name and Password
      4. SAML Assertion
   2. Obtaining an Access Token
      1. Access Token Format
   3. Using an Access Token
   4. Refreshing an Access Token
3. Security Considerations
4. References
5. Author's Addresses

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 1.  Overview

The **Consumer** is an application that wants to consume the services provided by the Service Provider. The Consumer may be making calls as itself or on behalf of a user. The Consumer may be a website, an application on a PC or mobile device, or an embedded application on a device.

The **Service Provider** provides REST web services. The Service Provider receives a short lived Access Token with each REST web service API call from the Consumer. By examining only the Access Token, the Service Provider can determine if it should execute the API call.

The **Token Issuer** provides authorization services for the Service Provider. The Token Issuer accepts long lived credentials from the Consumer, determines if the credentials are authentic, determines the access to the Service Provider the Consumer is authorized for, and returns a short lived Access Token to the Consumer. The Token Issuer may or may not be the same party as the Service Provider.

The steps in the protocol are:

1. The Consumer obtains long lived credentials directly from the Token Issuer or out of band as defined by the Token Issuer.
2. The Consumer presents the long lived credentials to the Token Issuer and receives a short lived Access Token.
3. The Consumer includes the Access Token when making REST web service calls to the Service Provider.
4. When the short lived Access Token is no longer valid, the Service Provider returns an error code to the Consumer, and the Consumer presents the long lived credentials to the Token Issuer to obtain a new, short lived Access Token.

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 2.  Simple Auth Dance

The Simple Auth Dance is the process in which applications (Consumers) obtain long lived credentials which are exchanged for short lived Access Tokens from the Token Issuer, and then used by the Consumer to access the Service Provider's REST web services.

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 2.1.  Obtaining and Using Long Lived Credentials

Depending on where the Consumer is running and if it is acting on behalf of a user: different types of long lived credential provisioning protocols are appropriate to balance ease of use, security and privacy. The Simple Auth Protocol defines some common, long lived credential provisioning protocols that may be accepted by a Token Issuer. The Token Issuer may define custom long lived credentials and provisioning protocols specific to the Token Issuer and/or Service Provider. The Token Issuer MUST accept at least one predefined OR one custom long lived provisioning protocol.  The Token Issuer MAY accept more then one long lived provisioning protocol.

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 2.1.1.  User and Browser

This long lived credential provisioning protocol is often appropriate when the Consumer wants to act on behalf of the user, is interacting directly with the user and a browser is readily available. The Consumer redirects the user to the Token Issuer, which authenticates the user directly. The Token Issuer then provides long lived credentials to the Consumer by redirecting the user back to the Consumer, or by providing them to the user who enters them into the Consumer. This enables the Consumer to act on behalf of the user without having direct access to the user's credentials.

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 2.1.1.1 Consumer Directs the User to the Token Issuer

The Consumer redirects the user's browser to the Token Issuer's User Authorization URL, with the following parameters:

sa\_consumer\_key:

REQUIRED. The consumer key

sa\_callback:

REQUIRED. An absolute URL to which the Token Issuer will redirect the User back after the user has approved the authorization request. Token Issuers MAY require that the sa\_callback URL is registered for the Consumer Key. Consumers which are unable to receive callbacks MUST use the special value "oob" to indicate it will receive the Verification Code out of band.

sa\_consumer\_state:

OPTIONAL. An opaque value that Consumers can use to maintain state associated with this request. If this value is present, the Token Issuer MUST return it to the Consumer's callback URL. Additional parameters: Any additional parameters, as defined by the Token Issuer. 

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 2.1.1.2.  Token Issuer Directs the User back to the Consumer

After the user authenticates with the Token Issuer and the user grants permission to the Consumer and the Consumer did not specify “oob” as its callback, the Token Issuer  redirects the user back to the Consumer's callback URL, with the following parameters added:

sa\_verifier:

REQUIRED. The verification code, if the user authorized the Consumer.

sa\_consumer\_state:

REQUIRED if the Consumer sent the value in the authorization request. The same value that the Consumer passed in [Section 2.1.1.1 (Consumer Directs the User to the Service Provider)](http://xml.resource.org/cgi-bin/xml2rfc.cgi#step%201).

Additional parameters:

Any additional parameters, as defined by the  Token Issuer.

If the Consumer specified "oob" as its callback, the Token Issuer MUST display a screen displaying the Verification Code, with instructions for the user to manually enter the Verification Code on the Consumer.

|  |  |
| --- | --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) | |
|  |

### 2.1.1.3.  Requesting the Access Token

The Consumer makes an HTTPS request to the Token Issuer's Access Token URL, using POST. The request contains the following parameters:

sa\_consumer\_key:

REQUIRED. The Consumer Key

sa\_consumer\_secret:

REQUIRED. The Consumer Secret

sa\_verifier:

REQUIRED. The Verification Code that was passed back to the callback URL, or if the Consumer specified "oob" as its callback, the Verification Code the user provided the Consumer.

sa\_callback:

REQUIRED. The callback URL that the Consumer used in [Section 2.1.1.1 (Consumer Directs the User to the Token Issuer)](http://xml.resource.org/cgi-bin/xml2rfc.cgi#step%201).

Additional parameters:

Any additional parameters, as defined by the Token Issuer.

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 2.1.2.  Username and Password

When a browser is not readily available to a Consumer and the Consumer has temporary access to the user's username and password for the Token Issuer, this provisioning protocol can be used. This allows the Consumer to access the Service Provider on behalf of the user without having to store the user's username and password.

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 2.1.2.1 Requesting a Verification Code

The Consumer makes an HTTPS request to the Token Issuer's Access Token URL, using POST. The request contains the following parameters:

sa\_consumer\_key:

REQUIRED. The consumer key

sa\_username:

REQUIRED. The user's username

sa\_password:

REQUIRED. The user's password

Additional parameters:

Any additional parameters, as defined by the Token Issuer.

If sucessful, the Token Issuer returns HTTP 200 with the Verification Code in the response body. The response body contains the following parameters:

sa\_verifier

Required. The Verification Token

Additional parameters:

Any additional parameters, as defined by the Token Issuer.

The parameters are formatted as application/x-www-form-urlencoded per 17.13.4 of HTML 4.01.

In case of error, the Token Issuer returns one of the following HTTP Status codes:

HTTP 401

The username and/or password are invalid. The Consumer should NOT retry the request using the same username and password.

HTTP 500

The Token Issuer is temporarily unavailable. The Consumer should retry later. Developers should consult the Token Issuer's documentation for additional information.

If HTTP 401 is returned, then the Token Issuer must also include the HTTP header:

WWW-Authenticate: SimpleAuth

The Consumer MUST discard the user's username and password once the Verification Token has been acquired.

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 2.1.2.2 Requesting an Access Token

The Consumer requests an Access Token as in 2.1.1.3. with the sa\_callback parameter set to "username-password".

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 2.1.3.  Account Name and Password

For a Consumer that is representing itself, an account name and password are well understood and deployed credentials. If the Consumer is acting on behalf of a user, this mechanism is NOT recommended.

The Consumer makes an HTTPS request to the Token Issuer's URL, using POST. The request contains the following parameters:

sa\_consumer\_name:

REQUIRED. The consumer name

sa\_consumer\_password:

REQUIRED. The consumer password

Additional parameters:

Any additional parameters, as defined by the Token Issuer.

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 2.1.4.  SAML Assertion

SAML assertions are readily available in some environments. The Consumer makes an HTTPS request to the Token Issuer's URL, using POST. The request contains the following ??? URL encoded ???? parameters:

sa\_SAML:

REQUIRED. The SAML assertion.

Additional parameters:

Any additional parameters, as defined by the Token Issuer.

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 2.2.  Obtaining an Access Token

The Consumer makes an HTTPS POST request with long lived credentials per 2.1. The Token Issuer examines the request parameters and determines if an Access Token is to be issued.

If sucessful, the Token Issuer returns HTTP 200 with the Access Token in the response body. The response body contains the following parameters:

sa\_token:

REQUIRED. The Access Token.

sa\_token\_expires\_in:

OPTIONAL. The lifetime of the Access Token in seconds. For example, 3600 represents one hour.

Additional parameters:

Any additional parameters, as defined by the Token Issuer.

The parameters are formatted as application/x-www-form-urlencoded per 17.13.4 of HTML 4.01.

In case of error, the Token Issuer returns one of the following HTTP Status codes:

HTTP 401

The long lived credentials are invalid. The Consumer should NOT retry the request using the same long lived credentials.

HTTP 403

Either the sa\_verifier or the sa\_callback is invalid. The Consumer should request that the user reauthorize the Consumer and restart the Simple Auth Dance from the beginning.

HTTP 500

The Token Issuer is temporarily unavailable. The Consumer should retry later. Developers should consult the Token Issuer's documentation for

additional information.

If HTTP 401 is returned, then the Token Issuer must also include the HTTP header:

WWW-Authenticate: SimpleAuth

### 2.2.1  Access Token Format

The Simple Auth Protocol does not specify the format of the Access Token. The format is mutually agreed to by the Token Issuer and the Service Provider.

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 2.3.  Using an Access Token

The Consumer accesses Protected Resources by including the Access Token in the HTTP Authorization header using the SimpleAuth scheme.

Authorization: SimpleAuth AccessToken

Where AccessToken is the Access Token.

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 2.4.  Refreshing an Access Token

To minimize the duration in which a compromised Access Token is vulnerable to replay attacks, Token Issuers SHOULD periodically expire Access Tokens and require Consumers to refresh them. Service Providers should return HTTP 401 to Consumers that attempt to access Protected Resources using an expired Access Token.

Upon receiving the HTTP 401 response when accessing Protected Resources, the Consumer should request a new Access Token by repeating the steps in [Section 2.2 (Obtaining an Access Token)](http://xml.resource.org/cgi-bin/xml2rfc.cgi#step%202)

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 3.  Security Considerations

... to be filled in ...

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 4. References

|  |  |
| --- | --- |
| [OAuth Core 1.0 Revision A] | OAuth Core Workgroup, “[OAuth Core 1.0 Revision A](http://oauth.net/core/1.0).” |
| [OAuth Problem Reporting Extension] | Kristian, J., “[OAuth Problem Reporting Extension](http://oauth.pbwiki.com/ProblemReporting).” |
| [OAuth Session Extension] | Tom, A., Alavilli, P., and G. Fletcher, “[OAuth Session](http://oauth.googlecode.com/svn/spec/ext/session/1.0/drafts/1/spec.html).” |
| [HTML 4.01] | <http://www.w3.org/TR/html401> |

|  |
| --- |
| [TOC](http://xml.resource.org/cgi-bin/xml2rfc.cgi#toc) |

### 5. Author's Address

|  |  |
| --- | --- |
|  | Simple Auth Working Group |
| Email: | [atom@yahoo-inc.com](mailto:atom@yahoo-inc.com) |
| Email: | dick.hardt@microsoft.com |