Client Development Guide

for

Maestro HMAC Silent Login

Prepared by Ryan Bergman

GeoLearning, Inc.

04-22-2009

Table of Contents

1. Introduction.......1
1.1 Purpose Of Silent Login....... 1
1.2 Silent Login vs "Single Sign On"?....... 1
2. Protocol Prerequisites.......1
2.1 Requirements for Client Development....... 1
2.1.1 Programming Tools and Libraries....... 1
2.2 Scope of SSO....... 1
2.3 Security....... 2
2.3.1 Key Management....... 2
2.3.2 Exposed links....... 2
2.3.3 Password Settings....... 2
3. Protocol Details.......2
3.1 Process Flow....... 3
3.2 URL Formatting....... 3
3.2.1 Example of a properly formatted URL....... 4
3.2.2 Code Examples....... 4
3.2.3 HMAC Validation....... 4

Revision History

Name	Date	Reason For Changes	Version
Ryan Bergman	July 30 2007	Initial	1
Ryan Bergman	Aug 20 2007	Added SHA-1 support	1.1
Ryan Bergman	Aug 04 2008	Note on password settings	1.2
Andrew Suckow	Feb 19 2009	Corrected HMAC examples	1.3
Andrew Suckow	Mar 4 2009	Note on password settings	1.4
Ryan Bergman	04/22/09	Clarified difference between SSO and Silent Login	1.5

1. Introduction

1.1 Purpose Of Silent Login

The purpose of the silent login process is for clients who desire an integration between their proprietary systems and the GeoLearning LMS environment. This process removes the need for the majority of users to manually provide a username and password to gain access to the LMS. A user of the LMS may select a link to the LMS system from the client site, will be identified, and will gain access seamlessly to the LMS where they can access their specific training requirements and history.

1.2 Silent Login vs "Single Sign On"?

"Single sign on" refers to the user experience of seamless access to multiple applications without explicitly authenticating to each application. The authentication technology used to facilitate such a user experience varies from the simple and insecure mechanisms such as trusting the IP address of the referrer to more sophisticated protocols such as OpenID and SAML. Single Sign On can be implemented point-to-point from a given application to every other application that the user may navigate to, or they can be implemented in a manner that does not impose any constraints on the order of navigation between applications.

The following document describes the one way HMAC Silent Login protocol support in GeoMaestro and illustrates the context in which it is applicable. This protocol allows users to authenticate to the LMS, but does not provide any kind of single point of identity management. As such it is not a true SSO but rather a simple integration gateway.

2. Protocol Prerequisites

2.1 Requirements for Client Development.

2.1.1 Programming Tools and Libraries.

The HMAC SSO implements over a HTTP URL. You may use any programming language to produce the link but the end result must be a user accessing the link over HTTP in a browser. For this reason you may find that using a language designed for web programming may be better suited than languages like C++ which are more geared towards desktop development. Some examples include Java, Microsoft .NET, PHP, ColdFusion MX, Perl, Python and Ruby.

This implementation also requires the use of a Hex encoded message digest. SHA-1 is currently supported as the primary hashing algorithm. These algorithms are extremely common and are typically available as a native library in most modern languages. Check your platform documentation for details. If your platform does not provide message digests many 3rd party libraries exist.

2.2 Scope of SSO

The SSO protocol described below allows a portal or other third party application to authenticate a user to the GeoMaestro LMS without user interaction. We will refer to such applications as the origin application and the LMS as the target application. It does not allow a user logged in to the LMS to authenticate to an external application. There are four preconditions for this SSO protocol:

1) The origin application and LMS must share a secret-key to establish trust.
2) The origin application and LMS must have their clocks synchronized. A small amount of clock skew (in the order of a few minutes) may be allowed.
3) The user must already have an active account in the LMS.
4) The user’s LMS loginid/username must be known to the origin application.

2.3 Security

2.3.1 Key Management

Part of this SSO process is the sharing of a secret key which both client and LMS have access to. Your LMS will intrinsically trust anyone with access to generate a proper unused link. It is therefore important that you take steps to protect the secret key. A good guide to key management can be found from the National Institute of Standards and Technology:

http://csrc.nist.gov/publications/nistpubs/800-57/SP800-57-Part1.pdf

2.3.2 Exposed links

Because the silent login is time sensitive and protects against replay links (those already accessed once). It is recommended that the client script perform the link creation and then directly forward the user to the LMS rather than formatting the link and placing it as an anchor on a HTML page. This will prevent users from using links on cached versions of pages.

2.3.3 Password Settings

Although the silent login does not make direct use of passwords, user accounts must still be populated with them. Be sure to populate passwords with sufficiently complex random passwords. Users who navigate to the LMS without going through the silent login will still be presented with the default login screen which includes the option to retrieve a forgotten password through email.

Password expiration and change requirements will not be enforced so long as the user is using the silent login to gain access to the LMS. These settings can be found under *Password Security* options of the application manager. Ignored settings during silent login include “Password Life Time”, “Admin Password Life Time” and “Admin Dormant Time”.

Within the password settings is a setting called “SSO Life Time”. This settings is not leveraged by this silent login.

3. Protocol Details

The SSO protocol primarily consists of the origin application creating a URL for the LMS with specific parameters such as the username of the user, a timestamp, an identifier for the shared-key used and an HMAC created using the aforementioned data keyed using the shared key and hashed using SHA-1.

3.1 Process Flow

1) Origin application formats a URL for the pass-through and either presents it to the user or automatically forwards to the LMS.
2) The LMS will make its own HMAC and compare it to the one that was passed. If the two are not the same then an error will occur.
3) The process will check the time represented in the datetime field. This field must be within 5 minutes of the server's time. If it is not then an error will occur.
4) The process will check the HMAC to ensure it has not previously been used to authenticate. If it has then an error will occur.
5) The process will validate the username and DomainID against the database. If a user is found then they will be logged into the LMS.
6) If the "OriginalURL" param has been passed the system will forward the user to that URL.

3.2 URL Formatting

Note that all URL parameters should be URL encoded.

Name	Description	Sample Value
Required Parameters		
URL	The main URL of the request. This will contain the server URL as well as the domain folder of the LMS.	https://gm1.geolearning.com/geonext/
Login Page	This is the gateway page and dictates which Message Digest your going to use.	sha1login.geo
Username	The username of the person being submitted to the LMS for pass-through	John.Doe
Timestamp	UTC date-time conforming to ISO 8601 specified up-to the second	2007-07-30T09:42:30Z
ID	Unique identifier for the shared-key. GeoLearning will provide this value to you when your domain has been set up.	1000
HMAC	A SHA-1 hex encoded	9c0e3cafefa5b18bd7456f7fb6dce46

	hash of username, timestamp and shared key concatenated together. Note that there is no separator between the 3 values. Characters should be in lower case.	
Optional Parameters		
OriginalURL	The File path to the page you want the user to begin at. After authentication the user will be placed on this page. This consists of everything after the server.	/geonext/myrequiredtraining?nav=MyRequiredLearning

3.2.1 Example of a properly formatted URL

https://localhost/geonext/sha1login.geo?username=Jimmy%40jim.org&
timestamp=2007-07-13T06%3A34%3A51Z&id=1000&
hmac=e9c0e3cafefa5b18bd7456f7fb6dce46

3.2.2 Code Examples

Working code examples are available in a variety of languages. Please check with your implementation specialist for them. Current examples exist in:

- C#.NET
- Java
- ColdFusion MX
- PHP
- Ruby
- Python

3.2.3 HMAC Validation

Here are some example params, hmac string and the resulting sha-1 hash to use to see if you are producing the correct strings. If you are having difficulty producing the proper HMAC, check the following:

a) The three sections are not delaminated in any way.
b) Make sure you are not URL encoding the raw parameters. URL encoding should only happen on the URL.
c) Make sure the parameters are in the proper order (name + time + SecretKey)
d) Remember that Message Digests like SHA-1 are case sensitive.
e) Your HMAC is all lower case.

For example the first row of examples would be hashing a string which looked like:

“John.Doe2007-07-30T15:47:52Z03569AD3AFE0B31661F7BC592F2AD7BF8719B94”

and will hash into string like this:

“bd6cb27eb0b5ff841c2e3126da5fb503413faacd”

UserName	TimeStamp	SecretKey	Resulting SHA-1 HMAC
John.Doe	2007-07-30T15:47:52Z	03569AD3AFE0B31661F7BC592F2AD7BF8719B94	bd6cb27eb0b5ff841c2e3126da5fb503413faacd
hsimpson	2007-07-30T15:51:40Z	03569AD3AFE0B31661F7BC592F2AD7BF8719B94	26da2b3744e9fd5203400b796272a40dcb2a5bec
Marge	2007-07-30T15:53:11Z	CDjScoDzketGQ60c9VUWdTo7lCqDsll6ljJzFPNGDKz	740c637732dee6f9baf6e16b5b56d0497f19f46e

Appendix A: Glossary

HMAC: A keyed-hash message authentication code, or HMAC, is a type of message authentication code (MAC) calculated using a cryptographic hash function in combination with a secret key. As with any MAC, it may be used to simultaneously verify both the data integrity and the authenticity of a message. Any iterative cryptographic hash function, such as MD5 or SHA-1, may be used in the calculation of an HMAC; the resulting MAC algorithm is termed HMAC-MD5 or HMAC-SHA-1 accordingly. The cryptographic strength of the HMAC depends upon the cryptographic strength of the underlying hash function, on the size and quality of the key and the size of the hash output length in bit.

SHA-1: designed by the National Security Agency (NSA) and published by the NIST as a U.S. Federal Information Processing Standard. SHA stands for Secure Hash Algorithm. It produces a 160-bit digest from a message with a maximum length of (2^64 − 1) bits and is based on principles similar to those used in the design of the MD4 and MD5 message digest algorithms.

UTC: Coordinated Universal Time, also known commonly as Greenwich Mean Time or GMT is the root time zone of the world.