

Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: September 4, 2009

T. Chu  
M. Hamrick  
M. Lentczner  
Linden Research, Inc.  
March 3, 2009

**Open Grid Protocol: Authentication**  
**draft-hamrick-ogp-auth-00**

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on September 4, 2009.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (<http://trustee.ietf.org/license-info>). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

Authentication in the Open Grid Protocol establishes an application layer association between a client application and a remote service

responsible for managing the end user's identity. The objective of authentication is to verify the user of a client application possesses appropriate credentials before granting capabilities sufficient to assert control over the user's agent and digital assets.

#### Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].

## Table of Contents

<a href="#">1.</a>	<a href="#">Agent Login (Resource Class)</a>	<a href="#">4</a>
<a href="#">1.1.</a>	<a href="#">Introduction</a>	<a href="#">4</a>
<a href="#">1.1.1.</a>	<a href="#">Account identifiers and Agent identifiers</a>	<a href="#">5</a>
<a href="#">1.1.2.</a>	<a href="#">Flexible Authentication</a>	<a href="#">5</a>
<a href="#">1.2.</a>	<a href="#">Service Location</a>	<a href="#">6</a>
<a href="#">1.3.</a>	<a href="#">Inputs</a>	<a href="#">6</a>
<a href="#">1.3.1.</a>	<a href="#">Agent Identifier</a>	<a href="#">6</a>
<a href="#">1.3.2.</a>	<a href="#">Account Identifier</a>	<a href="#">6</a>
<a href="#">1.3.3.</a>	<a href="#">Hashed Password Authenticator</a>	<a href="#">6</a>
<a href="#">1.3.4.</a>	<a href="#">Challenge-Response Authenticator</a>	<a href="#">6</a>
<a href="#">1.3.5.</a>	<a href="#">PKCS#5 PBKDF2 Authenticator</a>	<a href="#">7</a>
<a href="#">1.4.</a>	<a href="#">Response</a>	<a href="#">7</a>
<a href="#">1.4.1.</a>	<a href="#">Success</a>	<a href="#">8</a>
<a href="#">1.4.2.</a>	<a href="#">Maintenance Deferred Success</a>	<a href="#">8</a>
<a href="#">1.4.3.</a>	<a href="#">Authentication Non-Success</a>	<a href="#">8</a>
<a href="#">1.5.</a>	<a href="#">Errors and Exceptions</a>	<a href="#">8</a>
<a href="#">1.5.1.</a>	<a href="#">Authentication Failure</a>	<a href="#">8</a>
<a href="#">1.5.2.</a>	<a href="#">Agent Selection Failure</a>	<a href="#">8</a>
<a href="#">1.5.3.</a>	<a href="#">"User Intervention Required" Failure</a>	<a href="#">9</a>
<a href="#">1.5.4.</a>	<a href="#">"Non Specific" Failure</a>	<a href="#">9</a>
<a href="#">1.6.</a>	<a href="#">Preconditions</a>	<a href="#">9</a>
<a href="#">1.6.1.</a>	<a href="#">Client Preconditions</a>	<a href="#">9</a>
<a href="#">1.6.2.</a>	<a href="#">Agent Domain Preconditions</a>	<a href="#">9</a>
<a href="#">1.7.</a>	<a href="#">Postconditions</a>	<a href="#">9</a>
<a href="#">1.7.1.</a>	<a href="#">Client Postconditions</a>	<a href="#">9</a>
<a href="#">1.7.2.</a>	<a href="#">Agent Domain Postconditions</a>	<a href="#">10</a>
<a href="#">1.8.</a>	<a href="#">Side Effects</a>	<a href="#">10</a>
<a href="#">1.9.</a>	<a href="#">Sequence of Events</a>	<a href="#">10</a>
<a href="#">1.10.</a>	<a href="#">Interface (POST)</a>	<a href="#">12</a>
<a href="#">2.</a>	<a href="#">Login-Time Maintenance (Resource Class)</a>	<a href="#">14</a>
<a href="#">2.1.</a>	<a href="#">Service Location</a>	<a href="#">15</a>
<a href="#">2.2.</a>	<a href="#">Verb</a>	<a href="#">15</a>
<a href="#">2.3.</a>	<a href="#">Inputs</a>	<a href="#">15</a>
<a href="#">2.4.</a>	<a href="#">Response</a>	<a href="#">15</a>
<a href="#">2.5.</a>	<a href="#">Interface (GET)</a>	<a href="#">16</a>
<a href="#">3.</a>	<a href="#">Security Considerations</a>	<a href="#">16</a>
<a href="#">4.</a>	<a href="#">IANA Considerations</a>	<a href="#">17</a>
<a href="#">5.</a>	<a href="#">References</a>	<a href="#">17</a>
<a href="#">5.1.</a>	<a href="#">Normative References</a>	<a href="#">17</a>
<a href="#">5.2.</a>	<a href="#">Informative References</a>	<a href="#">17</a>
	<a href="#">Authors' Addresses</a>	<a href="#">17</a>



## **1. Agent Login (Resource Class)**

### **1.1. Introduction**

Authentication is the first step in associating a client application with the agent domain (the remote service responsible for agent identity management.) Before a client application may interact with the agent domain or one or more hosts responsible for virtual world simulation, it must authenticate itself by presenting credentials demonstrating its right to control the agent. Authentication is the process of presenting an "Identifier" and an "Authenticator" to the agent domain and receiving a "Seed Capability" providing further access to system resources or an actionable error description. The protocol defines an identifier as an agent or account information, distinct from its related authenticator.

Authentication begins by requesting the agent\_login resource; that is, POSTing the "LLSD" description of an identifier and an authenticator to a well-known URL. The agent domain managing this resource then makes an access control decision based on the verity of the credential and the state of the agent domain. The result of this authentication, whether success or failure, it is returned to the client application via a LLSD message. The content and form of these messages are provided below in "LLIDL format." [[I-D.hamrick-llsd](#)]

The authentication process results in one of seven classes of response from the agent domain:

- o success
- o deferred success due to maintenance
- o authentication non-success due to missing secret
- o authentication failure
- o agent selection failure
- o "user intervention required" failure, and
- o "non-specified" failure.

Responses to authentication requests are successes, non-successes and failures. A "success" indicates the client application should have enough information to progress past the authentication phase and begin using the service. A "deferred success" implies use of the system will continue after a "short" period. In either case, the agent domain does not expect the client application to re-submit the



agent\_login request. Authentication "non-success" results from a client requesting per-agent or per-account authentication parameters. After sending a "non-success", the agent domain expects the client to resubmit the agent\_login request "shortly." Failures of all type indicate the agent domain believes a condition exists requiring explicit user intervention. In the case of an authentication failure, the user should either retry the authentication request or recover their password. A failure due to "user intervention required" indicates the agent domain believes the user's account is in a state that required "out of band" recovery. Reading and accepting the agent domain's Terms of Service or Critical Messages are examples of recovering from "user intervention required" failures. Non-Specified failures indicate a non-recoverable problem that is not defined in this specification.

The section below on Processing Expectations provides more guidance.

#### **1.1.1. Account identifiers and Agent identifiers**

Client applications may authenticate using an "Account Identifier" or an "Agent Identifier". Either type of identifier may be used for authentication. An agent domain MUST support one of the two types of identifiers, and MAY support both. Client applications SHOULD support both identifier types.

An "Account" is an administrative object holding one or more references to an "Agent." This is advantageous in situations where:

1. the agent domain does not wish to use an agent first name and last name to identify a user, but wishes to use another identifier (such as an email address or account number,) or
2. the agent domain wishes to allow users with several agents to authenticate with the same authenticator, freeing them from the requirement of memorizing each individual agent authenticator.

Please note this spec does not imply a structure to the account identifier. Though an agent domain may use an email address as an account identifier, the protocol does not require it and treats the identifier simply as an opaque sequence of octets.

#### **1.1.2. Flexible Authentication**

This revision of the Open Grid Protocol defines, but does not require the use of, three authentication schemes: hashed password, challenge-response and PKCS#5 Key Derivation 2.





## **1.2. Service Location**

Each Agent Domain MUST have a well known and published authentication URL. The Second Life agent domain authentication URL is:

<https://login.agni.secondlife.com/cgi-bin/auth.cgi>

## **1.3. Inputs**

LLIDL descriptions are provided below for both agent identifiers and account identifiers. Client applications may use either as the basis for authentication.

### **1.3.1. Agent Identifier**

An agent identifier contains the first and last name of an agent.

### **1.3.2. Account Identifier**

An account identifier must contain the account\_name key. This is the opaque sequence of octets used by the agent domain to identify the user. If an account is associated with multiple agents, the client application SHOULD include the first\_name and last\_name of the agent the user wishes to use.

### **1.3.3. Hashed Password Authenticator**

When a hashed password is used as an authenticator, the string '\$1\$' is prepended to the UTF-8 encoding of the password and processed with the MD5 cryptographic hash function. [RFC1321] This revision of the Open Grid Protocol specification requires the use of MD5 with the hashed password authenticator. It also requires the presence of the algorithm key, and that the value of this key be the string 'md5'. Note that future versions of this specification may ALLOW or REQUIRE the use of other cryptographic hash functions.

### **1.3.4. Challenge-Response Authenticator**

The Challenge-Response scheme allows the agent domain to select a session specific "Salt" to be used in conjunction with the user's password to generate an authenticator. In this scheme the authenticator is the hash of the salt prepended to the hash of '\$1\$' prepended to the password. This revision of the Open Grid Protocol specification requires the use of SHA256 with the challenge-response authenticator. [sha256] It also requires the presence of the algorithm key, and that the value of this key be the string 'sha256'. Note that future versions of this specification may ALLOW or REQUIRE the use of other cryptographic hash functions.



To retrieve a session specific salt for use with the Challenge-Response authentication scheme from the agent domain, the client application sends a login request with a Challenge-Response authenticator without the secret item. If the agent domain supports this authenticator, it MUST respond with a 'key' condition including a salt and MAY include a duration in the response. If the duration is present, it denotes the number of seconds for which the salt will be valid.

The Challenge-Response Authentication Scheme is not currently deployed on the Second Life Grid.

#### **1.3.5. PKCS#5 PBKDF2 Authenticator**

The PKCS#5 PBKDF2 authenticator is an implementation of RSA Labs' Public Key Cryptographic Standards #5 v2.1 Password Based Key Derivation Function #2. [[pkcs5](#)] In this scheme, the hash of the string '\$1\$' prepended to the password is used in conjunction with a salt, iteration count and hash function to generate an authenticator. This revision of the Open Grid Protocol specification requires the use of SHA256 with the PKCS#5 PBKDS2 authenticator. It also requires the presence of the algorithm key, and that the value of this key be the string 'sha256'. Note that future versions of this specification may ALLOW or REQUIRE the use of other cryptographic hash functions.

As with the Challenge-Response authenticator, the agent domain MUST include the salt and iteration count in its response to an authentication request that is made without a secret item. Conforming agent domains may include a duration in their response indicating the number of seconds for which the salt and iteration count will be valid.

The PKCS#5 PBKDF2 Authentication Scheme is not currently deployed on the Second Life Grid.

#### **1.4. Response**

The response to the agent login message is notice of one of seven "conditions":

- o authentication success
- o maintenance deferred success
- o authentication non-success
- o authentication failure



- o agent selection failure
- o "user intervention required" failure, and
- o "non-specific" failure.

The specification recognizes three "non-failure" responses:

#### **1.4.1. Success**

Upon success, the agent domain will respond with a message containing the "Agent Seed Capability". Receipt of this capability indicates authentication was successful. This capability is then used for further interactions with the system.

#### **1.4.2. Maintenance Deferred Success**

This condition indicates per-agent (or per-account) login-time maintenance is being performed. It is not an error. The response includes a maintenance cap the client application should use to get information about currently executing maintenance. For more information about maintenance, see the Maintenance section below.

#### **1.4.3. Authentication Non-Success**

Authentication Non-Success is the response given when a client queries the agent domain for agent-specific or account-specific authentication parameters. In that it is the expected response to such a query, it is not an error or exception. But it is not an indication of successful authentication.

### **1.5. Errors and Exceptions**

#### **1.5.1. Authentication Failure**

An authentication failure indicates the client application did not provide enough information to authenticate the account or the agent.

#### **1.5.2. Agent Selection Failure**

An agent selection failure occurs when an account authentication request is ambiguous. In other words, the account a user has attempted to use to log in is associated with more than one agent account and the client application did not specify which account to use. The response includes a list of first\_name / last\_name pairs. It is expected that the client application will present this list to the user and ask which agent to use.



### **1.5.3. "User Intervention Required" Failure**

This error indicates that the agent domain cannot authenticate the user for non-technical reasons. The protocol does not attempt to describe why, or imply remediation for this error. But an agent domain that returns this response **MUST** provide a URL containing a message describing the condition leading to the error and remediation, if known.

### **1.5.4. "Non Specific" Failure**

This error indicates some other error exists which does not fall into one of the previous six conditions.

## **1.6. Preconditions**

### **1.6.1. Client Preconditions**

It is generally assumed that before a user attempts to log into an agent domain, they will not be actively connected to that agent domain.

It is also assumed that the user has registered their account and/or agent; user registration is outside the scope of this specification.

The client application **SHOULD** present the agent domain's Terms of Service and Critical Messages and allow a user to accept or decline them prior to attempting to authenticate.

### **1.6.2. Agent Domain Preconditions**

If the agent domain requires users to read and agree to the Terms of Service or acknowledge receipt of Critical Messages prior to authentication, it must maintain a record of which accounts and agents have accepted and acknowledged these items.

Agent domains that support the concept of "suspension" or "disablement" should also maintain a record of which accounts and agents are suspended or disabled.

## **1.7. Postconditions**

### **1.7.1. Client Postconditions**

Following successful authentication, the client application **SHOULD** note that the agent has been authenticated to the agent domain. The Open Grid Protocol is **NOT** stateless.





### **1.7.2. Agent Domain Postconditions**

After an agent (or account) is authenticated, a seed capability is allocated for the agent. The agent domain SHOULD maintain the association between agent credentials (first\_name and last\_name) and the seed capability so it may be re-used if the client attempts to re-authenticate the user.

### **1.8. Side Effects**

The agent domain SHOULD maintain the "presence" state of an agent. This state should include the agent's seed capability. If a previously authenticated and "present" agent re-authenticates successfully, the agent domain MAY return the same seed capability.

After successful authentication, it is expected that the client will issue another request against the seed capability. To defend against potential Denial of Service attacks against the agent domain, the agent domain MAY define a timeout period for the seed capability. If the timeout period expires without a request being made against the seed capability, that seed capability will expire. Successful authentication of an agent who is "not present" has the effect of starting this timer.

The Challenge-Response Authenticator is intended to be used with a new, randomly generated salt for each authentication request. If the agent domain supports the Challenge-Response authentication scheme, it must maintain the "most recently generated salt" for some period of time (generally until the expiration of the duration period given in the authentication non-success response.)

After the salt has "timed out" following an unsuccessful Challenge-Response authentication request, the agent domain MUST NOT allow the use of a previous or fixed salt value. That is, it is not correct, after the salt has expired, to use a null, fixed or previous salt. The agent domain MUST generate a new salt and return it to the client application. An unsuccessful authentication request with the Challenge-Response scheme also has the side effect of starting the salt duration timer. When this timer expires, the agent domain MUST NOT allow authentication with previously generated salts.

### **1.9. Sequence of Events**

It is possible for an authentication request to occur in conditions where multiple errors or exceptions COULD be returned. As the protocol does not support reporting multiple failure conditions, the following sequence is provided to determine the priority of failure conditions. This sequence of events is motivated by the following



principles:

- o The agent domain should leak no account status information to an unauthenticated user.
- o Maintenance should occur after successful authentication and before account status checking in case maintenance involves the representation of these states by the agent domain.
- o The agent domain should check for "administrative issues" after maintenance is complete.

The sequence for authentication is as follows. At the first error, the system produces an appropriate error response.

1. If the authenticator provided is a Challenge-Response or PKCS#5 PBKDF2 type AND a secret is not included, the system returns an authentication non-success response.
2. The secret and optional authentication parameters are used to verify the client is in possession of the shared secret. If authentication is unsuccessful, an authentication failure response is returned.
3. If per-user login-time maintenance must be performed, the agent domain allocates a maintenance capability and returns it to the client application as a maintenance deferred success response.
4. If an account credential was used for authentication and the account "contains" two or more agents and the client application did not provide the first\_name and last\_name of the agent to log in as, generate a list of all agents associated with this account and return an agent selection failure response.
5. If an "administrative issue" exists such as the user is suspended, banned, must agree to the terms of service or read critical messages, the system returns a "user intervention required" response, providing a URL referencing a web resource explaining the administrative issue and describing remediation steps.
6. Check to see if the authenticated agent is associated with an agent seed capability already. If so, return a success response referencing that seed capability.
7. Start the seed capability timer. Allocate an agent seed capability and return it to the client application via a success response.







; agent identifier

```
&identifier = {  
    type: 'agent',           ; identifies this as an "agent identifier"  
    first_name: string,  
    last_name: string,  
}
```

; request

```
&credential = {  
    identifier: &identifier,    ; account or agent identifier  
    authenticator: &authenticator ; 'hash', 'challenge'  
                                   ; or 'pkcs5pbkdf2'  
}
```

; response

; successful response

```
&response = {  
    condition: 'success',  
    agent_seed_capability: uri    ; URL of the agent seed cap  
}
```

; authentication failure

```
&response = {  
    condition: 'key',  
    salt: binary,           ; optional - salt for challenge and PKCS5  
    count: integer,         ; optional - iteration count for PKCS5  
    duration: integer       ; optional - the duration of the validity  
                             ; period of salt and count values in  
                             ; seconds  
}
```

; maintenance "non success"

```
&response = {  
    condition: 'maintenance',  
    maintenance_capability: uri, ; URL of the maintenance cap  
    completion: integer          ; an estimate for maintenance duration  
                                 ; (in seconds)  
}
```

; agent select failure

```
&response = {
```





```
    condition: 'select',
    agents: [ { first_name: string, last_name: string } ... ]
}

; administrative failure

&response = {
    condition: 'intervention',
    message: uri                ; a URI with human-readable text
                                ; explaining what the user must do to
                                ; continue
}

; non-specific error

&response = {
    condition: 'nonspecific',
    message: string              ; a string describing the failure
}

; resource definition

%%agent_login
->&credential
<-&response
```

## 2. Login-Time Maintenance (Resource Class)

An agent domain has the option of performing "per-user, login-time maintenance" as part of the authentication sequence. Performing maintenance after a user is authenticated and before an avatar is "rezzed" in a region has several advantages:

- o it reduces system-wide downtime
- o it distributes maintenance across time, and
- o it consumes computational resources only for those agents who use the system

The agent domain signals it is performing maintenance by returning a "Maintenance Capability" instead of a seed capability following successful authentication. The maintenance capability represents a finite sequence of transactions performed by the agent domain on the user's behalf. It is expected that maintenance is a task that will complete in a "tractable" amount of time.

The maintenance capability may be queried to retrieve information



about the transactions that are occurring, including:

- o a textual description of the maintenance being performed
- o an estimate for how long the maintenance will take to complete

### **2.1. Service Location**

The agent domain may provide a maintenance capability to the client application in response to successful authentication. This capability is communicated as an URL to a web based service that accepts LLSD queries.

'maintenance' capability from

### **2.2. Verb**

GET

### **2.3. Inputs**

There are no parameters to a maintenance capability request.

### **2.4. Response**

There are three responses to a maintenance capability: a description of ongoing maintenance, a new maintenance capability describing another sequence of maintenance transactions, or a seed capability. These responses are identified with the condition items: 'ongoing', 'next' and 'complete'.

The 'ongoing' response to a maintenance capability request includes a simple textual description of the maintenance performed, an estimate for how long the maintenance is expected to take, and a validity duration for the capability. The estimate for how long maintenance will take is provided so client applications may provide feedback to the user. The validity duration gives the viewer a minimum time period the agent domain will maintain the maintenance capability.

When the agent domain returns a 'next' response, it indicates that the current maintenance is complete, but a new maintenance must be performed before the agent may be placed into a region. The 'next' response includes the URL of the next maintenance capability as well as an integer describing the minimum time period the agent domain will maintain the maintenance capability.

When an agent domain returns a 'complete' response, it indicates that all maintenance is complete. The response includes the agent seed capability that may be used to place the user's avatar in a region. It also includes an item describing the validity period for the



current maintenance capability.

## 2.5. Interface (GET)

The following text describes the LLIDL description of the agent\_login messages.

```
&response = {
  condition: 'ongoing',
  description: string,
  duration: integer,           ; seconds before maintenance is complete
  validity: integer           ; seconds before this capability expires
}

&response = {
  condition: 'next',
  description: string,
  maintenance_capability: uri ; URL for the next maintenance capability
  validity: integer           ; seconds before this capability expires
}

&response = {
  condition: 'complete',
  agent_seed_capability: uri  ; the agent's seed cap
  validity: integer           ; seconds before this capability expires
}

%%maintenance
->undef
<-&response
```

## 3. Security Considerations

[RFC 3552](#) [[RFC3552](#)] describes several aspects to use when evaluating the security of a specification or implementation. We believe most common security concerns users of this specification will encounter are more appropriately considered as transport, network or link layer issues. However, the following "application security" issues should be considered.

The MD5 cryptographic hash functions has been deprecated and SHOULD be used only for compatibility with older applications.

The use of the hashed password authenticator could result in a replay attack if not used in conjunction with an appropriate confidentiality preserving transport. Implementations using the hashed password authenticator SHOULD utilize appropriate encryption schemes such as TLS [[RFC5246](#)] or S/MIME [[RFC3851](#)].



#### **4. IANA Considerations**

This document has no actions for IANA.

#### **5. References**

##### **5.1. Normative References**

- [I-D.hamrick-llsd]  
Brashears, A., Hamrick, M., and M. Lentczner, "Linden Lab Structured Data", 2008.
- [RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", [RFC 1321](#), April 1992.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [pkcs5] Kaliski, B., "PKCS #5: Password-Based Cryptography Specification Version 2.0".
- [sha256] ""Federal Information Processing Standards Publication 180-2 (+ Change Notice to include SHA-224)".

##### **5.2. Informative References**

- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", [BCP 72](#), [RFC 3552](#), July 2003.
- [RFC3851] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", [RFC 3851](#), July 2004.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), August 2008.





## Authors' Addresses

Tess Chu  
Linden Research, Inc.  
945 Battery St.  
San Francisco, CA 94111  
US

Phone: +1 415 243 9000  
Email: [tess@lindenlab.com](mailto:tess@lindenlab.com)

Meadhbh Siobhan Hamrick  
Linden Research, Inc.  
945 Battery St.  
San Francisco, CA 94111  
US

Phone: +1 650 283 0344  
Email: [infinity@lindenlab.com](mailto:infinity@lindenlab.com)

Mark Lentczner  
Linden Research, Inc.  
945 Battery St.  
San Francisco, CA 94111  
US

Phone: +1 415 243 9000  
Email: [zero@lindenlab.com](mailto:zero@lindenlab.com)

