This document provides a identity based attestation multi-level security protocol. IBOPS provides Identity Assertion, Role Gathering, Multi-Level Access Control, Assurance, and Auditing. IBOPS includes software running on a client device (Android, iPhone, etc.), a trusted IBOPS Server, and an IDS (Intrusion Detection System). The IBOPS allows pluggable components to replace existing components functionality accepting integration into the current operating environments in a short period of time. The IBOPS adheres to the principle of continuous protection in adjudicating access to resources. Accountability is the mechanism that proves a service level guarantee of security. The IBOPS allows the systems to meet security needs by using the API (Application Programming Interface). The IBOPS need not know whether the underlying system is a Relational Database Management System (RDBMS) or a Search Engine. The IBOPS functionality offers a “point and cut” mechanism to add the appropriate security to the production systems as well as to the systems in development.
# Overview

1. Preamble

2. Purpose

## Definitions, acronyms, and abbreviations

## Conformance

## The Intended Audience

## Security Considerations

1. Identification Assertion

2. Role Gathering

3. Access Control
   1. Discretionary Access Control
   2. Mandatory Access Control

4. Audit and Assurance
   1. Audit
   2. System Integrity

## IBOPS Interoperability

## IBOPS Overview

1. Application

2. Registration
   1. Developers and the IBOPS Service
   2. The End User and the IBOPS Service
   3. The End User device and the IBOPS Service

3. Prevention of Replay
   1. The Format of all Requests
   2. Subsequent API calls

## IBOPS API Overview

1. Identity Assertion API
   1. Developer API Key
   2. Application Identification

2. Enterprise concepts
9.2 Format of API calls........................................................................................................24
9.3 Genesis..........................................................................................................................25
  9.3.1 Post genesis communication uses...........................................................................26
  9.3.2 Is the Device Blacklisted?.......................................................................................26
9.4 API - Genesis (Enrollment).........................................................................................27
  9.4.1 /RegisterAccount.....................................................................................................27
9.5 API - QROpportunity.....................................................................................................28
  9.5.1 /QROpportunity........................................................................................................28
  9.5.2 /enterprise/RegisterSessionOpportunity.................................................................29
  9.5.3 /enterprise/GetSessionStatus....................................................................................29
  9.5.4 /enterprise/RegisterSession.....................................................................................30
  9.5.5 /enterprise/AuthenticationResponse.........................................................................30
9.6 Role API..........................................................................................................................31
  9.6.1 Dynamic Image Code Session Construction at a Glance........................................31
9.7 Access Control API........................................................................................................33
9.8 Auditing..........................................................................................................................34
9.9 API for Read Audit Logs.............................................................................................35
9.10 Administration...............................................................................................................35
  9.10.1 To Add or Update a User.........................................................................................35
  9.10.2 To Add or Update a Group.......................................................................................36
  9.10.3 To Add a Role...........................................................................................................36
9.11 Reporting.......................................................................................................................37
10 Client Device Requirements..............................................................................................37
11.0 Server Side Intrusion Detection System.......................................................................38
  11.1 API List Blacklist.........................................................................................................38
  11.2 API - Incident...............................................................................................................38
Identity based Attestation and Open Exchange Protocol Specification (Draft)

1 Overview

1.1 Preamble
Convenience drives consumers toward the biometrics based access management solutions; say studies from Ericsson, PayPal, IBM, and Microsoft.

According to the Ericsson’s study “Your body is the new password”, 52% of smartphone users want to use their fingerprints instead of the passwords, a further 61% want to use fingerprints to unlock their phones, and 48% want to use eye-recognition.

The study conducted by PayPal says that consumers approve biometrics for access management. In terms of readiness to switch from an old fashion password protection to the new technology, 53% of surveyed population would be comfortable replacing passwords with the fingerprints and 45% would choose a retinal scan, which is presumably an iris scan – the misplaced terminology points to the lack of a consumer education.

IBM Fellow and Speech CTO David Nahamoo states that over the next five years, your unique biological identity and biometric data – facial definitions, iris scans, voice files, even your DNA – will become the key to the safeguarding of your personal identity and information and will replace the current user ID and password system.

Microsoft Research funded a study that titled “The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes”, the cornerstone conclusion of which indicates that the vast passwords replacement transition should conform to the following criteria: nothing to carry, efficient to use, and easy recovery from a loss. The Microsoft study goes as far as concluding such criteria could be achieved mostly in the biometric schemes.

IBOPS communication architecture enables 2-way Secure Socket Layer (SSL or Transport Layer Security (TLS)) connection over the encryption mechanism to the server, which employs Intrusion Detection System (IDS).
1.2 Purpose

This standard provides a multi-level security protocol. IBOPS provides Identity Assertion, Role Gathering, Multi-Level Access Control, Assurance, and Auditing. IBOPS includes software running on a client device (Android, iPhone, etc.), a trusted IBOPS Server, and an IDS (Intrusion Detection System). The IBOPS allows pluggable components to replace existing components functionality accepting integration into the current operating environments in a short period of time. The IBOPS adheres to the principle of continuous protection in adjudicating access to resources. Accountability is the mechanism that proves a service level guarantee of security. The IBOPS allows the systems to meet security needs by using the API (Application Programming Interface). The IBOPS need not know whether the underlying system is a Relational Database Management System (RDBMS) or a Search Engine. The IBOPS functionality offers a "point and cut" mechanism to add the appropriate security to the production systems as well as to the systems in development.

2. Definitions, acronyms, and abbreviations

For the purposes of this document, the following terms and definitions apply. This section is arranged alphabetically.

<p>| AOP          | Aspect Oriented Programming |</p>
<table>
<thead>
<tr>
<th><strong>API</strong></th>
<th>Application Programming Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>App</strong></td>
<td>A mobile client application</td>
</tr>
<tr>
<td><strong>Bell-LaPadula</strong></td>
<td>The multilevel model that was proposed by Bell and LaPadula for enforcing access control in government and military applications. A subject can only access objects at certain levels determined by his security level.</td>
</tr>
<tr>
<td><strong>IBOPS</strong></td>
<td>Identity Based Attestation and Open Exchange Protocol</td>
</tr>
<tr>
<td><strong>CPU</strong></td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td><strong>CSRF</strong></td>
<td>Cross-site request forgery</td>
</tr>
<tr>
<td><strong>CTO</strong></td>
<td>Chief Technical Officer</td>
</tr>
<tr>
<td><strong>DAC</strong></td>
<td>Discretionary Access Control</td>
</tr>
<tr>
<td><strong>DOS</strong></td>
<td>Denial-of-Service (attack)</td>
</tr>
<tr>
<td><strong>DDoS</strong></td>
<td>Distributed Denial of Service</td>
</tr>
<tr>
<td><strong>DNA</strong></td>
<td>Digital Network Architecture</td>
</tr>
<tr>
<td><strong>DNS</strong></td>
<td>Domain Name System</td>
</tr>
<tr>
<td><strong>GPU</strong></td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td><strong>GUI</strong></td>
<td>Graphic User Interface</td>
</tr>
<tr>
<td><strong>GUID</strong></td>
<td>Globally Unique Identifier</td>
</tr>
<tr>
<td><strong>HTTP</strong></td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td><strong>HTTPS</strong></td>
<td>Hypertext Transfer Protocol Secure</td>
</tr>
<tr>
<td><strong>IDS</strong></td>
<td>Intrusion Detection System</td>
</tr>
<tr>
<td><strong>IDAP</strong></td>
<td>Identity Assertion Platform</td>
</tr>
<tr>
<td><strong>IP</strong></td>
<td>Internet Protocol</td>
</tr>
<tr>
<td><strong>IMEI</strong></td>
<td>International Mobile Equipment Identity</td>
</tr>
<tr>
<td><strong>JSON</strong></td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td><strong>JSP</strong></td>
<td>Java Server Pages</td>
</tr>
<tr>
<td><strong>JUnit</strong></td>
<td>A testing framework for Java programming language</td>
</tr>
<tr>
<td><strong>LAN</strong></td>
<td>Local Area Network</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>LDAP</td>
<td>Lightweight Directory Protocol</td>
</tr>
<tr>
<td>MAC</td>
<td>Mandatory Access Control</td>
</tr>
<tr>
<td>MCA</td>
<td>Mobile Client Application</td>
</tr>
<tr>
<td>NMap</td>
<td>Network Mapper</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnection model</td>
</tr>
<tr>
<td>OWASP</td>
<td>Open Web Application Security Project</td>
</tr>
<tr>
<td>PC</td>
<td>A personal computer</td>
</tr>
<tr>
<td>RC</td>
<td>Release Candidate</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Database Management System</td>
</tr>
<tr>
<td>RESTful</td>
<td>Refers to REST Representational State Transfer, which is an architectural style.</td>
</tr>
<tr>
<td>SAML</td>
<td>Security Assertion Markup Language</td>
</tr>
<tr>
<td>SHA512</td>
<td>Secure hash algorithm three</td>
</tr>
<tr>
<td>SOCKS</td>
<td>Socket Secure (Internet Protocol)</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SSH</td>
<td>Secure Shell</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Socket Layers</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TCSEC</td>
<td>Trusted Computer System Evaluation Criteria</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform resource identifier</td>
</tr>
<tr>
<td>VPN</td>
<td>Virtual private network</td>
</tr>
<tr>
<td>WAR</td>
<td>Web Application Archive</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
3. Conformance

IBOPS comprises the rules governing secure communication between a variety of client devices and the trusted server. This standard is based on the tested computer based an implementation of the Trusted Computer System Evaluation Criteria (TCSEC).

This document demonstrates the incentive values of IBOPS and provides a comprehensive guidance to the IBOPS solution implementation. IBOPS conforms to the Trusted Computer System Evaluation Criteria (TCSEC), which is the US Government Department of Defense (DoD) standard that sets basic requirements for assessing the effectiveness of computer security controls built into a computer system, created by the National Computer Security Center (NCSC), an arm of the National Security Agency (NSA) and is also frequently referred as Orange Book, section B1; to the Director of the Central Intelligence Directive 6/3 protection levels 3, 4, and 5 (PL3, PL4, PL5); and to the standards of Multiple Independent Levels of Security (MILS).

4. The Intended Audience

The Intended Audience of this document includes security evaluators, system underwriters, developers, and systems engineers.

5. Security Considerations

Security considerations include the security policies in place and unambiguously defined levels of security. One of the IBOPS main functions is to provide Authentication instead of Authorization in a way where the server does not retain the client information, but recognizes one client from another. The key components of security considerations include Identity Assertion, Role Gathering, Access Control, Auditing, and Assurance.

5.1 Identification Assertion

The IBOPS provides continuous protection to resources and assurance of the placement and viability of adjudication and other key features. Accountability is the mechanism that proves a service level guarantee of security.

The IBOPS identity assertion provides a guarantee that named users are who they claim to be. The identity assertion implies reliance on human biometrics, however, the IBOPS is an interoperable standard and can incorporate any identity asserter, or a number of asserters that provides this guarantee. The application of the Intrusion Detection System (IDS) provides active monitoring to prevent spoofing of the credentials set and blacklisting of a subject or device that makes malicious attempts.
5.2 Role Gathering

Role gathering is focused on the data confidentiality and privileged access based on the rules enforced by known system. To determine whether a specific access mode is allowed, the privilege of a role is compared to the classification of the group to determine if the subject is authorized for a confidential access. The objects structure is defined by the access control. Role gathering occurs on the system’s level or through the client/server call. The IBOPS server stores role gathering information to associate a unique user with a unique device.

5.3 Access Control

Access control whether a given subject (person, device or program) may read, write, execute or delete a given object. The community further divides access control into Discretionary Access Control (DAC) and a more granular form of access control called Mandatory Access Control (MAC).

5.3.1 Discretionary Access Control

The IBOPS supports access control between the named users and the named objects (e.g., files and programs). The adjudication mechanism is a role based and allows users and administrators to specify and control sharing of those objects by named individuals, or defined groups of individuals, or by both. The discretionary access control mechanism provides that objects are protected from unauthorized access. Discretionary access control provides protection at the group or individual level across singular or group of objection. The granularity ranges from individual to group.

5.3.2 Mandatory Access Control

The IBOPS should enforce a mandatory access control policy over all subjects and storage objects under its control (e.g., processes, files, segments, devices). These subjects and objects are assigned sensitivity labels that are a combination of hierarchical classification levels and non-hierarchical categories, and the labels are used in the adjudication as the basis for mandatory access control decisions. The client software must maintain labels or have the IBOPS server maintain the data, which forces adherence to labeling of the subject and objects. The IBOPS server maintains a trusted store as a component of IBOPS. The following requirements hold for all accesses between subjects and objects controlled by the IBOPS: a subject can read an object only if the hierarchical classification in the subject’s security level is greater than or equal to the hierarchical classification in the object’s security level. The non-hierarchical categories in the subject’s security level include all the non-
hierarchical categories in the object’s security level. A subject can write an object only if the hierarchical classification in the subject’s security level is less than or equal to the hierarchical classification in the object’s security level and all the non-hierarchical categories in the subject’s security level are included in the non-hierarchical categories in the object’s security level. Identification and authentication data should be used by IBOPS to authenticate the user’s identity and to ensure that the security level and authorization of subjects external to the IBOPS that may be created to act on behalf of the individual user are dominated by the clearance and authorization of that user.

5.4 Audit and Assurance

The worst possible case is a system is compromised and we do not know. To prevent this case, the aforementioned specifications rightly require auditing and proof of the security model, which is called assurance.

5.4.1 Audit

The IBOPS supports all auditing requests at the Subject/Object level or at the group level. The IBOPS uses Aspect Oriented Programming (AOP) to ensure that all calls are safely written to an audit trail. A RESTful web services and JSON interface provides a mechanism to read the audit trail. Auditing may occur at the subject per action, the object per action or the group per action. For example, a group of users called “Accounting” may audit all writes to General Ledger. Or, the “Chief Financial Officer” may have audits for reads of the Income Statement.

5.4.2 System Integrity

The JUnit tests exist for all boundary conditions of the IBOPS. The suite of tests includes testing all boundary components and conditions of the system.

6. IBOPS Interoperability

The IBOPS allows the systems to meet security needs by using the API. The IBOPS need not know whether the underlying system is a Relational Database Management System (RDBMS) or a Search Engine. The IBOPS functionality offers a “point and cut” mechanism to add the appropriate security to the production systems as well as to the systems in development. The architecture is a language neutral allowing REST, JSON and Secure Socket Layers to provide the communication interface. The architecture is built on the servlet specification, open secure socket layers, Java, JSON, REST and Apache Solr. Each and every tool adheres to open standards allowing maximum interoperability.
7. IBOPS Overview

The IBOPS provides Identity Assertion, Role Gathering, Multi-Level Access Control, Assurance and Auditing. The IBOPS includes software running on a client device (Android, iPhone, etc.), a trusted IBOPS Server, and the IDS. The IBOPS allows pluggable components to replace existing components functionality accepting integration into current operating environments in a short period of time.

The client/device application loads a one-time 2-way SSL key for the initial communication to the server. This key is replaced, in function, by the subjects 2-way SSL key during the identityGenesis phase. Identity Genesis, is an initial step and fuses a set of biometrics with a given subject.

The client/server application interaction with IBOPS could be described as a three steps process with two possible variations after the first step. For the purposes of this document, the IBOPS Client/Server Application architecture is presented in three components: client application, the IBOPS security server, and the server-side application (App Server on the diagrams).

As shown below on the figures 2 through 6, the most important part of the IBOPS Client/Server Application connection is the server-side application is not running through the IBOPS server, and the SSL connection terminates at the application server. The IBOPS deployment doesn’t require the application to trust the IBOPS system with the unencrypted content.

During Genesis, the client make a call to the IBOPS server, authenticates the user, and client-side application software, then receives, allocated by the IBOPS server, a certificate, which is specific to the client identity of a specific application.
During the next step (Figure 3), the client device/app calls the app server directly, the IBOPS server is not on this path. The SSL connection between client and server parts of the application starts and terminates at these points. All content exchange is not visible outside of the application to the IBOPS server or other not trusted within this application entity.

During the client session the app server calls the IBOPS server to get identification details and confirm the certificate has not been revoked already.

In the variation flow the first step is the same: Genesis. After the first step is complete, the IBOPS server contacts the application server component to notify that a new client has been registered and allocated.

This flow (Figure 5) differs from the previously described flow (Figure 3) in a matter of the identity details, and revocation check is procured in the client session (see Figure 6).

At the third step, when the client calls the application directly, the application server calls the IBOPS server to check if the certificate has not been revoked yet.

7.1 IBOPS Application
For example, the entire IBOPS suite could be used through the access control or simply added to identify assertion of already existing framework. The IBOPS enables trusted processing by performing the minimum actions in the production environment and in the most cases does not require the change of any application software.

2-way Secure Socket Layers (SSL), which is built on a top of 1-way SSL, provides communication starting at the client. The initial or Genesis communication establishes the origin of the client’s identity and passes the IBOPS compliant 2-way certificate that the client uses for a subsequent communication in conjunction with the session oriented Identity Assertion. It is important to note that the client application must have a pre-loaded 2-way SSL key that allows subsequent identityGenesis.

The IBOPS compliant server receives 1-way SSL communication with 2-way SSL identity. Communication is conducted both 1-way SSL and 2-way SSL. The server uses a data store to take trusted identity and gather the roles for processing on behalf of the identity. Auditing ensures the appropriate artifacts for continued verification and validation of the trusted access. The Assurance occurs through the simplification and documentation of the multi-level access control mechanism. The IBOPS requires an Administration Console, which is available after the registration process (See section 7.2 IBOPS Registration), which allows dynamic modification of Users, Groups, and Roles.

IBOPS must be implemented with an active Intrusion Detection System that provides prevention of any form of brute-forcing or denial-of-service (distributed or single DOS) attacks. The standard contains a custom rule that identifies and tracks the attempts to forge 2-way SSL certificates impersonation, a session replay, forged packets, and variety of other attempts to circumvent the IBOPS server.

![Figure 7. Client/Server Application Architecture: Variation of Client session](image)

**7.2 IBOPS Registration**

The registration process initiates the IBOPS adoption within an organization. IBOPS may appear as a cluster, but is considered a business component. Before the IBOPS admin sets up an environment, the organization must register for an API key at the IBOPS Server. The individual developers may apply for the API key as well.

At enrolment completion, the `originalSiteAdmin` may create additional site administrators. In the future, the enrollment information will be associated with the API key of the organization. The API registration pertains two domains: the enrolled original site admin and the issued API key,
which is based on the enrollment information, the organization, and use case. The registration process is complete when the application commencement is agreed. After, the IBOPS admin creates originalSiteAdmin for an organization; the original site admin may create a site admin (see Figure 3). The steps after the registration are described in the section 5 of this document.

![Roles Hierarchy](image)

**Figure 8. Instance of Roles Hierarchy**

### 7.2.1 Developers and the IBOPS Service

Prior to the development process that utilizes the IBOPS service, a developer needs to be registered in the IBOPS Admin Console. By providing the application name and using Axiom to identify the developer, the IBOPS establishes a new account and creates the API key, which would be identified with the application name and associated with the application.

### 7.2.2 The End User and the IBOPS Service

The communication between the application and the IBOPS server is established on the top of 2-way SSL. Genesis is a mechanism that establishes this connection. Genesis specifies how the users identify themselves to the IBOPS server, so the server would generate a private key to set up the 2-way SSL communication between the application and IBOPS server. Axiom is one of the mechanisms that IBOPS uses to identify users.

### 7.2.4 The End User device and the IBOPS Service

The application is responsible for providing a unique ID that identifies the device of end user.
The application uses the device association API to notify the IBOPS server about the link between the user and the user’s device.

### 7.3 IBOPS Prevention of Replay

#### 7.3.1 The Format of all Requests

This example demonstrates communication between a client and a server. An important note: all calls to the IBOPS server have an API call with the notable exception of CreateApplication, which actually creates the API Key. The request is coming from a client device to:

**Example:** [https://xyz.domain.com:8443/{IBOPS_Instance_Server}/](https://xyz.domain.com:8443/{IBOPS_Instance_Server}/)

The initial call receives 2-way SSL certificate and creates a user. The user is created in clustered persistent environment. The sums that prevent playback are assumed as one-way encrypted using SHA512. Switching SHA512 with any suitable algorithm does not change the algorithm. For the duration of this document, we assume SHA512.

For val1=<n1>, n1 is a SHA512 sum of an integer between -59 and 59 added or subtracted from the current time in ISO-8601 format. The server is required to use the xntp protocol for timing. For val2=<n2>, n2 is a SHA512 sum of an integer between -59 and 59 and is greater than the plaintext value of n1. The values for username and password are client dependent and used to reach the current identity asserter. Similar mechanisms exist for SAML, LDAP, ActiveDirectory in conjunction with a variety of mechanism for Asserting Identity and Role Gathering.

The initial call would be to:

[https://xyz.domain.com:8443/{IBOPS_Instance_Server}/?val1=<n1>&val2=<n2>&siteId=<client>&username=<username>&password=<password>&newPassword=<newPassword>

Let’s examine the consequence of our Genesis Request:

<table>
<thead>
<tr>
<th>username</th>
<th>email</th>
<th>val1</th>
<th>val2</th>
<th>sessionTimeout</th>
<th>roles</th>
<th>siteId</th>
</tr>
</thead>
<tbody>
<tr>
<td>scott</td>
<td><a href="mailto:scott@scottstreit.com">scott@scottstreit.com</a></td>
<td>5</td>
<td>40</td>
<td>3600</td>
<td>Admin</td>
<td>businessCustomer</td>
</tr>
</tbody>
</table>

The user **scott** has an email **scott@scottstreit.com**. The first replay value in a plain text is 5 and the second is 40. The **sessionTimeout** exists at the **sessionld**, **siteId** pairing. For an administrator of the business customer website the **sessionTimeout** exists one hour. The session timeout is defined on a per client basis.
In the greater detail the example works as follows, with the current time as 2013-12-22T17:46:03.647Z. We move back to the five-minute interval and get 2013-12-22T17:45:00.000Z with an SHA sum of 90ab8d22f8e63e5c060fc5039a57e740739eb94cbeaeb5e6ffaaeb88a6848093ca9a22e7e8cbb7775b318c34c2c69508117c2d6fd9d119c4287f6e106ef32d and for the 40 minute interval 2013-12-22T17:40:00.000Z with an SHA sum of b86a266abc0b617c7f892f2c926dd405fee060d3062dd2115ca71a53b60fd60b27064a9ffab3b91679203032a6c0ff7d25a5f7850eb35ff2e11b6202a9b4dc

Example: https://xyz.domain.com:8443/[IBOPS_Instance_Server]/genesisval1=d7b9d423a6fed1d7ff7e3367e12165fe61755f975a296741eb0484e7cb96e2c350c08fbdcd803157f748200c071f8424ad4d116157f78f7ca8cb03ea34ab49e2&val2=b86a266abc0b617c7f892f2c926dd405fee060d3062dd2115ca71a53b60fd60b27064a9ffab3b91679203032a6c0ff7d25a5f7850eb35ff2e11b6202a9b4dc

The values associated with val1 is:
d7b9d423a6fed1d7ff7e3367e12165fe61755f975a296741eb0484e7cb96e2c350c08fbdcd803157f748200c071f8424ad4d116157f78f7ca8cb03ea34ab49e2 is a 5 offset and for val2=b86a266abc0b617c7f892f2c926dd405fee060d3062dd2115ca71a53b60fd60b27064a9ffab3b91679203032a6c0ff7d25a5f7850eb35ff2e11b6202a9b4dc which happens to be the same for 40. The newPassword is the password for the 2-way SSL key, which is never going to be stored on the IBOPS server. To execute this operation the IBOPS Server must have the SHA512 sum for all integers between -59 and 59 to decipher the sums.

7.3.2 Subsequent API calls

For instance, at 2:18pm Zulu time user scott uses a client device (Android phone) to create a session. The call contains deviceId for a session, as well as the following parameters:

val1=<SHA512sum of current time rolled back to the nearest 5 minute interval>&val2=<SHA512sum of current time rolled forward to the nearest 20 minute interval>&command=<SHA512sum of an a low level operating system such as fopen>&version=<version of command>&val3=<SHA512sum for the command file>

To prevent the replay of a previous session or a replacement the key kernel object files, the IBOPS server must contain SHA512 sums for commands names and the files on a version-by-version basis. Using the IBOPS protocol in a conjunction with the IBOPS intrusion detection system prevents the replay attack. The IDS updates the list of the blacklisted objects, as threats and attacks, on the further attack recognition level.
8. IBOPS API Overview
All API names are in the RESTful JavaScript Object Notation (JSON) format. This part of the document is a work in progress and will change as implementation progresses.

8.1 Identity Assertion API

8.1.1 Developer API_Key
Individual developers need to apply for an API_Key for their applications that will use IBOPS. This will be done in Developer Center. Once individual developers have their own API_Key, all the API calls their applications make, will need to insert this API_Key as one of parameters. The IDAP will verify the API_Key at that API level.

8.1.2 Application Identification
Application Identification creates an application for use by a development team. The definition of the Application Creation process is provided below. Once application commencement is agreed, the overall IBOPS admin creates a user with the special role of originalSiteAdmin. Once the original site admin exists, the first action of the person with the originalSiteAdmin role associates their biometrics with identity. Subsequently, all actions have genesis and API.

Additionally, the originalSiteAdmin role may create users of a siteAdmin role. The siteAdmin role is used for an additional administration work.

Some terms are required to further understand the API.

9.0 API

Account means a user account which was validated (against an external system or by email validation mechanism). It can have associate one or multiple Mobile Devices. The enrollment process ends by creating a Client Certificate for the device which will be used for subsequent calls to authenticate against the platform.

9.1 Enterprise concepts
In the system can be registered one or multiple integrations. That is named Members data. Special data registered in this chunk of data is LoginDefinition field where it is described the credentials format for a certain integration.
When an user register a profile (*MemberProfile*), basically the system authenticate the user profile / credentials against the external system and associate the external user profile with his Account.

Once the user is enrolled and has profile for a certain enterprise integration registered, (s)he can authenticate against the external system. That means, user can attach to a session which was initiated (Session Opportunity), authenticate on mobile against registered biometric data (saved during enrolment process) and send the result to the backend.

### 9.2 Format of API calls
Each call receives the input parameters in JSON format and produce result in JSON format too. Each response contains the Error information (error JSON field)

```json
ErrorResponse {
  errorDescription (string, optional),
  errorCode (integer, optional)
}
```

codeError = 0 means success. Otherwise, an error occurred and description of the error lays in errorDescription field.

### 9.3 Genesis
The start of the process is Genesis or Enrollment. This fuses an initial identity with an individual. To do this, the device uses a 1 time 2-Way Secure Socket Layers connection. The device offers to the IBOPS Server, any and all information uniquely identifying the device and this information is sent to a separate and distinct IBOPS server which only responds to Genesis requests. At this point, the IBOPS Server responds with a variety of information.
The server responds with a 2-Way SSL Key containing identity, a password for encryption and decryption, and a set of values which prevent replay. This is the standard 2-way SSL communication with the password requested by the device and the very same password used for passivated encryption. We call this key the IBOPS key. The transport layer of encryption uses 571 bit Elliptic Curve Encryption. The passivated data on the client device is encrypted also using 571 bit Elliptic Curve Encryption. This passivated encrypted data is all data stored on the device for initial or subsequent use.

As far as storage, on the client (device) is a encrypted version of the biometrics and the 2-Way SSL key which offers the possible identity on the device. The matching and liveness tell us if the identity matches the genesis identity. The device has no key generation artifacts and all information is encrypted when stored (passivated) on the device.

On the server the device information that relates to a particular device. The key generation is from the server and never stored on the server. It is encrypted with the password only known by the client and also never stored on the servers. The server maintains identity information and no keys or other artifacts are stored.

9.3.1 Post genesis communication uses

9.3.2 Is the Device Blacklisted?
The IBOPS Server asks the Intrusion Detection System (IDS) if a device is blacklisted or not. If the device is blacklisted, all communication ceases. Every failed replay prevention value1 and replay prevention value2 result in machine learning logic on the IDS.

The intrusion detection system uses machine learning through the Semantic Web in determining how broad of an area to blacklist. Blacklisting may occur at the device level, the ip address level, the subnet level or beyond. This multi-tiered approach prevents trial and error attacks and restricts bad intention attempts.

Enterprise solution offer mechanism to enroll a profile using the validation against the external system. This can be done at backend integration level. After the genesis call happens, IBOPS will respond to mobile application with client certificate which identifies unique that device. All the subsequent calls will use the client certificate to authenticate against IBOPS server.

9.4 API - Genesis (Enrollment)

The genesis service is the initial setup for a user. It may work with the already stored initial identities or use an external axiom services for the initial identity.

9.4.1 /RegisterAccount

Request {
    name (string): Name of device,
    info (string, optional): Details information of the device,
    os (string): OS of the mobile device,
`externalId` (string): Identifier of mobile device (e.g. IMEI),
`pushRegId` (string): Push Notification Registration Id from the specific provider that mobile has,
`memberExternalId` (string): Enterprise integration external identifier (e.g. 'wsj'),
`loginData` (string): JSON with external system login information. It will be used to authenticate against the external system in order to validate the account,
`deviceFingerprint` (string): Device Fingerprint(SHA512 sum),
`val1` (string),
`val2` (string)

Response {
    `clientCertificate` (string): Base64 encoded p12 client certificate,
    `clientCertificatePassword` (string): Client Certificate Password,
    `id` (string): Internal Id of a new registered user profile,
    `loginData` (string): Encrypted JSON with profile data which must be saved on mobile device,
    `credentialsData` (string): Encrypted JSON with credential filled in by user,
    `error` (ErrorResponse, optional)
}
ErrorResponse {
    `errorCode` (integer, optional),
    `errorDescription` (string, optional)
}

9.5 API - QROpportunity
One of the application authentication flow is using QR code. On the business partner login page, it will be displayed a QR Code image which will contain a Session Opportunity Identifier. Mobile Application should start an authentication wizard which will scan the QR Code, register the session to signal that it is attached to the session and authenticate with his / her biometrics. After biometric authentication, should send to Backend the result of authentication.

9.5.1 /QROpportunity

Request {
    `name` (string): Name of device,
    `info` (string, optional): Details information of the device,
    `val1` (string),
    `val2` (string)
}
Response {
qrImage (string): Base64 encoded QR Code based on a random number,
error (ErrorResponse, optional)
}
ErrorResponse {
errorCode (integer, optional),
errorDescription (string, optional)
}

Biometric authentication, should send to Backend the result of authentication.

---

9.5.2 /enterprise/RegisterSessionOpportunity

Input
Request {
memberExternalId (string): External Identifier of enterprise integration (e.g. ‘garanti’),
}

Output
Response {
sessionQrImage (string): Base64 encoded image with session informations,
sessionId (string): Session Internal Identifier,
error (ErrorResponse, optional)
}
ErrorResponse {
errorCode (integer, optional),
errorDescription (string, optional)
}
9.5.3 /enterprise/GetSessionStatus

Input

GetSessionStatusRequest {
  sessionId (string, optional)
}

Output

SessionResponse {
  status (string): Status of session,
  opportunety, created, authenticated, failed, timeout, completed, canceled
  sessionId (string): Session Internal Id,
  data (Map[string, Object]): Extra values attached to session instance,
  error (ErrorResponse, optional)
}

ErrorResponse {
  errorCode (integer, optional),
  errorDescription (string, optional)
}

9.5.4 /enterprise/RegisterSession

Input

SessionRequest {
  sessionId (string): Session Opportunity Identifier
}

Output

RegisterSessionResponse {
  error (ErrorResponse, optional)
}

ErrorResponse {
  errorCode (integer, optional),
  errorDescription (string, optional)
}
9.5.5 /enterprise/AuthenticationResponse

**Input**
Request {
  sessionId (string, optional),
  result (integer, optional)
}

**Output**
Response {
  error (ErrorResponse, optional)
}
ErrorResponse {
  errorCode (integer, optional),
  errorDescription (string, optional)
}

9.6 Role API

The Role Gathering is retrieved from an authoritative Role Source, i.e., Active Directory, LDAP or relational database Big Data server, or conducted through an additional API call on the IBOPS server to find the list of the roles. Roles are gathered and stored in the IBOPS server.

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition</th>
<th>Input Parameters</th>
<th>Output Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>loadRoleGenesis</td>
<td>Given a userId, deviceId and the systems, go to the well-defined role-gathering source and replace the roles in IBOPS. This also cancels all open sessions. All sessions must be reconstructed after this API call. The duration of each session is a security policy decision. So, how long each session lasts and how long of inactivity prior to the creation of a new session (Time To Live), the device scanning result may be sent to the IBOPS</td>
<td>userId, deviceId</td>
<td>The roles are loaded into server memory no output.</td>
</tr>
</tbody>
</table>
9.6.1 Dynamic Image Code Session Construction at a Glance

The web page for a dynamic image returns sessionld. A sub-API call returns a MIME-encoded image that has the sessionld in the dynamic image. The next API call returns a URL of the image and the sessionld in JSON text format. At the conclusion of the session construction all Roles (labels) are associated with the User.

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition</th>
<th>Input Parameters</th>
<th>Output Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>sessionConstruction</td>
<td>This API is used to start a session that will create a sessionld to identify this this session. Besides the sessionld, the API will also return a dynamic image with embedded code information.</td>
<td>siteID</td>
<td>Returns sessionld</td>
</tr>
<tr>
<td>sessionCreation</td>
<td>This API calls the backend after the Mobile Client Application (MCA) scans a QR code displayed on the embedded device.</td>
<td>siteId, sessionld</td>
<td>The server will use the input parameters to create a new SolrDocument. All the input parameters will be saved to the Solr document. Also in this solr document, a field &quot;status = CREATED&quot; will be inserted too.</td>
</tr>
<tr>
<td>sessionStatus</td>
<td>This is an API to check the current session status, which is associated with the given sessionld.</td>
<td>sessionld</td>
<td>The following result codes are returned: sessionNotReady, validationInProgress, userAuthenticated, userRejected, sessionTerminated, sessionExpired, sessionLogoff, userLogoff.</td>
</tr>
<tr>
<td>sessionData</td>
<td>An API call after QROpportunity from the embedded device to the backend.</td>
<td>sessionld, siteId</td>
<td>The server will use input parameters to retrieve the data. If the Solr document is found, then the API call returns “status = CREATED”.</td>
</tr>
<tr>
<td>sessionTermination</td>
<td>This API is called from</td>
<td>sessionld, siteId</td>
<td>The server marks</td>
</tr>
</tbody>
</table>
the embedded device to the backend after a complete transaction.  

“status=ENDED” and keeps all session data.

An input device scans the dynamic image and validates the scanned sessionId with IBOPS, which triggers the triple association of user, device, and session. The IBOPS client software validates the biometric. The biometric status is sent to the IBOPS server. The biometric data itself is never sent to the IBOPS server for the privacy concerns.

The next steps: the device scans biometric and sessionId is created. The session status sessionId returns sessionNotReady, validationInProgress, userAuthenticated, userRejected, sessionTerminated, sessionExpired, sessionLogoff, and userLogoff. Session termination brings a logoff notification. Once received, the session is closed for a future activity as defined by the sessionTermination in sessionId. All sessionId creation failures must be governed by IDS, which can take the appropriate actions to terminate a sessionId creation. This may be blocking IP addresses, blacklisting domains, blacklisting users, or other means. The blacklisting has a hierarchy of a restricting access to the system based on the complex machine learning rules.

9.7 Access Control API
Given sessionId, the data label and the access are allowed.

The set of data in JSON format (JSONArray of JSONObjects) is a securityLabel field. The security label field is matched against the roles associated with the user through the session. If the data (JSONObject) is a subset of the roles, then the data is returned. Otherwise, the partial data of JSONObject is not returned.

As the API redirects the call, the returned data becomes restricted. At the redirect API call we intercept a getJSON call.

The access control algorithm is applicable for each User at the session construction time flattened the hierarchies. So, if the user is a Manager implies that the Manager label is both a Manager and a User, then:

If Bob is a Manager, the labels for Bob are Manager, User
If a Piece of data is Manager, the hierarchy is not flattened.

For adjudication, if the data is a subset of the users roles (groups), the adjudication allows the user to see it:

No read up, no write. Bell-Lapadula model.

At a given point in time, the user works at the non-hierarchical security level. So, irrespective of the number of flattened labels, the user works at one label at the time, when it comes to writes. Thus, if Bob is working as a manager, he may only write data as a manager. If he is working as a user, then he may only write data as a user. This prevents the security policy from violation by “write down.”

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition</th>
<th>Input</th>
<th>Output</th>
<th>Notes</th>
</tr>
</thead>
</table>

26
9.8 Auditing

All steps of the Identity Assertion, Session Creation, and Adjudication have an audit capability. The capability may be set for any user, groups of users or roles across any action (read/write) on any set of data. The audit is gathered RESTfully and then stored on a IBOPS Server.

API for Audit Request

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>startAudit</td>
<td>2-way SSL for identity, optionally a group; action as read, write,</td>
<td>A group or a user, action to audit (read, write, update, delete) or</td>
<td>Not available</td>
</tr>
<tr>
<td></td>
<td></td>
<td>optionally a piece of data to</td>
<td></td>
</tr>
</tbody>
</table>

|

<table>
<thead>
<tr>
<th>adjudicateAction</th>
<th>Adjudicate an action</th>
<th>An identity from 2-way ssl SSL certificate; a comma separated set of the labels.</th>
<th>Actions allowed: read, write, update, delete.</th>
<th>If IBOPS doesn’t store data.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>addData</th>
<th>addData to the IBOPS Store. If data already exists, this data becomes a newer version</th>
<th>An identity from 2-way ssl SSL certificate; data stored in the tag, value pairs; a comma separated set of the labels for each piece of data.</th>
<th>Success or failure.</th>
<th>If IBOPS stores multi-level secure data.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>deleteData</th>
<th>removeData from the IBOPS store.</th>
<th>An identity from 2-way ssl SSL certificate; the tags to remove.</th>
<th>Success, if data is removed; Failure, if there is a security exception or insufficient privileges</th>
<th>If IBOPS stores multi-level secure data.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>readData</th>
<th>readData from the IBOPS store</th>
<th>An identity from 2-way ssl SSL certificate; name, value pairs to the read function.</th>
<th>The data is in a JSON format that the user may see based on security labels.</th>
<th>If IBOPS stores multi-level secure data.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Definition</td>
<td>Input</td>
<td>Output</td>
<td>Notes</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>---------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>update, delete, optionally a data object. If data object is not specified, then all data will be audited for the user.</td>
<td>apply the audit.</td>
<td>2-way SSL for identity, optionally a group; action as read, write, update, delete, optionally a data object. If data object is not specified, then auditing of all data for the user will be stopped.</td>
<td>stopAudit 2-way SSL for identity, a group or a user, action to audit (read, write, update, delete) or optionally a piece of data to apply the audit.</td>
<td>Not available</td>
</tr>
<tr>
<td>auditRecord</td>
<td>2-way SSL for identity, action (read, write, update, delete), source of data. This writes an audit record.</td>
<td>2-way SSL for identity, a piece of data to audit (tag, value in JSON format), and the action, which is being audited.</td>
<td>auditRecord 2-way SSL for identity, action (read, write, update, delete), source of data. This writes an audit record.</td>
<td>Not available</td>
</tr>
</tbody>
</table>

### 9.9 API for Read Audit Logs

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition</th>
<th>Input</th>
<th>Output</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>readAudit</td>
<td>2-way SSL for identity, start date in ISO8601 format; end date in ISO8601 format. If there is an administrator role, then the audit record is returned in JSON format.</td>
<td>2-way SSL for identity, to show user an audit record; datetime for start; datetime for end; data records to report (allowing &quot;wild cards&quot;).</td>
<td>The audit records in JSON format.</td>
<td>Must have an administrator privilege to perform audit.</td>
</tr>
</tbody>
</table>

### 9.10 Administration

The mapping of Users to Groups and Groups to Roles and Attributes to Groups is provided by an API call. All calls require a 2-way SSL communication layer and should be conducted by the administrator role.

**Example:** UPDATE_URI=https://xyz.domain.com:8443/{IBOPS_Instance_Name}/JSONUpdate

**Example:** UPDATE_URI=https://xyz.domain.com:8443/{IBOPS_Instance_Name}/JSONUpdate
### 9.10.1 To Add or Update a User

<table>
<thead>
<tr>
<th>Input Parameters</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>the users name</td>
</tr>
<tr>
<td>id</td>
<td>the unique identifier for a user</td>
</tr>
<tr>
<td>login</td>
<td>the user login</td>
</tr>
<tr>
<td>password</td>
<td>the password used for a role gathering source</td>
</tr>
<tr>
<td>category</td>
<td>always &quot;User&quot; the persistence engine</td>
</tr>
<tr>
<td>email</td>
<td>the primary email for the User</td>
</tr>
<tr>
<td>groups</td>
<td>a comma separated list of group ids for which the user is a member</td>
</tr>
<tr>
<td>siteId</td>
<td>the siteId (organization) of the user</td>
</tr>
</tbody>
</table>

### 9.10.2 To Add or Update a Group

<table>
<thead>
<tr>
<th>Input Parameters</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>group name</td>
</tr>
<tr>
<td>id</td>
<td>the unique id of the group</td>
</tr>
<tr>
<td>description</td>
<td>description of the group in text format with spaces allowed</td>
</tr>
<tr>
<td>category</td>
<td>always &quot;Group&quot;</td>
</tr>
<tr>
<td>attributes</td>
<td>a comma separated list of attributes that are associated with every User in the Group</td>
</tr>
<tr>
<td>roles</td>
<td>a comma separated list of roles in non-hierarchical format. All hierarchies are flattened</td>
</tr>
<tr>
<td>users</td>
<td>a comma separated list of users ids that are members of this group.</td>
</tr>
<tr>
<td>siteId</td>
<td>the siteId (organization) of the group</td>
</tr>
</tbody>
</table>
9.10.3 To Add a Role

<table>
<thead>
<tr>
<th>Input Parameters</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>role name</td>
</tr>
<tr>
<td>id</td>
<td>the unique id of the role</td>
</tr>
<tr>
<td>description</td>
<td>description of the role in text format with spaces allowed</td>
</tr>
<tr>
<td>category</td>
<td>always &quot;Role&quot;</td>
</tr>
<tr>
<td>siteId</td>
<td>the siteId (organization) of the role</td>
</tr>
</tbody>
</table>

9.11 Reporting

The administration level report is available in the auditing API.

10 Client Device Requirements

For an identity assertion, there are several requirements to the client devices. These requirements include:

1) The client devices must not perform key generation. The key generation happens during the Genesis process and is stored on the device.
2) The client device must implement the aforementioned value1 and value 2 replay prevention algorithms.
3) All communication via the client devices and the server must be via 2-Way Secure Socket Layers.
4) All passivated client device data must be encrypted. There shall be no unencrypted artifacts on the client devices.
5) All data transfer between the client device and the server must be encrypted at the transport layer.
6) The client software must blacklist itself if the user fails to authenticate more than X consecutive times. The client software must have a client intrusion detection system capable of seeing the patterns of trial and error for blacklisting itself. All applications designed for use with the IBOPS specifications, must include some form of the intrusion detection, whereby the software can detect spoofing attempts and restrict the access to the backend system. This
would be defined as X amount of tries, which then cause the client application to stop working for X period of time or indefinitely until the device can be properly assured that it is safe and valid.

7) Liveness requirement. It is required that all applications, which are intended to comply with the IBOPS specification, include some form of liveness detection or an ability to ensure that the enrolled or authenticated user is an actual person and not an image or other representation of the user. For a face recognition system, it could be blink detection, for instance. The liveness requirement should be in place for anti-spoofing prevention and blocking of the false access into the system. The choice of a liveness is up to the enterprise organization that should determine which use case fits the best its particular needs.

8) The false negatives must be below 1.2%.
9) The false positives must be below 0.05%.
10) Biometric matches must occur within 5 seconds.
11) Liveness checking must be configurable on the server and at three levels. Level 1 must take less than 8 seconds. The false positives in this mode must be below 0.05 percent. Level 2 liveness checking must take less than 12 seconds and the false positives in this mode must be.

11.0 Server Side Intrusion Detection System

11.1 API List Blacklist

For the case of a replay attack, for instance, an incident may be added to the IDS. If an incident is promoted to an attack, then the response will have blacklist as "true", otherwise, the device is valid.

/listAttacks

Input

ht
Request {
id=(string, mandatory): The actual device with the incident.
}

Output

Response {
blacklist(true|false)
error (ErrorResponse, optional)
}
ErrorResponse {
errorCode (integer, optional),
}

Kalim Sheikh 8/26/14 11:38 AM
Comment [1]: missing information?
IBOPS Guide does not have rate info for liveness
11.2 API - Incident

For the case of a replay attack, for instance, an incident may be added to the IDS. The IDS uses a variety of techniques to increase, or decrease the area of an incident find if an area, domain, deviceld is under attack.

/checkSecurity

Input

Request {
  ip=(string, optional): Source IP address of device or network gateway.
  mac=(string, optional): MAC address of device or network gateway.
  devId=(string, mandatory): The actual device with the incident.
  dom=(string, option): Domain of device or IP address. Used to find larger area for attack.
  fField=(string, option): From field or the field that was not an expected value. Could be value1 or value2 or anything else
  aVal=(string, option): Accepted value. The value received which was not correct.
  dVal=(string, option): Desired value. The value that should have been received.
}

Output

Response {
  error (ErrorResponse, optional)
}

ErrorResponse {
  errorCode (integer, optional),
  errorDescription (string, optional)
}
Glossary

Admin Console: An online portal that facilitates the registration and enrollment with IBOPS.

Application: A unique software/system, which is created using the IBOPS Application Programming Interface (API) key.

IBOPS Admin: An IBOPS administrator, who sets up an environment and creates an Original Site Admin based on the enrollment information during the registration.

IBOPS Cluster: A set of loosely or tightly connected computers, devices that communicate using IBOPS.

IBOPS Server: An instance of the server, such as in the client/server paradigm, which supports IBOPS architecture.

IBOPS IDS: An instance of the Intrusion Detection System on the private cluster that supports IBOPS architecture.

Client device IDS: An instance of the Intrusion Detection System running locally on a user device.

Jena Rules: Syntax and a system of machine learning rules for inferencing.

IDS Cluster: A set of loosely or tightly connected Intrusion Detection Systems that supports IBOPS.

Liveness: An aspect of algorithm that defines an animated object in Computer Vision.

Original Site Admin: An administrator created by the IBOPS administrator with the privilege to create other administrators within the same organization. The Original Site Administrator should be able to assert his/her unique identity according to the client requirements (See section 5.1.2(i) Genesis API/Client Requirements Note).

Site Admin: An Application administrator who is created by the Original Site Administrator.

Trusted Adjudicated Data: The data, which is stored in IBOPS with Multi-level Secure adjudication in the IBOPS server.

User: A unique user, whose identity is being asserted by IBOPS that may have several devices.

User device: A single device that has biometric-driven client software.