THE WHITEPAPER

The Identity of Things (IDoT):
Access Management (IAM) Reference Architecture for the Internet Of Things (IoT)
The Identity of Things (IDoT):
Access Management (IAM) Reference Architecture for the Internet Of Things (IoT)

Why the Identity of Things Is Important

In the great Internet of Things (IoT) gold rush, early adopters have largely left behind identity and access management. As a result, many organizations are scrambling to deal with the consequences. Successful IoT implementations have complex relationships to people, things, and services, and the only sustainable and secure method for securing the IoT long term is to enable persistent identity across applications, devices, and things.

We have reached a point where consent and control over devices and data is critical to the success of IoT, both in the consumer and industrial spaces. IoT solutions must offer a set of identity controls that properly govern who has access to what. In the identity world, these controls are the well-known concepts of authentication and authorization. Smart locks are an early example of such controls in the consumer IoT space; they let people access buildings at certain times based on clearly defined policies. In this scenario the lock must be able to authenticate users and check authorization policies to verify access.

Many IoT manufacturers are unintentionally reinventing the wheel because they do not realize that the world of identity and access management has already solved many of their problems. In fact, many problems can be addressed by applying existing identity standards and infrastructure to common IoT use cases.

Applying Identities to Devices

What attributes make up the identity of a device? Is there a common schema or data model that IoT manufacturers can leverage to make the registration, verification, and authentication process simple, repeatable, and non-proprietary?

Once a collection of attributes has been defined and collected from a device, the attributes need to be stored during the device registration process. For some devices, registration may need some sort of unique verification to confirm that the device itself is legitimate. This registration process is quite similar to how users register with an online bank. The mapping and reconciliation of an attribute from an untrusted source (e.g., via a form) to an authoritative source (e.g., an internal bank-account database) is likely.

How can a device prove the validity of the attributes associated with it? A simple approach could be to compare a value provided by the device to data held by the manufacturer of the device, such as a hard-burned unique identifier or a piece of cryptographic data stored in a Secure Element or Trusted Execution Environment.
Building Relationships
Device-to-person relationships are one simple example of the interaction between the physical and digital worlds. However, multiple other relationships need to exist to make the devices fully effective. Nearly all of the actors (devices, people, services, and data) will have many-to-many relationships and interactions.

Some relationships will simply be for temporary data access, whilst others will be persistent and long-standing, such as the person-to-device relationship or the device-to-manufacturing-plant relationship. These relationships need to be registered, verified, and then ultimately terminated.

A simple example of a relationship could be a smart lock that allows the owner permanent access but grants only limited access during specific times of day to contractors.

IoT Component Architecture
The IoT landscape contains a mixture of hardware, software, and services components that will be deployed in a variety of scenarios or phases. Several basic characteristics are common across most scenarios. Devices or things such as sensors, wearables or home automation components are likely to be low-power, low-memory, and low-storage machines. User interfaces are likely to be limited.

As can be seen above, Figure 1 describes a basic architecture from a service-interaction perspective. This scenario focuses primarily on how data flows from a set of devices to a centralized hub or cloud service.

“Little data” generated locally by the devices transforms into “big data” as it travels to a centralized store in the cloud. Data in isolation has little value, but when aggregated the data offers useful insight that can improve processes or identify common trends.

Figure 1: Basic IoT Template Architecture - Services
Figure 2 provides an expanded view that includes common scenarios at each stage of the data flow. The brokering level, for example, will require things such as device registration, authentication, and revocation. Cloud service or data storage platforms will require the ability to share data, probably via API’s, which would also require registration and authentication/authorization services.
Figure 3 outlines key technical protocols and standards that may be leveraged throughout the flow. Each scenario will have locally specific equivalent protocols, especially when it comes to the Supervisory Control And Data Acquisition (SCADA) infrastructure.
IoT Implementations - Open Standards and Platforms

IoT implementations will benefit from open and modular platforms that allow for rapid integration, enhanced interoperability, and the ability to create complex, smart service mashups in a simple, repeatable manner.

**Communications**
Communication methods must be able to support devices that require low power and little processing. Not all devices will have full TCP/IP stack features. Examples of communications protocols supported by devices include Bluetooth, CoAP, 6LoWPAN, and MQTT (detailed in the Appendix).

**Data Handling**
Data needs to be moved between devices, brokers, and cloud services in a repeatable and standardized way, allowing for simple exchange and management of data. Examples of this are JSON, EXI, and REST (detailed in the Appendix).

**Authentication, Authorization, and Privacy**
Authentication, authorization, and privacy components can be applied at every stage of the IoT data flow. When it comes to aggregated data, privacy concerns are more important than consent and control. Example of protocols include OAuth2, OpenID Connect, UMA, ACE, and FIDO. (detailed in the Appendix).

**Provisioning**
Provisioning - or the creation and management of persona and device attributes - will occur at both the device bootstrap and initialization stages and the persona registration stage. There are many already-developed standards in this field, including LWM2M, OpenICF, and SCIM (detailed in the Appendix).

**Cryptography**
Cryptography applies at all stages of the data flow, from basic HTTP/SSL-style communication to the communications methods employed at the device level. Encryption of data objects and of data at rest is also a priority, but it should be no different from that used in a non-IoT environment. Examples include DTLS, JWT, and JOSE (detailed in the Appendix).

**Scripting and Extensibility**
A key function within an IoT platform is the ability to develop extensions or manage metadata and logic. Using widely adopted languages such as JavaScript, Lua, and Groovy allows for rapid adoption, stronger interoperability, and the opportunity for more sophisticated mashups.
IoT Component Architecture Combined with Identity

The IoT component architecture is a complex, evolving landscape and has a strong focus on interoperability and data management. Traditional Identity and Access Management (IAM) services were built for a company’s internal use to assist with manual on- and off-boarding and to establish access privileges to company data and systems behind the firewall. Today, a company must implement a dynamic Identity solution that is capable of serving and connecting employees as well as customers, partners, and devices, regardless of location, to empower the digital transformation. There are several identity-integration touch-points where registration, authentication, and authorization services are available. In short, an identity solution should be capable of the following:

**Identity Registration** – This is the simple use-case of a persona providing attribute values to a persistent store. The basic self-service form, or perhaps a social login, can be used here; a web service ultimately gets to know about a ‘real’ person. Identity registration is more commonly used in the consumer IoT market.

**Device Registration** – Each of these ‘smart’ devices can be networked and is likely to contain a manufactured globally unique identifies (GUID) and perhaps some data publishing or subscribing capabilities. So how does the device register? And to what does it register? The device needs a network and the ability to send a JSON object or some other representation of itself to a service that can capture such data. Perhaps the device identity data will need to be verified or reconciled to confirm that the thing attempting registration is actually a real thing and not a fake.

**Device-to-Identity Linking** – There are now a minimum of two identities; one is a set of attributes that map to a person, the other is a set of a device’s attributes. Certain attributes in each object may need to be verified to a certain degree of assurance; then relationships between the identities can be built. For example, the device owner wants to claim ownership of the device via a linking process, entering a code or the GUID or in some other way proving possession of the thing and that the thing is real.

**Device Authentication** – When the device starts working, for example when it captures and sends data on the user’s behalf, it needs to authenticate to a service to prove that it is real and has authority to act for the user. The device needs to authenticate against an identity; the use of shared secrets and passwords is highly ill advised. The recommended approach is cryptography - perhaps JWT tokens or PKI in some shape - that employs large numbers and requires limited human involvement and limited computational power.

**Device Data-Sharing** – Picture a device that is authenticated to a service or API and perhaps is also either acting either on the user’s behalf to a third-party service or is at least capturing data that can be shared with a third-party service. What is the best method of sharing that data effectively in a transparent and simple way? OAuth2 and the more recent UMA can help here. They enable third parties that can’t be controlled directly to gain access to the data that the device has captured or perhaps even to a level of access to the device itself. This three-way interaction requires that simple registrations and authorization decisions be made in a way that both humans and devices can understand, easily revoke, and sustain.

**Multifaceted Relationships** – The more relationships the individual has with things, the more chance there will be relationships between those things, and certainly there will be many-to-many relationships between your device and other people’s devices. How can those relationships be handled? Two angles must be catered to - persistent relationship storage (the personal identity contains sub-attributes, i.e., device objects) and temporary storage of relationships with data consumers.
Device De-registration – The consumer and enterprise worlds both need to remove a device from the IoT landscape when, for example, the device changes ownership. A de-registration or de-provisioning process is needed to remove any associations, credentials, certificate materials, configuration, and relationship data. That de-provisioning process may be scheduled, be based on a trigger event, or be done via manual disassociation.

Permission Revocation – An entire device may be de-provisioned or only certain attributes or permissions associated with the device or the data that the device has captured. The second approach is likely to be employed in federated authorization landscapes that have multiple temporary relationships between data owners, data consumers, and data custodians. Such relationships could be accomplished via the use of token revocation infrastructures such as OAuth2 or through things such as the expire at (exp) attribute within a JWT claim.

Logging and Event Analytics – All of the device interactions, from registration and association to a broker or person, through to data capturing and publication, must be logged and analyzed for things like fraudulent or malicious use. Devices themselves may not be able to log or generate log data, but central hubs, registration services, and authentication and authorization services do. A mechanism is needed to identify unique devices and their associated transactions, and perhaps to correlate them to other identity-related actions and activities. This would provide a complete 360-degree view of the transaction landscape.

ForgeRock and Identity

ForgeRock, the fastest-growing open-source identity vendor in the world, is building secure, customer-facing relationships across any app, device, or thing. ForgeRock is focused on using online identities to grow revenue, extend reach, and launch new business models. Its Open Identity Stack secures over half a billion identities and powers solutions for many of the world’s largest companies and government organizations. ForgeRock’s Identity Relationship Management (IRM) platform is a flexible, reliable, and scalable alternative to traditional, proprietary identity and access management (IAM) platforms. It is the only open-source, unified, and modular platform in the market that can deliver Internet-scale identity. Designed for ease of access and unparalleled security, the IRM platform also features a small footprint, easy integration, lightning-fast deployment times, and a single REST API across the platform.
Figure 4: IoT Component Architecture with Identity Touch Points

Data Management

Device Registration
Pub / Sub

Bootstrapping

APIs
REST / JSON

USER REGISTRATION

OAUTH2 / OIDC

LWM2M / COAP / CREDENTIALS / ATTRIBUTES / CERTS / JWTs

LOCAL BROKERS

OWNER REGISTRATION
DEVICE REGISTRATION

DEVICE DATA / PUBLICATION
AUTHN/AUTHZ

DATA POLICY CHECKING

Figure 4: IoT Component Architecture with Identity Touch Points

DATA MANAGEMENT

USERS / DEVICES / CERTIFICATES
RELATIONSHIPS / CREDENTIALS / PERMISSIONS

CLOUD SERVICES

LOCAL BROKERS

OAUTH2 / JWT / CERT AUTHN / SCOPE /
TOKEN VALIDITY

LWM2M / COAP / CREDENTIALS / ATTRIBUTES / CERTS / JWTs
OpenDJ – OpenDJ is the only 100-percent open-source directory that combines the security of a proven directory with the accessibility of a database. Lightweight and easy to embed, it allows users to easily share real-time identity data across enterprise, cloud, social, and mobile environments - a practical necessity for managing today’s identity challenges. With OpenDJ, users no longer need to be an LDAP expert; they can choose either LDAP or REST to access identity data using a single solution that can replicate data across on- and off-premises applications.

OpenAM – OpenAM has a highly scalable, modular, easy-to-deploy architecture that includes the following capabilities:

- Authentication based on dynamic, context-based access is responsive to user location, time zone, device, IP address, time of day, and more, providing endless personalization possibilities and mitigation of risk.
- Authorization and Entitlement Management enable users to access applications and services based on permissions and policies defined by the business, using our powerful policy-engine tools.
- Federation and Single Sign-On with a single identity let users access services that span the cloud and mobile devices, on-premises and off, eliminating the need for multiple passwords and user profiles as well as the complexity that creates friction and slows adoption.
- Social Sign-On supports integration with “sign-up and login with Facebook”-style access to eliminate the need for user registration and allow rapid consumer adoption.
- Adaptive Risk combines contextual information to evaluate the risk of users attempting to access resources; users deemed suspicious require a higher level of authentication or identity proofing before access is granted.
- Mobile Support uses lightweight standards like OAuth 2.0 to enable a very simple REST API framework for building native and web apps for iOS, Android, and more.

OpenIDM – OpenIDM is the only 100-percent open-source provisioning solution purpose-built for Internet scale. This solution includes role-based provisioning, high-availability, “out-of-the-box,” workflow synchronization (with delivery guarantees), support for SaaS connectors, customizable user interfaces requiring no programming, and password management.

A simple REST API-based interface makes it easy to provision services across enterprise, cloud, social, mobile, and legacy environments. All services are designed as standalone modular resources.

OpenIG – OpenIG provides a simple standards-based approach to extend identity to any web application and application programming interface (API) that utilizes HTTP. OpenIG is a perfect complement to existing Web Access Management (WAM) systems or can be used as a standalone gateway; it provides a flexible policy enforcement point to support your current environment while you migrate toward a modern, standards-based platform.

OpenIG can intercept any HTTP traffic and statefully transform messages, seamlessly check for authorization and authentication, and deliver the password capture and replay that is typically used to enable integration with legacy applications that employ the same user identity, such as Microsoft Outlook Web Access (OWA) or SharePoint.

OpenIG supports standards that enable integration with business partners and across complex organizations, whether on-premises, off-premises, in the cloud, or on mobile devices using standards such as SAML 2.0, OAuth 2.0, and OpenID Connect.
The Future

Today’s current progression and requirements for IoT are generally focused on interoperability and device connectivity. Ultimately, it seems the end result of increased connectivity and data collection is yet more data. The concept of big data is not new, and there are multiple complementary industries where the storing, correlating, querying, and analyzing of big data have matured.

Whether the IoT solution is focused on customer service and improved experiences, healthcare monitoring, energy efficiency within a “smart city,” or helping to develop autonomous machines within the Industry 4.0 landscape, data is what drives new revenue and business development opportunities.

That data will allow smart, chainable services that use data output from one process as the data source for other services to flourish. Such services are akin to the many Twitter client-analytics tools used today. That basic approach, building on the principles of the Unix pipeline, will provide new business opportunities ranging from service improvement to more predictive and self-managing objects.

Conclusion

The drive for the increased “online-ification” of previously dumb devices, and the improved data collection and analytics capabilities associated with existing devices, will undoubtedly continue.

The resulting data - from a generator, owner, custodian, and sharing perspective - will require a high level of internal and external security controls in which identity relationship management, including registration, linking, authentication, authorization, and context, will have a key role to play.

How those identity services are applied depends, of course, on the underlying device architecture and the native protocols and standards being used, but modularity, scalability, and extensibility will play a significant role.

About ForgeRock

ForgeRock™, the fastest growing identity management vendor in the world, is building secure relationships across the modern Web. Focused on using online identities to grow revenue, extend reach, and launch new business models, ForgeRock’s Open Identity Stack powers solutions for many of the world’s largest companies and government organizations. Founded in 2010, ForgeRock has a leadership team that has 80 combined years of experience in the software industry and includes open-source icons and innovators. Investors include three of the leading global venture capital firms – Accel Partners, Foundation Capital and Meritech Capital. For more information and free downloads, visit http://www.forgerock.com or follow ForgeRock on Twitter at http://www.twitter.com/forgerock.
Appendix

Communications Protocols

Bluetooth Low Energy (BLE) - BLE is a “smart” version of the Bluetooth standard especially for ultra power efficiency and intricate application deployment [1].

Constrained Application Protocol (CoAP) - CoAP is a specialized web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks. The nodes often have 8-bit microcontrollers with small amounts of ROM and RAM, while constrained networks such as 6LoWPAN often have high packet-error rates and a typical throughput of tens of kbit/s. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation [2].

IP6 Over Low Power Personal Area Networks (6LoWPAN) - Well-established fields such as control networks, and burgeoning ones such as sensor (or transducer) networks, are increasingly being based on wireless technologies. Most (but certainly not all) of the nodes these networks employ are among the most constrained that have ever been networked wirelessly. Extreme low power (so low that nodes will potentially run for years on batteries), extreme low cost (total device cost in dollars is in the single digits), and Moore's law will continuously reduce that cost. All are seen as essential enablers towards the nodes' deployment in networks where there are numerous low-power, low-memory devices [3].

Message Queue Telemetry Transport (MQTT) - MQTT is a publish-subscribe-based lightweight messaging protocol for use on top of the TCP/IP protocol. It is designed for connections with remote locations where a small code footprint is required and/or network bandwidth is limited. The Publish-Subscribe messaging pattern requires a message broker, which is responsible for distributing messages to interested clients based on the topic of a message [4].

Data Handling

JavaScript Object Notation (JSON) - JSON is a lightweight data-interchange format that is easy for humans to read and write and for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language [5].

Efficient XML Interchange (EXI) - The development of the EXI format was guided by five design principles; it had to be general, minimal, efficient, flexible, and interoperable. The format satisfies these prerequisites, achieving generality, efficiency, and flexibility while at the same time keeping complexity in check [6].

Representational State Transfer (REST) - REST is an abstraction of the architecture of the World Wide Web; more precisely, REST is an architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements [7].
Authentication, Authorization, and Privacy Standards

OAuth2 – OAuth 2.0 is the next evolution of the OAuth protocol, which was originally created in late 2006. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living-room devices. This specification is being developed within the IETF OAuth WG and is based on the OAuth WRAP proposal [8].

OpenID Connect (OIDC) – OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol. It allows clients to verify the identity of the end-user based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the end-user in an interoperable and REST-like manner.

OpenID Connect allows clients of all types, including Web-based, mobile, and JavaScript clients, to request and receive information about authenticated sessions and end-users. The specification suite is extensible, allowing participants to use optional features such as encryption of identity data, discovery of OpenID Providers, and session management when it makes sense for them [9].

User Managed Access (UMA) – UMA is a profile of OAuth 2.0 that defines how resource owners can control protected-resource access by clients operated by arbitrary requesting parties. The resources may reside on any number of resource servers; a centralized authorization server governs access based on resource-owner policy. This revision of the specification is part of the UMA “candidate V1.0” process [10].

Access In Constrained Environments (ACE) – The IETF has recently developed protocols for use in constrained environments, where network nodes are limited in CPU, memory, and power. REST architecture is widely used for such constrained environments. Internet protocols can be applied to these constrained environments and often only require minor tweaking and profiling. In other cases, new protocols have been defined to address the specific requirements of constrained environments. An example of such a protocol is the Constrained Application Protocol (CoAP).

As in other environments, authentication and authorization questions arise in constrained environments. For example, a door lock has to authorize a person who is seeking access using a digital key. Where is the authorization policy stored? How does the digital key communicate with the lock? Does the lock interact with an authorization server to obtain authorization information? How can access be temporarily granted to other persons? How can access be revoked? Such questions have been answered by existing protocols for use cases outside constrained environments; however, in constrained environments, additional and different requirements pose challenges for the use of various security protocols. In particular, the need arises for a dynamic and fine-grained access control mechanism where clients and/or resource servers are constrained [19].

Fast Identity Online (FIDO) – The FIDO Alliance is a 501(c)6 non-profit organization nominally formed in July 2012 to address the lack of interoperability among strong authentication devices as well as the problems users face with creating and remembering multiple user names and passwords. The FIDO Alliance plans to change the nature of authentication by developing specifications that define an open, scalable, interoperable set of mechanisms that supplant reliance on passwords to securely authenticate users of online services. This new standard for security devices and browser plugins will allow any website or cloud application to interface with a broad variety of existing and future FIDO-enabled devices that the user has for online security [18].
Provisioning

**Lightweight Machine-to-Machine (LWM2M)** – The motivation of LWM2M is to develop a fast deployable client-server specification that provides machine to machine service. LWM2M is principally a device management protocol, but it should be extended to meet the requirements of applications and to transfer service and application data. LWM2M implements the interface between M2M device and M2M Server, which gives the Service Provider the choice to provide service to M2M users [11].

**Open Identity Connector Framework (OpenICF)** – An ICF connector allows provisioning software such as OpenIDM to manage identities maintained by a specific identity provider. ICF connectors provide a consistent layer between target resources and applications and expose a set of programming functions for the full lifecycle of an identity [12].

**System for Cross-Domain Identity Management (SCIM)** – The SCIM specification makes managing user identities in cloud-based applications and services easier. The specification suite seeks to build upon experience with existing schemas and deployments, placing specific emphasis on simplicity of development and integration, while applying existing authentication, authorization, and privacy models. The suite’s intent is to reduce the cost and complexity of user-management operations by providing a common user schema and extension model, as well as by binding documents to provide patterns for exchanging this schema using standard protocols. In essence, SCIM makes it fast, cheap, and easy to move users into, out of, and around the cloud [20].

Cryptography

Cryptography will apply at all stages of the data flow, from basic HTTP/SSL-style communication within the cloud to the communication methods at the device level. Encryption of data objects and data at rest is also a priority, but should be no different from a non-IoT environment.

**Datagram Transport Layer Security (DTLS)** – The DTLS protocol provides communications privacy for datagram protocols. The protocol allows client/server applications to communicate in a way that prevents eavesdropping, tampering, or message forgery. Based on the Transport Layer Security (TLS) protocol, DTLS provides equivalent security guarantees [13].

**JavaScript Web Token (JWT)** – JWT is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or MACed and/or encrypted [14].

**JavaScript Object Signing & Encryption (JOSE)** – JOSE is a text format for the serialization of structured data described in RFC 4627 that is often used for serializing and transmitting structured data over a network connection. With the increased usage of protocols in the IETF and elsewhere, there is now a desire to offer security services that use encryption, digital signatures, and message authentication code (MAC) algorithms that carry their data in JOSE format [15].
Scripting and Extensibility

Key to an IoT platform is the ability to develop extensions or manage metadata and logic. Leveraging non-proprietary and widely adopted languages that require limited deployment effort or no compilation allows for rapid adoption and simpler interoperability and mashups.

**JavaScript (JS)** – JS is a dynamic computer programming language most commonly used as part of web browsers. Implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. JS is also used in server-side network programming with frameworks such as Node.js as well as in game development and the creation of desktop and mobile applications [16].

**Lua** – Lua is a powerful, fast, lightweight, embeddable scripting language. Lua combines simple procedural syntax with powerful data description constructs based on associative arrays and extensible semantics. Lua is dynamically typed, runs by interpreting bytecode for a register-based virtual machine, and has automatic memory management with incremental garbage collection, making it ideal for configuration, scripting, and rapid prototyping [17].

**Groovy** – Groovy is an agile and dynamic language for the Java Virtual Machine that builds upon the strengths of Java but has additional power features inspired by languages such as Python, Ruby, and Smalltalk. This language makes modern programming features available to Java developers; it has an almost-zero learning curve and provides the ability to statically type check and statically compile code for robustness and performance. Groovy supports Domain-Specific Languages and other compact syntax, making code easy to read and maintain; it also makes writing shell and build scripts easy with its powerful processing primitives, object-oriented (OO) abilities, and an Ant DSL. In addition, the language seamlessly integrates with all existing Java classes and libraries and compiles straight to Java bytecode, so users can employ it anywhere they can use Java [21].
References

[1] - Bluetooth Smart
https://www.bluetooth.org/en-us/marketing/bluetooth-smart-technology


[3] - 6LowPAN IETF Working Group
https://datatracker.ietf.org/wg/6lowpan/charter/

[4] - MQTT
http://mqtt.org/

[5] - JSON
http://www.json.org/

[6] - EXI
http://www.w3.org/XML/EXI/

[7] - REST
http://en.wikipedia.org/wiki/Representational_state-transfer

[8] - OAuth2
http://oauth.net/2/

[9] - OpenID Connect
http://openid.net/connect/

[10] - User Managed Access
https://docs.kantarainitiative.org/uma/draft-uma-core.html

http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases/oma-lightweightm2m-v1-0

[12] - Open Identity Connector Framework
http://openicf.forgerock.org/


[14] - JavaScript Web Token

https://datatracker.ietf.org/wg/jose/charter/

[16] - JavaScript

[17] - Lua
http://www.lua.org/about.html

[18] - Fast Identity On Line
https://fidoalliance.org/about

[19] - Access in Constrained Environments
https://datatracker.ietf.org/wg/ace/charter/

[20] - SCIM
http://www.simplecloud.info/

[21] - Groovy Programming Language
http://groovy.codehaus.org/

For additional information, please contact your sales representative or go to Forgerock.com/services.