IoT Agent Design Principles

Technical Briefing

A technical briefing offered by Ardexa on the design principles of the Ardexa IoT agent

Authors

George Cora – CEO and Director of Ardexa Pty. Ltd.

David Mohr – CTO and Director of Ardexa Pty. Ltd.

Property of Ardexa Pty. Ltd.

Released 11th April 2016

Summary

Based on our IoT Design Principles, Ardexa have developed a software agent that functions on small form, single board computers. These computers act as gateways for sensors, actuators and applications. The gateway, through the Ardexa agent, enables a range of functionality from scheduling processes, handling output of varying formats, including metadata, allowing full remote control and allowing bi-directional file transfers. This means there is no need for libraries of SDKs to handle IoT functionality. Developers are then freed up to develop solutions without having to worry about handling communications, security, metadata, caching, etc, and can ***focus on what's important***.

This guide discusses the agent design criteria, and why Ardexa has arrived at key design criteria for the Ardexa IoT agent. The 10 key features of the agent are:

- ***Ardexa Design Feature 1***: The Ardexa IoT agent initiates all connections to the cloud. It does not accept any incoming connections. The connection established to the cloud includes a "back channel" that allows full remote control access to the device. Developers are therefore free to completely firewall their devices, and not allow any inbound services to the Thing.
- ***Ardexa Design Feature 2***: The Ardexa IoT agent uses digital certificates to identify and approve access to the cloud. These certificates are not shared across Things, and so can be revoked or 'quarantined' if need be. The user need not be aware of the certificates, as they are installed and managed automatically during the agent installation.
- ***Ardexa Design Feature 3***: The Ardexa IoT agent includes metadata in each stream sent by the device. This method carries almost no extra overhead if implemented well, and the cloud service will know that exact data format without the need for synchronizing. If need be, the cloud service can query the metadata for all the agents.
- ***Ardexa Design Feature 4***: The Ardexa IoT agent is able to collect log entry from many sources simultaneously, and apply metadata to fields in the stream.
- ***Ardexa Design Feature 5***: The Ardexa IoT agent is able to schedule many processes to be run at regular intervals. The output from these processes will then be tagged with metadata and sent to the cloud, in whatever format required by the IoT developer.
- ***Ardexa Design Feature 6***: The Ardexa IoT agent is able to monitor changes to any number of listed files in real time. A date/time method is used for detection, or the more secure cryptographic hash. If required, the file itself is sent to the cloud, or simply a notification of the change.
- ***Ardexa Design Feature 7***: The Ardexa IoT agent sends data to the cloud in real time. In event of connection loss, events will be cached, ready for when the connection is re-established.
- ***Ardexa Design Feature 8***: The Ardexa IoT agent is able to be fully remote controlled from the cloud, in real time.

- ***Ardexa Design Feature 9***: The Ardexa IoT agent enables real time file transfers in either direction, and in real time.
- ***Ardexa Design Feature 10***: The Ardexa IoT agent enables the option of allowing a regular heartbeat message to the cloud.

Ardexa IoT Mission

Ardexa has a corporate mission to be a highly trusted and undisputed leader in Internet of Things (IoT) communications. At Ardexa we feel that solutions are available today to deal with security, integrity and reliability issues confronting a massive explosion of connected devices. The flaws in most network designs today arise from a lack of vision, "compartment thinking", poor knowledge and applications of security principles.

We believe that a significant number of IoT implementations today rely on software 'System Development Kits (SDKs)', 'frameworks' or 'libraries'. The success of this approach is highly dependent on trained architects and programmers who know what they are doing and have the time to dedicate in developing a secure, functional IoT solution. In almost all instances, this is not the case.

To alleviate this problem, from time to time, we will provide resources, such as open-source software and whitepapers. We believe that this community approach to many solutions will speed up the development and reputation of our industry.

At Ardexa, we allow developers to focus on what's important.

For more information, please contact us on iot@ardexa.com.

IoT Design Principles

Ardexa has produced a whitepaper titled "IoT Design Principles". In it we discuss and define a number of key principles which form the foundation of our products. The principles are discussed in the whitepaper, but are replicated here as a summary, and as the basis for discussion of the Ardexa IoT agent.

- **Principle 1**: Things require software to manage sensors and handle asynchronous communications.
- **Principle 2**: Things must always initiate connections to the Internet and must not accept inbound connections.
- **Principle 3**: Each and every Thing must be identified and authenticated, in a secure, reliable and scalable manner.
- **Principle 4**: Things must provide the ability to be controlled remotely, including the ability to transfer files.
- **Principle 5**: Metadata and flexible data formats are fundamental to growing and scaling IoT systems.
- **Principle 6**: Cloud services provide cost-effective scalability and always on reliability for

IoT services.

- **Principle 7**: Cloud services must integrate seamlessly with existing IT systems.

Definitions

Definitions of the terminology used in this whitepaper are as follows:

1. ***Gateway/Device***: A small form, single board computer similar to a Raspberry Pi. These are mostly ARM or x86 based processors, with on-board storage, RAM and peripherals. They are low cost and low power, yet run various forms of Linux or Windows.
2. ***Sensor/Actuators***: A simple or complex electronic component designed to carry out a well defined task. Actuators or sensors may contain a micro controller, and some may be controlled by a gateway. A sensor usually simply collects information, whereas an actuator can control equipment such as opening/closing valves, relays, lights or related equipment.
3. ***Thing***: A Thing is defined as the software and hardware that encompasses both the gateway and sensor.
4. ***Metadata***: Metadata is "data about data". It may be a stream of data that describes elements such as the source of the data, the units used in the data stream (eg; Celsius, Fahrenheit, Newtons, etc), the author or device producing the data and perhaps the structure of the data (eg; data separated by commas, tabs, etc).
5. ***Data format***: Some data may be a single integer describing something like a temperature reading. Another piece of data may be 1 or more strings. Other again may be values separated by commas. Data formats can vary widely depending on who or what has produced the data.
6. ***Agent***: The software produced by Ardexa that handles secure communications, caching, metadata, task scheduling, remote control, file transfers, and more. This software works on the gateway, and allows sensors and actuators to communicate to the cloud.

The following sections describe the the key components of the Ardexa IoT agent, based on the principles outlined above.

SECURITY

In the 1990's, security of computer systems, both large and small, was in the hands of a very narrow team with non-transparent processes. No-one is in doubt that security, in this stage of Internet development, is a key component of any interconnected system. Yet, the skills, specialization and resources required to effectively manage and maintain a satisfactory level of security continues to increase. Tools and methods to manage risks are becoming cheaper and more effective, but the range of threats and their likelihoods continue to increase with new devices, applications and processes such as mobile devices, cloud services and now IoT devices.

The three security issues that must be addressed in an IoT infrastructure are firewalls, encryption and authentication. Without a sound handle on these 3 issues, then it is very likely that at some point in the future the IoT infrastructure will be compromised.

Firewalls and Services

Firewalls[1] manage and protect a secure, internal or corporate network from being compromised, so long as they are configured correctly. A firewall will block all TCP, UDP and ICMP communications whilst also undertaking a range of proxy and security tasks, and yet allowing requests from the internal network to the outside (Internet). All modern firewalls can allow certain types of requests (which use a specific port) from the Internet, to the internal network. This is the idea of providing a service to the public via the Internet.

All cloud services allow certain ports (services) to be accessed by unauthenticated users. Access to a web server, email server, login page for some cloud based service all require access through a firewall to a specific port. But these services are mostly managed by very competent security administrators, where the application is written and tested for security vulnerabilities and where software at all levels is constantly updated and patched.

Handling server and/or service (application) security is not a one-off task, but a continuous process to ensure vulnerabilities are mitigated. For example, the 'Apache' web server is a commonly used, open source application used in many millions of websites. Querying the list of vulnerabilities for this web server using cve.com[2] (a security site that details vulnerabilities and exposures for software) shows that there are almost 900 security vulnerabilities and exposures that mention 'Apache'. Any single one of these could cause a significant compromise of an internal network, if left unchecked.

So the problem for a Thing (IoT hardware and software) is that; if the software is not kept up to date, which in most cases it probably won't, or if there is a misconfiguration (which there is likely to be), then the Thing becomes vulnerable. If the Thing allows access from the public Internet through a service, and is within an internal network, it will then become the 'launch point' for a more extensive compromise of an internal or secure network. This is how most attacks begin.. by compromising a weak or insecure system (be it a computer, application, process, or human) within an organization.

And this is why almost all network administrators will not agree to 'punch' holes from an the public Internet, across a firewall, into a secure network. Expecting requests to come in from the Internet to a Thing is both unrealistic, and more importantly, very very risky.

Furthermore...Anytime a machine connects to a network, it does so using an IP address. These addresses can be assigned either statically, so that a machine has the same address every time it boots up, or dynamically, so that machines may come and go and share a pool of addresses. Static addresses are not always an option when connecting to the Internet and managing dynamic addresses by having the gateway update a registry somewhere requires almost as much work as setting up a secure tunnel.

A secure tunnel is configured by the Thing after it has connected to your Internet service. Therefore, when addressing your Thing, you need only know its identity, which is either conveniently mapped to an established connection if your Thing is online, or requests are queued ready for the next time the Thing logs in.

Therefore, the IoT agent **must** originate connections to the cloud. In doing so, a back channel from the Internet to the Thing can be established without the need for a hole to be 'punched'

though the firewall. This back channel is configured by the agent, so the agent can shut it down, proxy requests, authenticate users and processes, etc. It is not a separate connection, but one established by the agent. This is very important in maintaining security.

```
Ardexa Design Feature 1: The Ardexa IoT agent initiates all connections
to the cloud. It does not accept any incoming connections. The connection
established to the cloud includes a "back channel" that allows full
remote control access to the device. Developers are therefore free to
completely firewall their devices, and not allow any inbound services to
the Thing.
```

Authentication and Encryption

Why is there a need authenticate a Thing? In both remote management (cloud controlling the agent) and data forwarding (agent sending data to the cloud), it is necessary that the credentials of the Thing and the cloud be established before a communications channel is opened to, or from, the Thing. In both these cases, one entity must approve the identity of the other. Authentication is about establishing that a Thing is known and authorized to access a resource.

UserID and password combinations are the most commonly used authentication mechanism. This mechanism is useful for humans logging into a network. For Things however, there are serious limitations with this approach. A userID/password can be easily compromised, and can be very difficult to determine when or how it was compromised. A Thing is not likely to alert a security administrator of a compromised account, and changing a password on a Thing is not as easy as contacting a human on a telephone.

Also, a userid is usually associated with a person, and hence establishing the identity of a person is an intuitive task. But not so for a Thing. How does an administrator know which actual hardware device is using the userid, say, "device23"? How would an administrator know if this device is hijacked, and replaced with another device that has been compromised? Worse, what if the same password was used to authenticate (say) 20 devices? In this case, it is almost impossible to determine which particular Thing is authenticating to the cloud.

The best option for providing authentication and encryption is using Transport Layer Security (TLS) with digital certificates. TLS is the successor to SSL and provides an open source framework from which to provide security services at the communications layer. Digital certificates[3] enable:

- Positive unique identification per device during the connection phase. This allows positive identification per Thing ***before*** the device is granted access to the cloud,
- An agreed, minimum set of encryption algorithms to be used for encryption, and
- The ability to revoke the authority for a specific Thing, in event of compromise, loss or theft.

Digital certificates allow a device to be independently tagged as "compromised", without affecting the status of all other devices. There is some overhead in generating certificates, as

administrators need to have the tools and processes to effectively manage a public key infrastructure[4]. Using digital certificates can mandate the level of encryption to be used on the connections opened up by the Thing. For high security installations, a stronger encryption algorithm may be required, whereas standard grade encryption would be suitable for most installations.

```
Ardexa Design Feature 2: The Ardexa IoT agent uses digital certificates
to identify and approve access to the cloud. These certificates are not
shared across Things, and so can be revoked or 'quarantined' if need be.
The user need not be aware of the certificates, as they are installed and
managed automatically during the agent installation.
```

DATA MANAGEMENT

A big part of IoT functionality is to collect data, sometimes lots of it. The data may or may not be:

- compressed
- sent in real time
- in a variety of formats
- routed to internal and/or external analytics engines

As cloud services become cheaper, faster and more accessible, collecting large volumes of any form of data is becoming less of an issue. In fact, most situations requiring an IoT agent should be encouraged to collect as much data as possible, within reason. Cloud services these days can process and analyze billions of events in a matter of seconds, and/or store gigabytes of data at reasonable prices. The developer's trend is clearly to collect and store larger portions of available data in the pursuit of greater value.

Data Formats and Meta-Data

Data collected from a Thing may present itself in any number of formats. It may arrive as a single integer, a series of floating point numbers, a string, a photograph, a video stream or any of these combinations or more. For instance, data from a residential device could contain a temperature reading, a series of decimal numbers indicating the state of an alarm system, an image showing the latest face at the front door, and a file showing the current configuration and/or power output of a solar inverter.

Each of these data elements are very different. It is neither desirable nor possible to somehow develop a 'common format' that defines all possible types of data. To do would be to artificially limit the developer.

To handling these varying data formats, each data stream must come with its own set of metadata[5]. Metadata describes the data, and is used by the recipient to index the data for storage and analysis. Without metadata, the sender and recipient must be very clear about the exact format of the data. For example, the line below shows an event generated by a Thing:

20130513T11:22:33Z,12.3,77,offline,variable,23,45.23

What does this mean? There is no way anyone person or computer can understand the complete line without meta-data describing each comma separated element. And when a new element is added, say to describe a new pressure reading, the metadata must be updated to describe this new reading. If this metadata is not included as part of the Thing, then any changes to the format must be manually synchronized with the cloud service that collects, stores and analyzes the data. Without this synchronization, data could be lost, corrupted or misinterpreted. So what happens when the synchronization is lost. It leads to extensive data corruption.

Using an IoT agent that includes metadata capabilities, results in less complex rules about agreed formats of data. New data sources can be added in real time, and capabilities for IoT data collection and analysis can be implemented independently of the cloud service, without relying preconceived data formats.

```
Ardexa Design Feature 3: The Ardexa IoT agent includes metadata in each
stream sent by the device. This method carries almost no extra overhead
if implemented well, and the cloud service will know that exact data
format without the need for synchronizing. If need be, the cloud service
can query the metadata for all the agents.
```

Collecting Log Events

One of the more common forms of data collection for an IoT agent is data written to log files. In this format, all the data is contained in a single line, which may contain data elements separated by commas or tabs, and where each line represents an event or a reading at a particular time. As such, there is a date and time entry associated with the event. Meta-data is rarely contained within the event itself. Usually the meta-data must be stored elsewhere. Here is an example of a single line, log entry:

64.242.88.10 - - [07/Mar/2004:16:35:19 -0800] "GET /mailman/listinfo/business HTTP/1.1" 200 637

In this log example, entries are separated by tabs. Some entries are numbers that can be treated as decimals, some as string and some need to be treated as a date/time. An IoT agent should be able to collect log line entries from multiple sources, if need be, and then annotate metadata for each separate field.

```
Ardexa Design Feature 4: The Ardexa IoT agent is able to collect log
entry from many sources simultaneously, and apply metadata to fields in
the stream.
```

Collecting Application/Process Output

In most cases, an IoT developer will need to collect data directly from specialized programs or applications. An application, program or script is likely to send its output either to standard output (ie; prints a result to the console or terminal window after it has been run), or to a file. If the application writes events, line by line to a file, then this can then be classified as a log file, as described in the previous section. But in a lot of cases, a program will print a number after it has been to (say) read temperature, pressure or the state of whether a door is opened or closed.

Most IoT developers will want to capture the output from an application that runs at regular intervals. There may be more than 1 application, and they may run at various intervals independent of one another. An IoT agent configuration for such a system could look something like this:

Application Name	***Frequency (secs)***	***Log to File***	***Filename***
check-users	*10*	*no*	
measure_pressure	*1*	*yes*	*/var/log/pressure*
diagnostics_check	*3600*	*no*	
show-latest-photo	*450*	*no*	*/opt/latest_photo*

The above table defines four applications that should be run at set intervals. The 'frequency' column defines the period in seconds in which the application should run. So in the above example '*diagnostics_check*' will be run every hour, and the 'Log to File' attribute states that the application will not write the results to a file. In the case of '*show-latest_photo*', this may be a face recognition application that writes a file to '/opt/latest_photo'. This photo may then need to be uploaded to the cloud.

Monitoring applications is crucial in IoT. The agent must be able to schedule a process, and capture the output, in whatever data format presented by the application.

```
Ardexa Design Feature 5: The Ardexa IoT agent is able to schedule many
processes to be run at regular intervals. The output from these processes
will then be tagged with metadata and sent to the cloud, in whatever
format required by the IoT developer.
```

File Change Notifications

Sometimes it is necessary to be alerted whenever there is a change to a file. This may be due to an application altering the configuration or running parameters of a system, via a file. In such instances, the entire contents of the file may need to be reviewed and sent to the cloud, or it may enough to just be alerted to changes in the file.

There are two broad methods to monitor file changes. The first is where the changes are notified as a change in the date/time of the file. The second is to positively detect changes by the use of a cryptographic hashing algorithm[6] such as SHA1. A cryptographic hash will produce a unique number based on the contents of the file. The size of the number does not change, regardless of the size of the file. If a hashing algorithm is used, it is necessary to initially store the value of the 'original' hash, which will then be used to detect changes to the file.

There may be many files that need to be monitored. The 'file change notification table' managed by the IoT agent may look something like the following:

File Name	Send Entire File	Use Hash
/etc/controller.cfg	yes	no
C:\monitor\controller.txt	yes	yes
/etc/shadow	no	yes

In the above example the IoT agent will monitor the '*/etc/controller.cfg*' file and send the entire file to the cloud if there is a change in the date/time of the file. For the file entry '*/etc/shadow*' example, the file the entire file will NOT be sent, and changes to the file are verified by a cryptographic hash. The hash provides a unique number for a file, and hence a direct indication if they file has changed in any way. Whilst the file is not sent to the cloud, a notification of change will.

```
Ardexa Design Feature 6: The Ardexa IoT agent is able to monitor changes
to any number of listed files in real time. A date/time method is used
for detection, or the more secure cryptographic hash. If required, the
file itself is sent to the cloud, or simply a notification of the change.
```

Real-Time and Caching

Sending events in real-time is in most cases preferable to caching, since it allows the monitoring of an IoT infrastructure to be conducted in real-time, and allow real times response to issues. In some cases, such as security monitoring or residential alarm systems, real-time monitoring is crucial. However it is inevitable that the connection to the Internet will suffer loss at some stage. It may also be a scheduled regular occurrence, where (for example) a vehicle may lose reception to the cloud.

```
Ardexa Design Feature 7: The Ardexa IoT agent sends data to the cloud in
real time. In event of connection loss, events will be cached, ready for
when the connection is re-established.
```

REMOTE MANAGEMENT

The Internet of Things would hardly be a revolution if people could not remotely manage their house alarm systems, be alerted to a possible vehicle maintenance issue that has been remotely diagnosed, use mobile devices to turn on their heating systems, and so on. All of the scenarios call for remote management. And this is where most IoT implementations will face serious security and functional issues.

In the previous sections, it was discussed why it was important to not allow incoming connections to the Thing. The remote management channel must be part of the same connection established by the agent, to the cloud.

Real-Time Channel

An IoT agent should be configured to establish a connection to the cloud service, and hold it open permanently. Establishing and maintaining an open connection is a relatively low overhead operation, and it enables commands to the sent to the agent in real time. Should the connection be lost, a new one is established, again by the agent. A 'back channel' can then be established, that allows communications in real time from the cloud to the agent. But since its part of the original connection, it does not need to 'punch through' the firewall. Both these channels are encrypted and authenticated.

The back channel, in most cases, offers the most responsive form of remote management. Remote applications can run, changes can be made, data monitored all in real time. Communications overhead associated with maintaining an open channel are low, if implemented in the correct way. A communications channel that is constantly resetting itself due to poor communications may cause some usage.

The type of remote control is very important. Some implementations presume that remote control is about sharing a graphical screen or user interface. Tools such as 'VNC' or 'nomachine' allow such remote control. Sharing out screens or graphical interfaces is very much geared towards humans. It is a process unsuitable to automation. Furthermore, everything can be accomplished from a robust command line, but not every task will be able to be accomplished via a graphical interface.

A robust IoT remote control will enable full, real time access to run commands on the Thing, and the output redirected so it may be read by the cloud in real time. There may or may not be a human at the cloud controlling the Thing.

```
Ardexa Design Feature 8: The Ardexa IoT agent is able to be fully remote
controlled from the cloud, in real time.
```

File Transfers

File transfers are a critical component of IoT activity. A lot of features rely on this ability, including upgrading software, uploading new configurations and downloading data files. An IoT agent should allow files of any nature, including binary files, to be download or uploaded.

The IoT agent configuration should be able to be changed in real time, thereby enabling new functionality. If the configuration is a file, then updating the configuration is a simple step of downloading (to the agent) the new configuration, and then restarting the agent. Similar to the remote commands, agent configuration should be managed remotely in real-time. Changes to the remote management configuration should be in real-time, via the channel opened up by the agent.

```
Ardexa Design Feature 9: The Ardexa IoT agent enables real time file
transfers in either direction, and in real time.
```

Heartbeats and Statistics

An agent's health is best determined by monitoring a regular heartbeat, issued by the agent. Alternatively it may be undertaken through polling via the real-time channel, if one has been established. Either way, by notification or polling, the heartbeat offers the only way for the cloud service to maintain a view of the agent.

An IoT agent may not be able to send regular heartbeats. It may be off-line for extended periods of time, during which the cloud service must manage this situation.

```
Ardexa Design Feature 10: The Ardexa IoT agent enables the option of
allowing a regular heartbeat message to the cloud.
```

Ardexa team has been leading the security industry for many years

At Ardexa, we are building a strong reputation for setting a new standard in IoT implementations. We will maintain the highest industry methods and resources to protect your business. We look for clients that are collaborative in nature and are aiming to lead. We are ready to evaluate your specific needs or simply converse on what is possible.

The Ardexa team has focused their efforts on building the leading end-to-end IoT solution allowing you to focus on what matters most: your device and the insights it can provide.

Please contact us to discuss your IoT initiatives. For more information, please contact us on iot@ardexa.com.

References

1. *https://en.wikipedia.org/wiki/Firewall_(computing)*
2. *https://cve.mitre.org/about/index.html*
3. *https://en.wikipedia.org/wiki/Public_key_certificate*
4. *https://en.wikipedia.org/wiki/Public_key_infrastructure*
5. *https://en.wikipedia.org/wiki/Metadata*
6. *https://en.wikipedia.org/wiki/Cryptographic_hash_function*