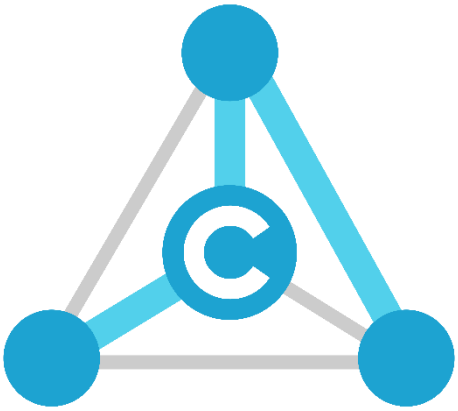

Securely access Manufacturing Data from Anywhere with Any Device

Implement Information-Driven Manufacturing to Improve Processes and Efficiencies

Chris Muench

C-Labs, LLC

February 1st, 2014

www.c-labs.com

Contents

Introduction 3
"The Internet of Things" and "System of Devices" 5
The C-DEngine Solution 8
Usage Scenarios of the C-DEngine.....9
C-DEngine Architecture 11
C-DEngine Based Products.....17
Screening the Market 18
About C-Labs. 25

Introduction

According to the ARC Advisory Group, "Information-Driven Manufacturing is a manufacturing strategy that builds on the concepts of collaboration and network manufacturing and embraces newer technologies to achieve and sustain a competitive edge."[1] Key trends helping to foster growth in Information-Driven Manufacturing include the continued development of M2M connectivity and the global use of mobile devices.

M2M connectivity allows manufacturers to be more competitive by increasing plant efficiency and manufacturing agility. Manufacturers are now able to monitor all of their machines and processes ensuring that they are on-line and operating at maximum performance. Critical factory floor data and KPI's can easily be shared throughout the enterprise providing real-time performance.

Any data just "born" on the edge can impact decision making on all levels of an organization, such as:

- A factory manager who needs to see if all PLCs are still working properly
- A production manager who needs to know the current production status
- A remote maintenance worker that has to gather information about a failure condition before returning to the factory
- An executive who needs to know a specific customers order production status

Upon any failure situations, it is essential for all these stakeholders to get the right information as quickly as possible, independent of their preferred device and location.

In order to improve manufacturing efficiencies by reducing plant and process down time, factory workers must have HMI and factory data delivered securely to any device. By monitoring KPI's and process trends in near real-time, factory workers will be able to detect failure scenarios before they happen, as well as monitor multiple systems across the globe from any location or device. As the costs for networking infrastructure are steadily decreasing, companies see an increased value in connecting all their computing devices to enable faster business decisions.

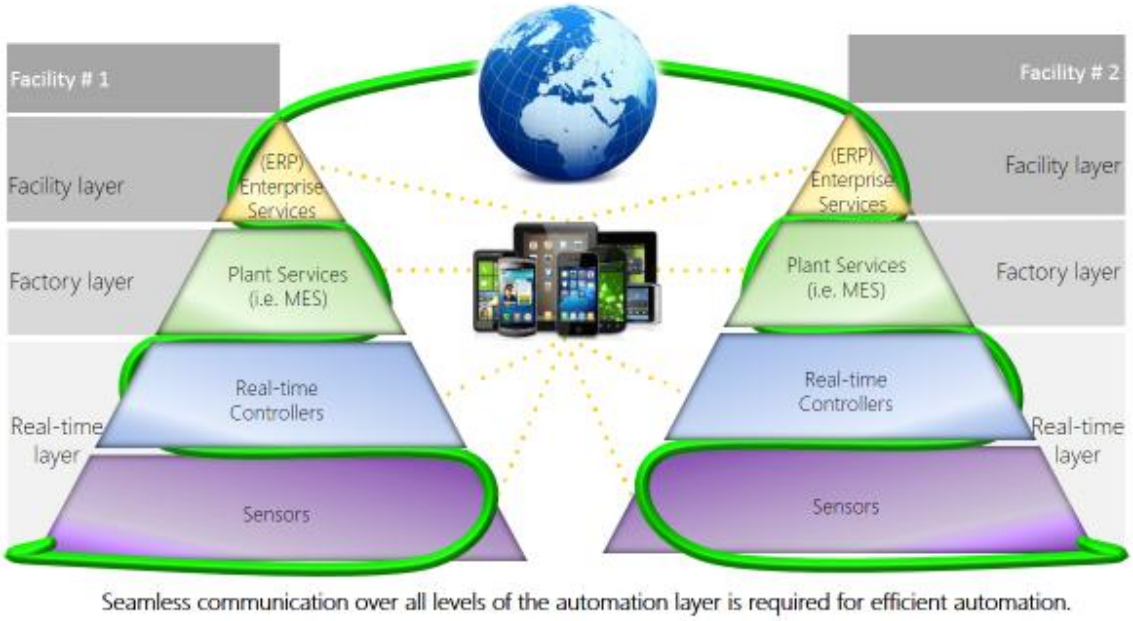

Figure 1: Communication Overview

To effectively implement Information-Driven Manufacturing, there must be direct communications between the factory floors, IT services and mobile devices in near real-time. In many infrastructure environments, these devices and services are located on different network topologies, sometimes behind multiple layers of firewalls. And for companies with multiple facilities, maybe across continents, it is important to see all relevant data at a given time

Creating a unified communication infrastructure requires bridging these network boundaries without compromising security policies and factory performance.

Since a factory floor can have many devices talking to each other, and each potentially has edge born data, the biggest value lies in a real "**Inter**communication **net**work of Devices (Things)" (Aka "The Internet of Things").

C-Labs is providing solutions such as the Factory-Relay base on the C-DEngine for the Internet of Things (IoT) with the specific needs of industrial communication requirements in mind.

[1] Information-Driven Manufacturing. Gorbach and Chatha. ARC Strategies Feb 2013

“The Internet of Things” and “System of Devices”

For many years, the M2M (Machine to Machine) movement has tried to establish a new era of communication, focusing on devices, not on users, which is common in traditional communication scenarios. M2M is enabling communication between two “Machines” in order to make smart decisions without the intervention or involvement of users. While M2M traditionally requires a direct connection between the two machines, most solutions available today use an intermediary to make this connection work. Other solutions put GSM (Global System for Mobile) modules or other modems directly into the data-providing machines in order to allow access from or to the other machine. This scenario typically circumvents any IT enforced security policies and guidelines. In these cases, the connection between the machines is initiated by a machine and not by a human (Machine Centric).

The “Internet of Things” is an evolution of M2M that not only connects two machines to each other, but allows for a “many-to-many” network of machines – now called “Things” or “Devices”.

Figure 2: Internet Of Things

Industrial Networks have supported these “System of Devices” for a long time. Even before the introduction of the “Industrial Ethernet,” factories were and still are using devices connected using

Fieldbus. PLCs are connected to many remote I/Os and other data providers, as well as data consumers such as HMIs, forming a “Network of Machines/Devices/Things”.

Building and Home Automation systems have similar network topology. They traditionally connect through installation buses using serial technologies that are comparable to industrial automation’s fieldbus.

In addition today’s automobiles have an internal system of devices connected via the CAN bus, a serial bus very similar to the field and installation bus. FlexRay, which is comparable to industrial Ethernets, can be found in newer generation cars.

All of these industries are moving away from serial based busses towards Ethernet based networks and are extending the concept of these connected devices towards “The Internet of Things”.

In contrast to the “Internet of Things” many existing solutions provide only the “Internet **for** Things” by using a Cloud Service as the intermediary. These solutions simply “host” a traditional website in the cloud and call it a “Cloud Service”. Their devices connect to that website which stores data in the cloud. The website is then waiting for mobile devices to connect and pick up this data. These solutions have existed for many years. The only new component is, that these websites are no longer hosted on dedicated web servers but are on more flexible and scalable cloud computing clusters. Most of these solutions are custom-made and require high development and maintenance efforts. This is called “The Internet **FOR** Things”.

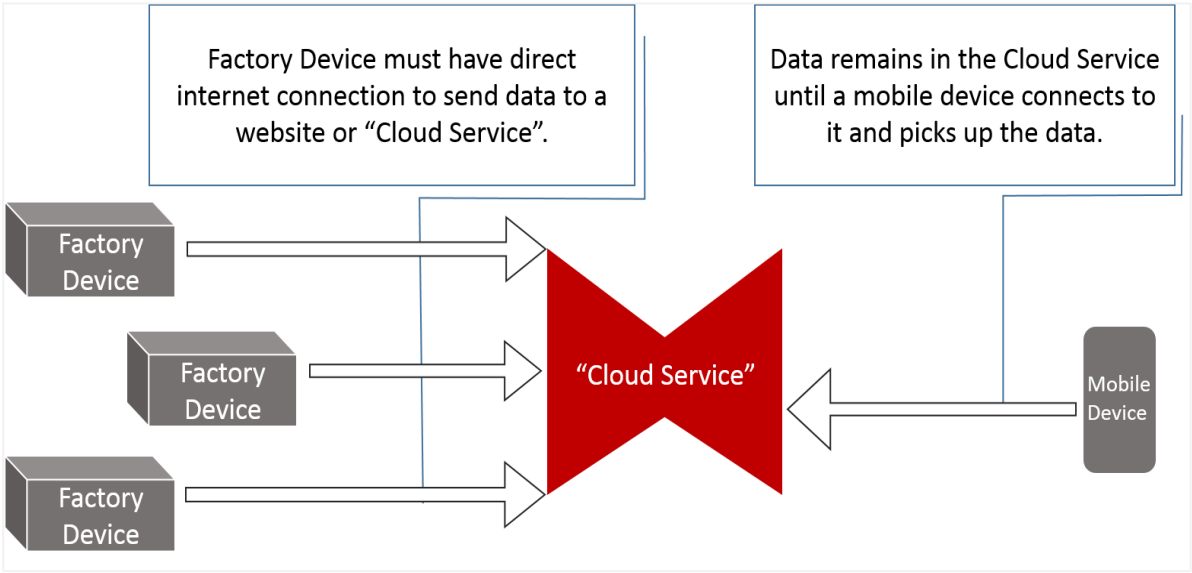

Figure 3: Internet FOR Things

Typical examples of the “Internet FOR Things” you might relate to are solutions such as:

- “Comcast’s Home Security” (http://www.comcast.com/homesecurity/features-web.htm?SCRedirect=true)
- NEST Thermostats (www.NEST.com)
- NinjaBlock (www.ninjablock.com)
- Netatmo Weatherstation (www.netatmo.com).

The “Internet OF things” does not necessarily require a web/cloud service but allows devices to talk directly to each other.

The word “Internet” implies that these devices are able to talk to each other over the internet. But in a real “Internet of Things” the “Internet” stands for “Intercommunication Network” and also describes the direct connection between two or more devices. In order to access a device on the factory floor with a mobile device while you are in the factory, you don’t want to connect to the cloud but connect directly to the device. While you are working in your office, you will connect through your local intranet – still not using the cloud. Only when you are outside the factory will you use the internet to connect back to the factory device. The user experience on the mobile device should always be the same making the access completely seamless.

Depending on the network topology the device is running in, the communication rules and policies are different. On the internet, the security requirements are much stricter and the availability and scalability requirements are often not provided by the devices themselves. Again, an intermediary is used to provide all necessary requirements. The best intermediaries for availability and scalability are computing services in the cloud (public or private). But in the end, this web/cloud intermediary is just another device/”Thing” other devices/Things are talking to.

A major concern for many companies, is the lack of trust in the protection of valuable data stored in the cloud and the associated security mechanisms. This prevents a more aggressive move towards cloud solutions.

The C-DEngine Solution

The C-DEngine offers users a secure and private way of getting data between the plant floor and business decision makers without exhaustive overhead and policy changes. This technology eliminates many of the security concerns that have slowed the adoption of M2M system implementation.

The C-DEngine enables the rapid development and deployment of M2M connectivity using distributed applications that send data via the cloud between devices WITHOUT developing custom web or cloud solutions, or proprietary web portals. This dramatically lowers engineering development investments, reduces the time and cost of applications and solutions. Manufacturers can focus on developing their core business applications as if the application are running on a local PC. The C-DEngine will distribute the application automatically.

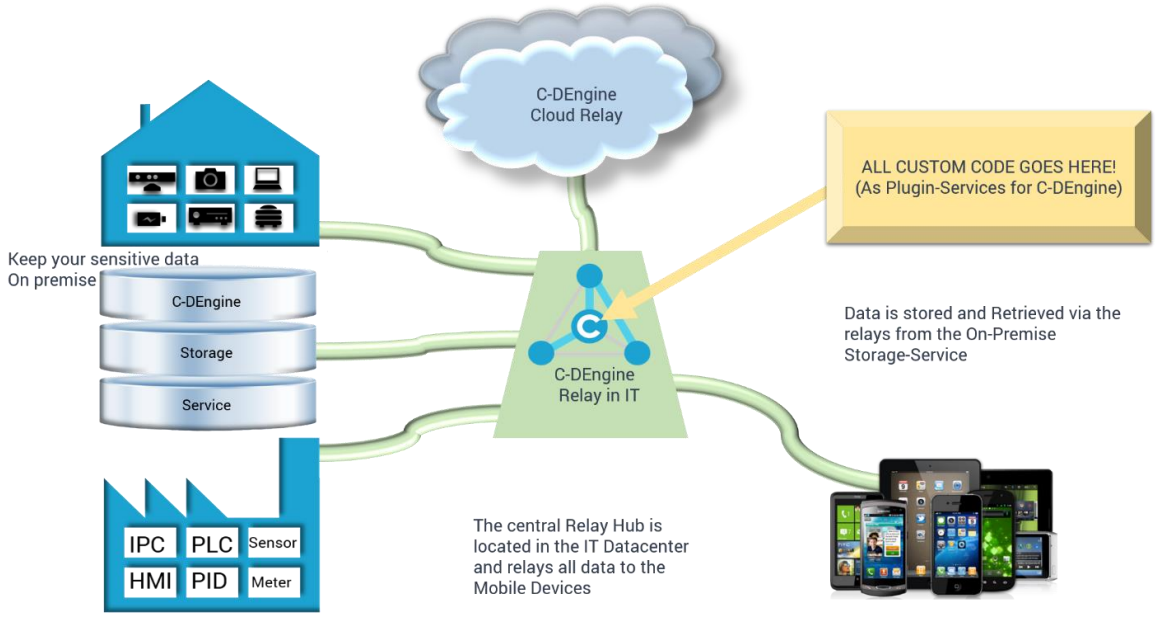

Figure 4: C-DEngine Overview

The C-DEngine was developed to address the issues of connecting the manufacturing environment in a secure manner. The C-DEngine uses the cloud for what the cloud is great for: scalable processing and computing – and not necessarily for storage. Data is routed (relayed) through the cloud with an option of processing the data (i.e. for compression, event handling, calculations, and other operations) without a digital footprint. Without any "processing plugin" (also referred to as a "Plugin-Service") in the Cloud Node of the C-DEngine, the data is just relayed from one device to another while both devices connect securely via HTTS/SSL to the cloud – the perfect intermediary in traditional M2M scenarios. Running the C-DEngine on factory devices and IT services allows companies to build a completely independent "Internet of Things".

Distributed applications, solutions and systems can be built WITHOUT any knowledge of underlying communication infrastructure while still complying with enterprise IT communication policies. Solutions can easily be deployed in existing infrastructures adhering to any security guidelines, regulation and policies as the C-DEngine only requires passive (outbound) access to the internet and does not change or open any ports in the existing firewalls.

Solutions are now secure and reliable because security is established on the lower IP layers (i.e. SSL/TLS or IPSec), the protocol layer and by using data encryption of the C-DEngine telegrams. Implementations with the C-DEngine do not require modifications to existing security policies.

Usage Scenarios of the C-DEngine

C-Labs main focus for the C-DEngine is in industrial automation scenarios where OEMs and Manufacturers access KPIs and other data "born on the edge". This data, born on industrial devices such as PLCs and HMIs, can be accessed even from mobile devices connected to the internet. With traditional solutions a proprietary communication path has to be established between the industrial device and the mobile device using VPNs or even open inbound ports. This is done by either by-passing the IT security or requesting that the IT administrator set up the communication infrastructure. The C-DEngine dramatically simplifies this process by using only Port 80 outbound to a service in the cloud that provides the connectivity between the industrial device and the mobile device. With its unique mesh-based relaying capability, the C-DEngine can route data properly and securely through DMZs in multi-tier IT infrastructures. This allows design of network architectures where the industrial devices are running on isolated private networks because only an outbound access to the DMZ is required.

Web based HMIs targeting mobile devices are on the rise while traditional HMIs with proprietary technologies struggle to keep up. While those web based HMIs are easier to expose via the internet, security and scalability concerns are increasing.

One important aspect of the C-DEngine is its ability to deliver tight security. Similar to the RSA Hardware Key Chains displaying a new number/key every 30 seconds, the C-DEngine uses a lock-step security that changes the communication endpoint between two nodes after every call. Additional security and encryption levels can be established on protocol (TLS/HTTPs), telegram (Encryption of C-DEngine Telegrams), device (Specific device ID key) and user level (Username/email + password). The integrated ISM (Intelligent Service Monitoring) of the C-DEngine allows administrators to monitor the distributed system at any time and respond to unexpected events.

Home/Building Automation Scenarios

Building Automation is becoming more sophisticated and remote access to information is important to building managers – even when they are not on site. Access to information and control of devices is also a

great value-add for tenants of buildings, hotels and modern homes. In addition many control options for your home devices, such as preheating an oven, changing thermostat settings, turning lights on and off from another location, many other scenarios are emerging. Monitoring energy usage has become one of the key scenarios for home/building automation. A service, based on the C-DEngine (such as the C-Labs Home-Relay) can be installed in every hotel room, building segment or home and connects to HVACs, building controllers, energy meters and other local devices in a building that was not designed to support mobile devices. This service includes user friendly UX pages optimized for mobile screens, and relays these pages securely via the IT DMZ and the cloud to an authorized user on a mobile device.

Looking towards the increased urbanization of cities, the future possibility of "Mega-Cities" and possible regulation and energy restriction scenarios, it might become necessary for the government to monitor specific KPIs of buildings for regulatory or energy planning purposes. Buildings with high energy peaks or averages could get penalized and buildings with no peaks and low averages could get incentives. In order to detect these peaks and co-relate them to events inside the building, requires constant monitoring of all energy consuming devices. For building managers this data is very valuable to create energy purchasing portfolios, schedules and reporting.

Connected Car Scenarios

Early on, C-Labs was looking at scenarios that included the automobile industry in distributed system scenarios that are now known as the "Connected Car." Many cars already report car-health information to the dealership that sold the car, and have the car dealer call the owner to arrange a maintenance session. With the C-DEngine in the car and the cloud relay ready to send data back to the manufacturer, many new scenarios are possible. Parents are able to remotely monitor their child's driving style, or create an engine performance log for road trips. Connected cars can also communicate with other cars to detect ad-hoc traffic situations (V2V) and calculate more accurate trip ETAs and plans for refueling routes. Other scenarios for connected cars include playlist synchronization with personal computers at home (V2C), social media posts while driving by pushing a button on the steering wheel using the connected cameras already mounted in newer car models (V2I).

For car manufacturers it might be interesting to learn more about certain car models by analyzing the driver's behavior while driving. Similar to the opt-in feature in software products that are asking the user if they want to "participate in improving the product", connected cars could ask the driver the same question if they would be willing to help improve the next generation of the car in exchange for a free checkup or other incentive. Distributed systems built with the C-DEngine can ensure that no personal identifying data is transmitted to the manufacturer and only statistically valuable data is consolidated at the manufacturers IT center and in their on-premise databases.

The Internet of Things has many more areas of impact such as in personal health-care and monitoring and the gaming industry. This white paper only addresses "Automation."

C-DEngine Architecture

The C-DEngine (C-Labs Distributed Engine) is a runtime library written in .NET and optimized for many different platforms including embedded devices, enterprise servers, elastic cloud computing services and mobile devices. On non-Windows platforms, the C-DEngine uses the latest editions of Mono for Linux (Debian, Ubunto, Suse), Android (MonoDroid), iOS (MonoTouch) and MAC (MonoMac). For solution architects and developers, the C-DEngine provides an asynchronous and event driven API optimized for multi-core and elastic computing environments – scaling from small single-core ARM based embedded devices to high performance cloud architectures. In addition to the lock-step security system, all necessary communication APIs and integrated device and user management components are included. The C-DEngine also provides UX components for HTML5, Silverlight and Windows RT based clients. The C-DEngine uses a plain HTTP/HTTPS and WebSocket protocol to communicate between the nodes adhering to all enterprise class firewall and security policies. Solution developers can focus on writing data providers and user experiences while the C-DEngine takes care of the communication between the nodes.

One unique feature of the C-DEngine is that solution architects can build distributed systems that include the cloud for relaying, computing and processing tasks without storing any data in the cloud. Other features include usage and telegram tracking for costing, redundant routing, failover routing, and encryption of telegram payloads and auto-discovery of compatible service on a network.

What is a thing?

For the C-DEngine a "Thing" is the virtual representation of an electronic device that can be uniquely addressed on a bus or network. "Things" can expose properties, events and services.

The C-DEngine allows developers to build simple systems using a local (on-site "Thing Hosting") relay, a cloud-relay and a data consuming device (i.e. a mobile browser). The local relay contains all the logic for collecting the data of the system including the UX components. The cloud relay just routes this data and the consuming mobile device displays it.

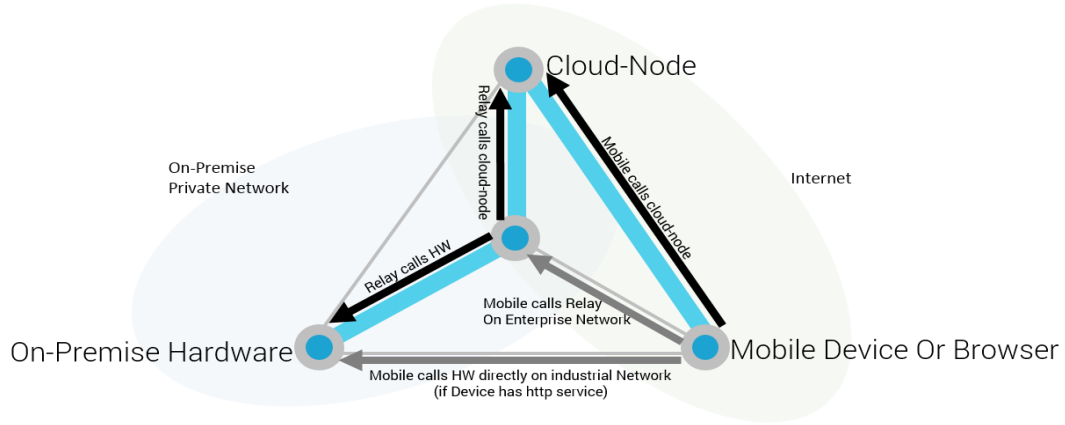

Figure 5: C-DEngine Overview

More complex systems include multiple local relays also called “C-DEngine Agents” that talk to each other. Each one of these agents can host any number of “Things”. A designated local relay sends the data to the cloud-relay where it is relayed to multiple consuming nodes, creating a real “many-to-many” distributed system. Only the data required by a device behind a relay is actually sent via the relay in order to keep the telegram flow optimized.

Communication security and session tracking is always established between two nodes, while data security is established between a providing node and the consuming node. This allows for shared cloud scenarios where multiple agents/relays of one customer sends data only to authorized devices with the same security context. Another customer can use the same cloud relay, but can only have his/her devices talk to each other.

In environments that require more scalability and even tighter isolation, dedicated cloud relays can be activated and even redundant and failover routes can be declared in the devices and the relays.

One big difference to traditional systems is the amount of engineering necessary to build those services when using the C-DEngine. In a C-DEngine system, development and engineering is required ONLY on the node that is hosting the “Thing” (A relay or an agent). No additional engineering for the cloud service or mobile device is necessary. Therefore, any device application developer automatically becomes a distributed systems developer.

Additionally all devices can talk directly to each other and do not require the cloud node to communicate. This allows access to local on-premise devices from any network and therefore provides a real “Internet of Things”.

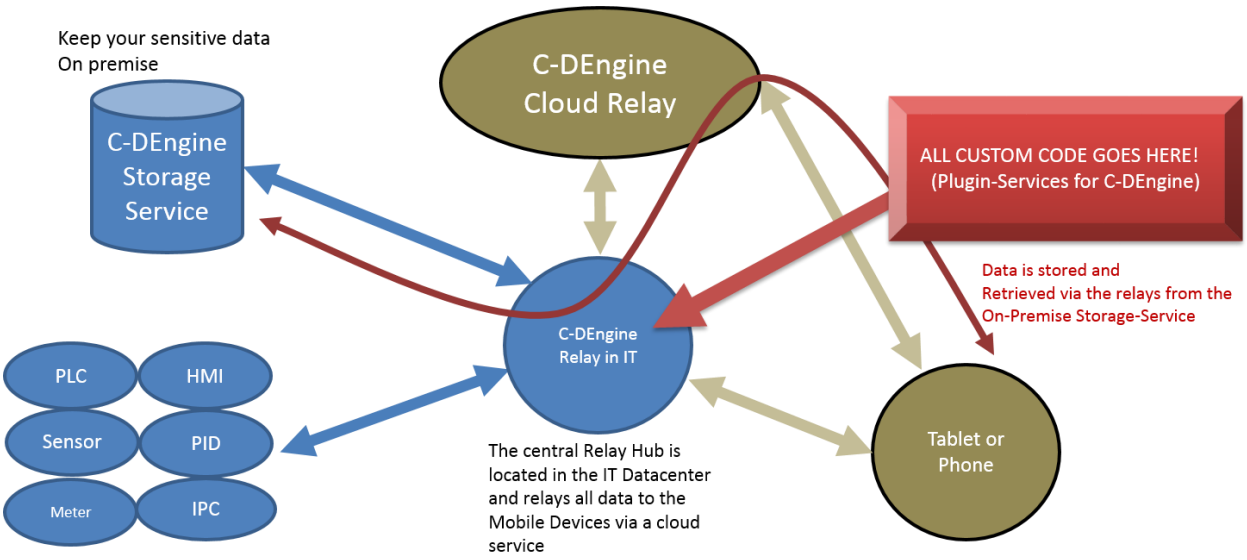

Figure 6: Nodes in a C-DEngine distributed solution

The picture above shows a typical setup of nodes participating in a C-DEngine distributed solution. Let's look at the flow of data in such a solution.

The goal is to see live data on the Mobile Device (Consumer) that is produced or "Born" in the On-Premise hardware (Provider). If the On-Premise Hardware does not have any inbound ports open, the mobile device cannot directly connect to the hardware, a very common scenario.

Many companies offer solutions that require the On-Premise hardware to push their data to an internet based web service or web server. The mobile device then connects to a website on that same internet server and can view the data. This solution has several issues:

1) A direct connection between the on-premise hardware and the internet service is required – even though only outbound (no inbound port on the On-Premise Hardware).
2) In order for the device to push data to the internet service with SSL/HTTPS, the device has to support these security algorithms. Most devices in factories and even consumer homes do not have SSL capabilities and the data is transmitted in plain text that can be "monitored."
3) The internet service has to be developed to manage different devices and users which can be a very complex development process.

With C-DEngine based solutions there are multiple benefits:

1) The On-Premise device is only talking to an On-Site Relay that might even be hosted in the DMZ of the company.
2) While traditional solutions require the On-Premise Hardware to actively push the data to the cloud, the C-DEngine relay can also actively call into On-Premise Hardware allowing for passive implementations of the On-Premise Hardware (i.e. have a small JSON webserver in the HW instead of an actively pushing data out).
3) The On-Site Relay is talking to the cloud relay using SSL/HTTS and "Upgrades" the On-Premise Hardware to the higher security standard.
4) The C-DEngine is a relay technology and does not require any cloud development to relay the data from the On Site relay to the Mobile Device. Authentication is established between the Cloud and the mobile device and the cloud only requests data from an On Site relay that fits this authentication. The authentication service is part of the C-DEngine and does not have to be configured or developed.
5) The complete UX visible on the mobile device is hosted in the On Site relay and transferred to the mobile device by request only. Therefore all custom code is running in the On Site relay on premise and not necessarily in the cloud.
6) The C-DEngine Cloud relay node also supports plugins, which can be used to enhance a solution with aggregation and processing services.

Security of Distributed Solutions with the C-DEngine

Security is a very important aspect of any solution no matter what industry the solution gets deployed in. Manufacturing and Industrial Automation is no exception, and it has become more important for manufacturers and solution vendors to put security first.

Solutions with the C-DEngine will have multiple levels of built-in security. The C-DEngine "upgrades" the security of on-premise devices that cannot include complex security and encryption algorithms due to footprint or performance reasons. As seen in the image below, the On-Site relay can require that only SSL and HTTPS requests are accepted.

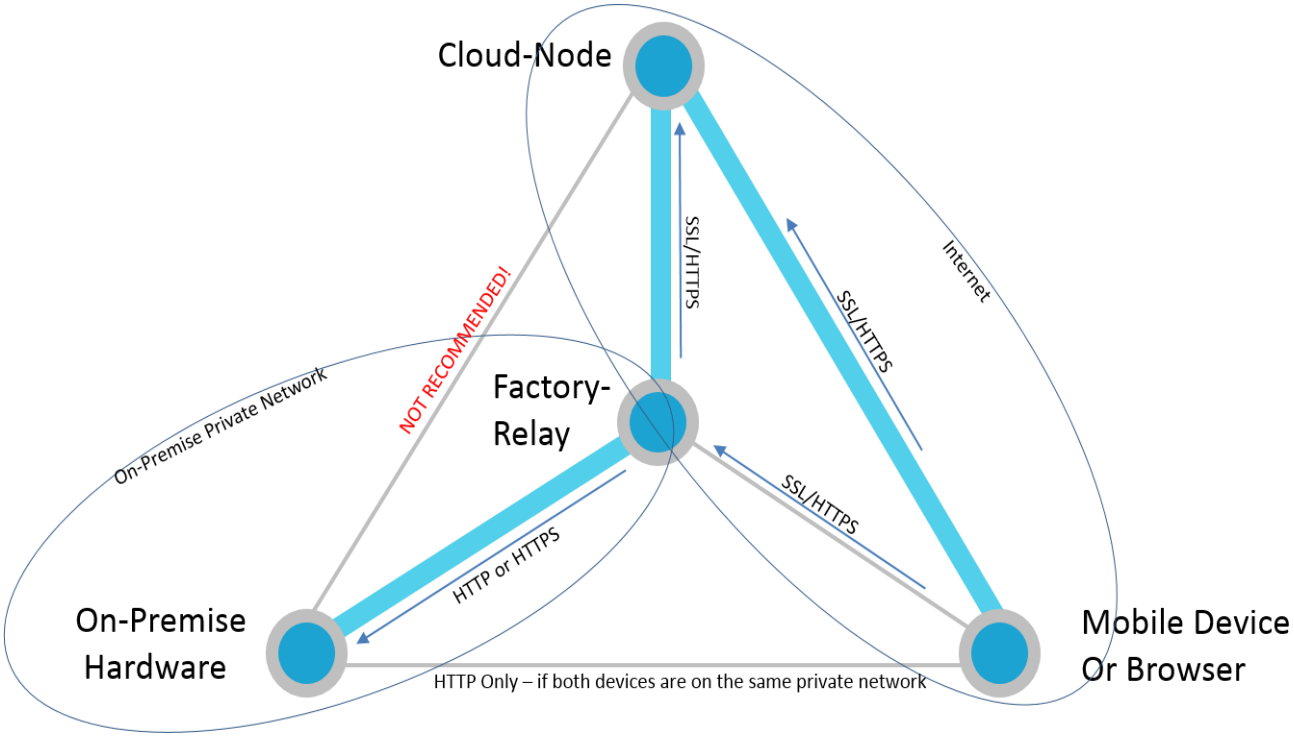

Figure 7: C-DEngine Security Overview

The Cloud requires SSL/HTTPS for all nodes to connect. Even on-premise devices that have no SSL/HTTPS support can securely participate in solutions that allow access by devices connected via the internet.

Additional security is built in each C-DEngine node. When two nodes are talking to each other, they create a specific security context, based on RSA keys, for just this connection and session. In the picture above there is a security context between the On-Premise hardware and the On-Site relay that is different from the security context between the On-Site relay and the Cloud-Node. This ensures that if one of the nodes goes down, the rest of the C-DEngine mesh network is not compromised. Mobile devices in particular are very loosely connected devices and can “come and go” from the network very frequently. Once a mobile device has gone away, the cloud will terminate the security context for the mobile device in seconds. Even if the device comes back during these few seconds, it has to request a new security context and cannot “cache” any old context.

While traditional solutions connect the On-Premise hardware directly to the Cloud, we do not recommend this path except for very controlled environments. Allowing a direct path from an unsecure device to the cloud increases the possibility for intruders or at least listeners.

The C-DEngine has built-in mechanisms to prevent DOS- (Denial of Service), Replay and Injection attacks. The most dominant of these mechanisms works because each node running the C-DEngine changes the Connection Url after each call to a Url. Replay attacks record a previous call and play the call back against the same Url. Since that Url is no longer valid the call will not succeed.

C-DEngine Based Products

The Factory-Relay

C-Labs licenses the C-DEngine to companies that want to implement distributed systems and solutions, but don't want the complexity of managing all the communication and security implications a custom development brings.

The Factory-Relay is a product based on the C-DEngine that delivers a ready-to-use application of the C-DEngine. It provides customers with the ability to relay existing websites of industrial devices via the cloud to mobile devices without any engineering effort.

Existing web HMIs or maintenance and configuration pages of industrial devices can be accessed by mobile devices connected to the internet via the Factory-Relay using a simple configuration. To extend the functionality of the Factory-Relay, the plugin system allows developer to write custom plugins that can read proprietary data from devices and deliver them securely via the Factory-Relay to mobile devices. The plugin-storefront is a market place for generic plugins that customers might want to download and add to their Factory-Relay.

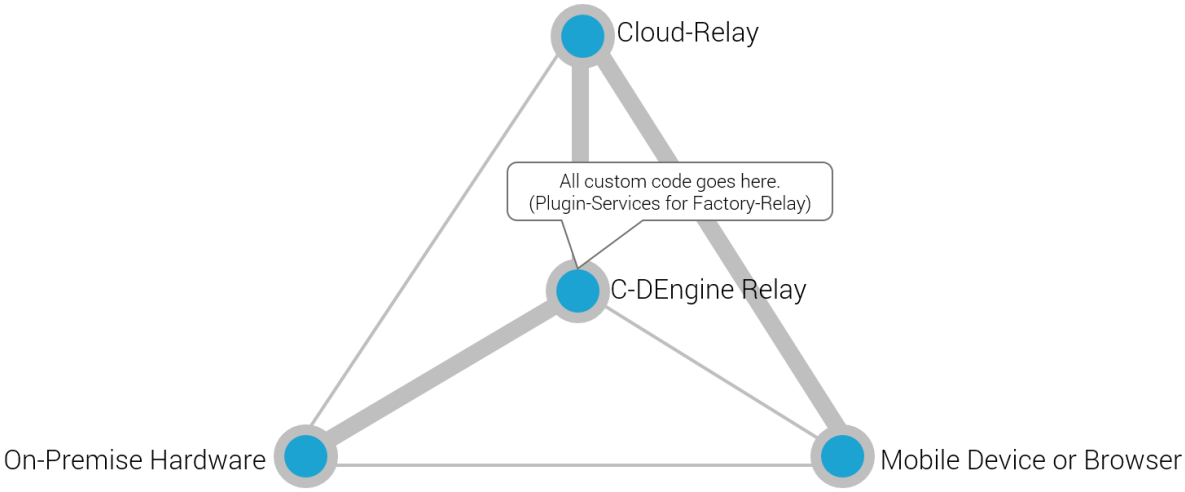

Figure 8: C-DEngine and Rapid Application Development

Screening the Market

There are some technologies commercially available on the market or available as a community maintained shared library that could be used to build distributed solutions.

None of these technologies provide all the features the C-DEngine has as core functionality including:

- Mesh-based relaying of data
- Plugin-services on any nodes with the exact same API
- Failover routing
- Multicast publishing
- Common UX Meta Model combining all plugin-services of one solution in a unified UX
- Access to ANY data by ANY device from ANY-where

Hardware based M2M solutions

Most of the hardware based solutions require changes and or additions to the firewall policies or even circumvent these policies completely by providing cell phone (GPRS) access inside these proprietary hardware components.

MQTT (Message Queuing Telemetry Transport)

Invented by IBM, this transport protocol is mainly used to deliver device data to a central broker for further analyses and computing. It does not support device to device communication and requires direct socket connections between the device and the broker requiring device and broker on the same network. MQTT does not specify any security or reliability rules and is therefore unusable for communication between internet connected devices and factory equipment.

DDS (Data Distribution Service)

DDS is a data centric real-time publish/subscribe transport protocol. It is designated for high-speed device to device communication on a single network. DDS can be extended using DDS Routing services across WANs, but requires specific router configurations and open ports to establish the routes. Currently, DDS does not support any security besides the network layer security such as TLS/SSL.

OPC (DA/XML/UA)

OPC a not a competitor to the C-DEngine but rather a protocol that could be relayed via the C-DEngine. The only requirement to relay OPC is an OPC proxy on one side and an OPC stub on the other side. Both proxy and stubs would be plugin-services for the C-DEngine and handle the traffic between two nodes – wherever they are connected in the world.

Proof-of-concept implementations of plugins for OPC-DA and OPC-XML exist. A OPC-UA plugin are planned.

SignalR

SingalR is a real-time communication library for ASP.NET allowing to build distributed systems of nodes via the internet. It has high-speed communication paths and scales very well in cloud environments. It uses standard HTTP ports and has multiple fallback options for connectivity (such as WebSockets, Persistent-Connection and Http polling). It does not provide any integrated security nor does it allow for automatic relaying of telegrams. Singular requires development on each node to design distributed solutions.

SignalR may be able to be used as a communication channel for the C-DEngine – especially for the communication between Relay- and Cloud Node and Cloud Node to Mobile.

NodeJS

NodeJS is a small library that allows developers to build nodes very quickly. NodeJS is a mini-Web-Server that other nodes can connect to – comparable to the C-Labs "passive node". In order to create distributed multi-node mesh based systems, all communication protocol details and the security have to be developed on top of NodeJS.

Xmpp (see http://xmpp.org/about-xmpp/technology-overview/)

XMPP is the Extensible Messaging and Presence Protocol built for standard communication between different devices. While XMPP allows for a forwarding server it does not provide this functionality out of the box. XMPP is inherently synchronous and requires inbound ports to be open on both nodes communicating with each other. There is a possibility of relaying XMPP telegrams (called XML stanzas) via the C-DEngine.

Other M2M Solutions

There are many custom and proprietary M2M solutions and technologies available. Most of them provide connectivity via GPRS or other cell-based communication paths. Most of them bypass all IT security in place at larger installations and present a high security risk. Those technologies that are using an intermediary are often using this intermediary to bridge between two nodes and do not allow sending data over multiple nodes limiting their usability to simple scenarios. And lastly these solutions are built for communication only and do not provide extensibility to run processing services on participating nodes.

FAQs

Communication Questions

How is a single device (node with the C-DEngine) identified in the network of engines?

Every node has a unique DeviceID. Telegrams are routed through the C-DEngine Relay network until the telegram has reached the node with the designated DeviceID.

There are three different types of nodes:

- Passive Nodes: It a REST/WebSockets Service processing incoming requests and sends responses to Active and Relay nodes. Cloud nodes and traditional Web-Server nodes are passive nodes by default, but can be configured into Relay Nodes.
- Active Nodes (Phones, Web-Browser, JavaScript, Silverlight, Metro and other Client-based end nodes): Active Nodes can only send out REST/WS requests to Relay Nodes or Active Nodes. They themselves cannot relay any telegrams.
- Relay Nodes (Services, Applications, Cloud and Web-Server nodes with enabled REST/WS Services): These nodes contain both REST/WS service and can send out REST/WS requests.

What is an "Agent"?

An agent is a very small version of the C-DEngine running on devices that are not directly addressable by mobile devices. These agents deliver the UX Metadata of the Things they manage/host but do not provide the full HTML5 runtime to other connected C-DEngine nodes. Typically a Factory-Relay/Home-Relay is consolidating the meta-data of all agents and provides the complete HTML5 UX to the mobile device.

How does one node connect to the other nodes? Must an address identifier be known?

Active and Relay-Nodes know the addresses of their next nodes. This can either be configured during startup or these addresses can be found dynamically using UPnP or Bing discovery. Each node retains a list of addresses that can be configured as a "redundancy failover" list or a "multicast connection" list.

- Redundancy Failover: If a node with the active address cannot be reached, the node will automatically select the next address from the list and tries to establish communication with that node.
- Multicast Lists: a node establishes connection to all addresses in the list.

In both cases there are (configurable) rules if and when addresses will be removed from these lists.

A passive node only retains a list of all incoming active nodes. If a node does not send a heartbeat or data telegram in a (configurable) timeframe, the active node is removed from the list.

Is there a directory or broker service?

No there is no directory or broker service. The C-DEngine relay network is self-healing (using the redundancy failover list and discovery mechanisms) without the need of a directory. It is based on the mesh-network paradigm.

Do we understand it correctly that a central cloud service is always a must?

A central cloud service is ***not*** a must. The central cloud relay service is required only if:

- Internet connected Mobile or Remote nodes want to participate in the C-DEngine Relay network
- The nodes providing data are behind firewalls that only allow Port 80 OUTBOUND traffic

If there are only nodes (including phones with Wi-Fi, Browsers or other client-type devise) on the intranet, there is no need for a central cloud service as long as there is at least one active and one passive node on the network.

If there is an INBOUND PORT open for a relay service (i.e. on a Server in an IT DMZ zone), the user can connect Internet based devices to that relay service directly. This is a major advantage of using the C-DEngine, because getting INBOUND ports open in Enterprises is a big policy and IT hassle requiring justifications and paperwork.

Is this the concept idea: the provider says I have data, the consumer says I want data, the cloud service requests it and provides it?

Yes, in essence this is correct. The complete workflow is:

- The provider sends the data to a "PublishingTopic". Only those nodes that have a subscription to the "PublishingTopic" will receive the data.
- If a consumer wants the data, it has to send a subscription of the "PublishingTopic" to the cloud.
- The cloud relays the subscription to all nodes it knows about. Each receiving node relays it again to all nodes they know about and so on.
- Once the subscription has reached the publishing node, the data is relayed back across the network of nodes knowing about the "PublishingTopic" subscription until it has reached the consumer again.

If the provider has only port 80 (or 443) outbound, how can the cloud service request the data when the consumer wants it?

Data can only be sent between Active and Relay nodes, Relay and Relay nodes and Active and Passive nodes. Two Active nodes cannot talk directly to each other (both are only sending and cannot receive) and two passive nodes cannot talk directly (both waiting for incoming requests).

In this case, the provider would be an Active Node (sending telegrams via port 80 or 443 using HttpRequest) to the cloud node. The consumer (i.e. a phone) again would be an active node requesting the data from the cloud node. This is VERY similar to a traditional Web Service concept, except with a C-DEngine cloud-node there is no need to develop any cloud-based code to relay the data to a mobile device. Optionally solution developers can develop plugins that run in the cloud-node.

If there is only a passive provider (i.e. a Web Server in a PLC) another node will be needed between the passive provider and the cloud. This active node can also do mapping between an unsecure provider and the secure cloud-node. The passive provider would talk HTTP to the Relay-Node and the Relay-Node would talk HTTPS to the cloud node, and can additionally encrypt the payload of the message.

You mentioned "Bing" and "UPnP" to search for devices. Please explain who is searching for what and when?

UPnP – Universal Plug and Play (http://en.wikipedia.org/wiki/Universal_Plug_and_Play)

UPnP is an industry standard used to find and control other devices on a LOCAL network (UPnP cannot be routed across subnets).

Every Passive and Relay node can be configured as an "UPnP device" providing metadata associated with the Engine (i.e. Application Name and supported Sub-Services/Plugins). Every Active and Relay Node can search for these devices using the UPnP M-SEARCH (http://en.wikipedia.org/wiki/Universal_Plug_and_Play#Discovery). The C-DEngine ONLY uses the discovery protocol specified in the UPnP standard. C-Labs may consider extending the support of UPnP in a future version of the engine, but not for the Industrial Automation Market.

Once a Relay or Passive Node has been identified and validated using the metadata, the address of this node is added to the Redundancy - or multicast-list and a connection can be established.

Bing Discovery

Bing discovery can be used by Active and Relay-Nodes to find Cloud-Nodes of the C-DEngine. Similar to UPnP, the Bing search APIs are used to find signatures of C-DEngine Cloud-Nodes. Once a Cloud-Node is identified and validated, the Active or Relay-Node can establish a connection to the Cloud-Node. In order for Bing-Search discovery to work, the Cloud node has to be registered with or indexed by Bing.

How do I know that all nodes responding are all nodes? Maybe some nodes are unavailable.

A C-DEngine node only knows about other nodes that have active subscriptions filed with it. If all subscriptions of another node are removed, the connection to this other node is removed as well. If a node is not "refreshing" its subscription every (configurable) time interval, the node is considered unavailable and removed from the redundancy- or multicast list.

Is the data pushed or pulled?

Depending on what kind of nodes are talking to each other, pushing or pulling takes place:

- Active- to Passive-Node or Active- to Relay-Node: Active-Node "Pushes" information to the Passive-Node but has to pull results back (automatically done in the engine).
- Relay- to Relay-Node: Both Nodes push information. This is the fastest communication possible.

When is the data provided to cloud?

Only on demand by a consumer connected to the cloud. If the cloud has no active subscription, the provider does not send any data to the cloud. This saves traffic and bandwidth and is more secure.

Application designers can build cloud service plugins for the C-DEngine that allows building services around cloud storage, consolidation, analyses or other elastic computing and scalable solutions.

What happens if a connection is lost, i.e. if a node is not available?

All nodes try to re-establish communication with the nodes that lost connection. If re-connection fails, the next node in the redundancy-list is being connected to. If this list is empty or no connection can be established, the node will cycle back through the redundancy-list. Parallel UPnP and Bing can be used to find other compatible nodes.

How is the security done by the engine itself?

The C-DEngine uses the REST and WebSocket protocols. It supports all underlying security mechanisms such as SSL/TLS and IPSec.

For each session between two nodes, a unique RSA key combination is created for just that particular session. Once the session is over the RSA keys are no longer used.

Optionally the application developer can encrypt the Payload of each telegram.

Additionally the C-DEngine is providing necessary protection against replay attacks, telegram injection, and telegram manipulation attacks.

Scoping (Security Context)

What is a Scope ID?

All ScopeIDs are GUIDs encrypted with rotation and serial number as strings.

Example: OdNCUYkdR5H2PZ5nYb8XO1w8_TkdGx3vwWq2d42dCR8=

What types of Scope ID are supported by the C-DEngine?

Currently the C-DEngine Supports four levels of ScopeIDs:

ApplicationID - APPID (Application Scope):

An ApplicationID specifies the scope of a “Distributed Application” (sometime also known as “Distributed Solution” or “System of Services”). Only services on devices/nodes with the same ApplicationID can send telegrams between each other.

ApplicationIDs can only be generated by C-Labs or by companies who purchased a “Universal C-DEngine” license. Without a valid ApplicationID the C-DEngine will not start. The ApplicationID is the license key to execute the C-DEngine and has to be licensed from C-Labs.

The ApplicationID can be compared to the private key of a digital certificate. It is also used as one of the keys for the encryption of telegrams. **Companies should never expose the ApplicationID to any 3rd party and will never be sent between nodes**.

ScopeID – SID (Site Scope)

The ScopeID defines specific devices in an application scope that can talk to each other.

ScopeIDs can be generated by companies with a valid licensed ApplicationID. There is no limit to how many ScopeIDs can be generated per ApplicationID.

ScopeIDs are encrypted using the AID as one of the encryption key. Therefore it is not possible to generate arbitrary ScopeIDs without having a valid ApplicationID.

FederationID (FID) – (Federation of two SIDs)

A federation ID allows two ScopeID realms to talk to each other. In the example above if CustomerA and CustomerB would agree on a common FederationID, devices of CustomerA could talk to devices from CustomerB and vice versa.

SEcurityID (SEID) – (Security Tokens per telegram)

SecurityIDs are individually and on the fly generated for each telegram sent via the C-DEngine. This ID ensures protection against:

- Telegram Replay and DOS Attacks (Sending the same telegram with the same SEID will be rejected)
- Telegram Manipulation Attacks (Changing the content of a telegram will change the SEID if the SEID is not properly updated. Using the secure encryption algorithm, the telegram will be rejected.)
- Telegram Injection Attacks (In order to generate a new C-DEngine Telegram, the complete encryption algorithm and other meta information of the C-DEngine telegrams have to be known before a new valid SEID can be created)

How are ScopeIDs generated?

ApplicationIDs can only be generated by C-Labs or using a tool provided by C-Labs to "universal" customers.
ScopeIDs can be generated on the fly using either an intrinsic engine method call. Once a ScopeID is changed on the fly, telegrams will no longer be relayed/accepted using the previous set ScopeID.

This is useful for consumer scenarios in the car/music/diagnostic example mentioned before. If a consumer buys a new car or phone he can activate a scope using a registration screen. By entering an 8 digit key (i.e. on the phone) a new ScopeID is generated. By entering the same key in the other device (i.e. Car) the two devices can talk to each other. This is very similar to Bluetooth pairing except that BT does not encrypt against the pairing key or send telegrams via Cloud-Relay-nodes.

What's the difference between Scoping and User Management?

Scoping ensures that on the lower communication layers, telegrams are ONLY sent between nodes having the same scope IDs. Relay-nodes will ONLY forward telegrams with the exact same and valid scope ids.

This reduces the communication traffic between nodes and ensures an extra layer of security (telegrams not being sent are safer than telegrams that are sent and later discarded on the target node).

User Management is very application specific. Application developers can decide to use the C-DEngine built-in user management that, develop their own user management, use ActiveDirectory or any other directory service, or do not require user management at all (like the Car/Phone scenario).

The C-DEngine includes a minimalistic user management system based on access levels and roles mapping users to specific ScopeIDs.

The UM defines a set of users and a set of roles and stores this information in a node. For each user an encrypted UserID/Password token is generated. If a telegram enters a node, the application will validate this token and hands a UserID to the C-DEngine plugin. It can look up the roles of this user (again using a C-DEngine API call), resulting in allowing or preventing access to certain assets of the node. The use of this UM is not required.

The disadvantage of a UM is that all telegrams have to be sent to a node for permission check. With an increasing number of users in the system the node has to handle more telegrams. Especially for resource limited device this is neither cost- nor resource (bandwidth, footprint and computing power) efficient.

About C-Labs.

Winner of Frost and Sullivan's Entrepreneurial Company of the Year Award for Plant-To-Enterprise Integration Solutions North America 2012, C-Labs delivers cost-effective enterprise software solutions allowing manufacturers to distribute real-time business intelligence, simplify communications between the factory floor and the IT enterprise, and increase efficiency within a factory. C-Labs licenses the C-DEngine, an App Framework allowing solution developers to rapidly build highly distributed IoT solutions, across multiple network topologies and delivering modern NMI User Experiences. The C-Labs C-DEngine combines the strength of smart connected devices, cloud computing, mobility and real-time distributed business intelligence through secure and firewall friendly communication protocols that comply with existing enterprise policies. To learn more about C-Labs visit www.c-labs.com

FROST & SULLIVAN

Entrepreneurial Company of the Year Award
Plant -to-Enterprise Integration Solutions, North America, 2012

About Chris Muench.

Chris Muench is the President and Chief Executive Office at C-Labs, which he founded in 2009 and focuses on Solutions for the Internet of Things. He brings twenty-five years of industrial automation software development experience and passion together in C-Labs to deliver exceptional solutions for today's modern factory automation environment. Chris has delivered solutions while working at Siemens Energy and Automation and at Microsoft. He has technical expertise in Windows Embedded, Cloud- and Mobile Computing, Natural User Interface (NUI) and Experience (UX), and advanced communication protocols including OPC. His credentials include a Bachelor in Data Technology and he has been a Microsoft MVP for ten years. Chris can be reached at chris.muench@c-labs.com.