July 2023

REMOTE AGENT-BASED OBSERVABILITY TECHNIQUE FOR IOT APPLICATION TELEMETRY DATA COLLECTION

Akram Sheriff

Rajiv Asati

REMOTE AGENT-BASED OBSERVABILITY TECHNIQUE FOR IOT APPLICATION TELEMETRY DATA COLLECTION

AUTHORS:
Akram Sheriff
Rajiv Asati

ABSTRACT

Techniques are presented herein that support a remote observability capability (comprising remote visibility, monitoring, and troubleshooting) for Internet of things (IoT) devices that are connected behind an IoT gateway (by agnostic network connectivity) through a dynamically deployed Enterprise Agent that is able to monitor all of the underlying state changes (through a southbound interface in an operational technology (OT) protocol agnostic manner) in IoT devices and then relay the collected telemetry data to a cloud-based facility. Aspects of the presented techniques leverage an embedded Subscriber Identity Module (eSIM)-based Java Card facility to collect telemetry data through a metrics, events, logs, and traces (MELT) framework and an OpenTelemetry (OTel) application programming interface (API) along with a wireless modem. Further aspects of the presented techniques leverage an IoT SIM Applet For Secure End-2-End Communication (SAFE) agent.

DETAILED DESCRIPTION

A remote observability capability (comprising remote visibility, monitoring, and troubleshooting) is of paramount importance to enterprise customers that rely on operational technology (OT) solutions such as smart manufacturing, smart building facilities, etc. With such a capability, a customer (such as an enterprise, an Internet of things (IoT) provider, or an application provider) can triage a problem concerning an OT device that is deployed (to a fixed location or in motion) in a public environment or in a private facility without always sending a technician to the field. Not only does such a capability save time and money, but it also becomes the most convenient option during a crisis situation (e.g., during a pandemic) or during a period of human resource shortages (e.g., following a pandemic).

While the OpenTelemetry (OTel) concept and a metrics, events, logs, and traces (MELT) framework have gained quite a bit traction recently, OT devices and gateways do not always benefit from the same. One reason for this is that IoT devices do not have an OTel agent (or a custom agent or other native code instrumentation mechanism) for conducting IoT observability at the edge in an edge-native deployment. Further, in different vertical environments (such as manufacturing, utilities, and smart farming) customers tend to be reluctant to run custom instrumentation code on their IoT devices for gathering observability data according to the MELT framework.

The underlying reason for the above-described limitations is the security shortcomings of the involved IoT devices. IoT devices conventionally apply insufficient security mechanisms due to resource constraints, cost factors, and the complexity of building additional layers on top of existing protocols. Also, many manufacturers do not go beyond basic security functions (such as, for example, default credentials) which increases the vulnerability and threat posture of the devices.

It is important to note that the above-described lack of remote observability in an OT environment is not an unknown problem. While there have been some attempts to address that problem, the solutions thus far have been proprietary or they have introduced fragmentation in supporting devices from multiple vendors. Choosing from the many options and standards that are available may reduce interoperability, cause fragmentation, and limit the scalability of a security solution.

As one example, consider an IoT Subscriber Identity Module (SIM) Applet For Secure End-2-End Communication (SAFE) agent that may be employed in support of a remote troubleshooting requirement. Under such an arrangement, there are certain limitations that arise when an enterprise leverages this approach for remote troubleshooting.

The IoT SAFE paradigm is designed as a useful solution for personas such as an IoT service provider (to whose servers the IoT devices may safely connect) but not for an enterprise that builds and/or deploys solutions for their OT benefit. As a result, an enterprise is forced to procure various classes of IoT devices from different IoT providers and when the devices become operational the data from those devices may be processed and then used in the enterprise’s cloud. An enterprise may need to involve an IoT provider only when there is a technical problem with an IoT device that must be triaged by the device provider. In such a case, access by the provider to the device needs to be secure, time-bounded, and controlled. It is always possible that an IoT provider may work with multiple partner vendors, hence it is possible that multiple users may access the device.

An enterprise needs more control over securely onboarding the provider and effectively managing a troubleshooting session. It is quite common for an IoT device to operate multiple IoT applications coming from various providers, and it is also common for a solution to be used with different IoT devices. Accordingly, when a system is misbehaving an enterprise may need to isolate the problem to a specific vendor or involve all of the concerned parties more securely to triage the problem. Additionally, an enterprise needs observability data (including data access and exports, command usage, the number of authentic sessions, etc.) for the remote troubleshooting of IoT devices by the third party IoT providers.

As a result, an IoT SAFE-based approach by itself is not a viable solution for an enterprise for IoT deployments.

Techniques are presented herein that support a novel mechanism for meeting the above-described remote visibility, observability, and troubleshooting requirement of IoT devices that are connected behind an IoT gateway and which are controlled through a dynamic Enterprise Agent.

Aspects of the presented techniques leverage an embedded SIM (eSIM)-based Java Card facility to collect telemetry data through a MELT framework and an OTel application programming interface (API) along with a wireless modem (such as, for example, a SIM card in the case of a cellular long-term evolution (LTE) modem). Java Card technology offers a platform for developing smart card applications using the Java programming language, comprising a subset of the Java platform that is designed to run on resource-constrained embedded devices (such as smart cards and SIM cards) which typically have limited processing power, memory, and storage.

Among other things, the presented techniques encompasses a series of preparatory activities. Under one of those activities an enterprise may construct a Java Card-based application, which may be referred to herein as an Enterprise Agent, which may be installed in a SIM card. Such an application is meant to carry out two functionalities.

A first functionality encompasses processing and enforcing the instructions (i.e., the rules) that are provided by an enterprise for a remote troubleshooting session with an IoT provider. Those instructions (or rules) may include, but are not limited to, an application for which troubleshooting is allowed (since an IoT device can hold multiple applications), the number of simultaneous sessions that are allowed (since an IoT provider and their partners may access a device for troubleshooting, the number of sessions, the location of access, etc. needs tighter control), the use of SIM cryptography for a data payload (since the IoT SAFE standards suggest that for additional security SIM cryptography may be used on a data payload, an enterprise may enforce such a requirement on all third-party IoT provider access), and a time interval for access to a device (i.e., access to the device may be limited to just a specified time period).

A second functionality encompasses acting as a proxy channel between internal device middleware and an IoT SAFE agent. An IoT SAFE interface forms a lower layer in device middleware which invokes the IoT SAFE agent for Datagram Transport Layer Security (DTLS) session establishment. An Enterprise Agent may provide the wrapper implementation of such an interface, interpret the session establishment request, and then consult the rule set that was provided by the enterprise for allowing or discarding the request. If the request is to be allowed, then it may be forwarded to the IoT SAFE agent.

It is important to note that typically an IoT SAFE agent in a SIM may be accessed through the standard modem or module AT commands (as described in the 3rd Generation Partnership Project (3GPP) Technical Specification (TS) 27.007) which provide a means for forwarding a message to a SIM. In the same manner, an IoT SAFE message may be encapsulated within an AT+CSIM command and then exchanged directly between a SIM and a device application using an International Organization for Standardization (ISO) 7816 standard-based interface that is located in the modem or module. In the presented techniques, an Enterprise Agent may support and receive an AT command, processes the rule set, and accordingly forward the command to an IoT SAFE agent.

As part of an on-boarding process, an enterprise may deploy an Enterprise Agent in the SIM card of an IoT device and also provision the enterprise key and keying materials in the Enterprise Agent secured storage. That key is a root key that may be used to derive other keys at runtime. While the derived keys may be provisioned in IoT SAFE agent storage, the root key is fully protected inside of the Enterprise Agent storage.

Figure 1, below, presents elements of an illustrative embedded agent-based solution that is possible according to the techniques presented herein and which is reflective of the above discussion.

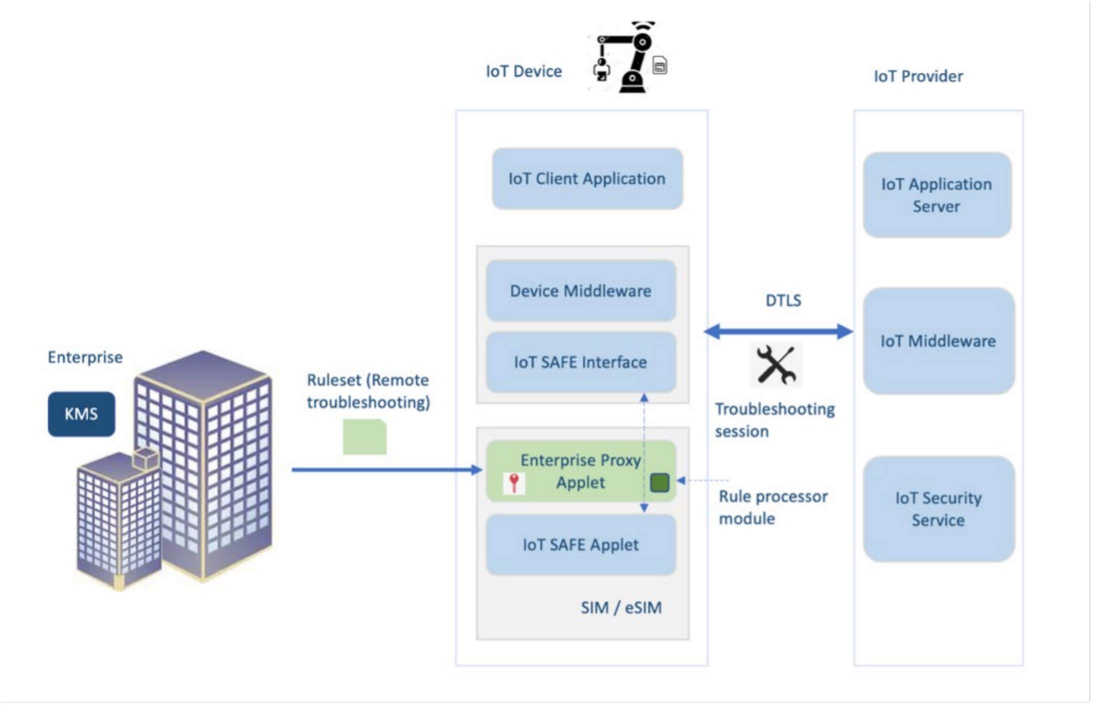

Figure 1: Illustrative Embedded Agent-based Solution

Among other things, the presented techniques encompass a controller agent-based IoT observability workflow under which an enterprise may employ a management system to monitor their IoT devices. Through such a system, if there is any malfunction then the enterprise may identify the problematic application.

For observability and troubleshooting support, the pre-shared keys that were described above may be employed. An enterprise system may request that an Enterprise Agent create a time-bounded troubleshooting key for an application. The Enterprise Agent may generate such a key from the enterprise root key and the associated keying materials and then share the generated troubleshooting key with the enterprise. Additionally, the troubleshooting key, along with any mapping information, may be stored in a vault area of an IoT SAFE agent. The enterprise may then contact an IoT provider to triage an issue and share the generated troubleshooting key with the provider through an out-of-bound secured channel.

Next, an enterprise may create a set of rules that may be used for the underlying troubleshooting session and then share that rule set with the Enterprise Agent. Such a rule set can comprise any set of instructions that are to be enforced during a troubleshooting session.

When the IoT provider initiates the troubleshooting session, the DTLS security handshake must be offloaded to the IoT SAFE agent according to the relevant standards and the incoming instructions may be proxied through an Enterprise Agent. As defined in the Global System for Mobile Communications Association (GSMA) standards, the IoT SAFE agent is discoverable in device middleware. Similarly, with an Enterprise Agent in place the presence of the same will be discoverable to device middleware and the Enterprise Agent may proxy the request to the IoT SAFE agent. The Enterprise Agent may consult the rule set to ensure that the DTLS session can be allowed and, accordingly, forward the request to the IoT SAFE agent. Under the presented techniques the DTLS session establishment will be taken care of by the IoT SAFE agent hence that activity works in a standard fashion.

It is important to note that if there are any deviations from the given rule set (for instance, if the allowed DTLS session is time-bounded and the incoming request is not specific to the given time limit) then the DTLS session establishment request may be discarded. Similarly, if only one DTLS session is allowed then further session establishment requests from an IoT provider may be discarded. Further, if there is any request level probing needed to ensure the access restrictions, then this is possible as the key is derived from the enterprise key.

Once the troubleshooting key expiration time arrives, the Enterprise Agent may dynamically remove the key from the IoT SAFE agent.

Figure 2, below, presents elements of a sequence diagram that encompasses aspects of the above discussion.

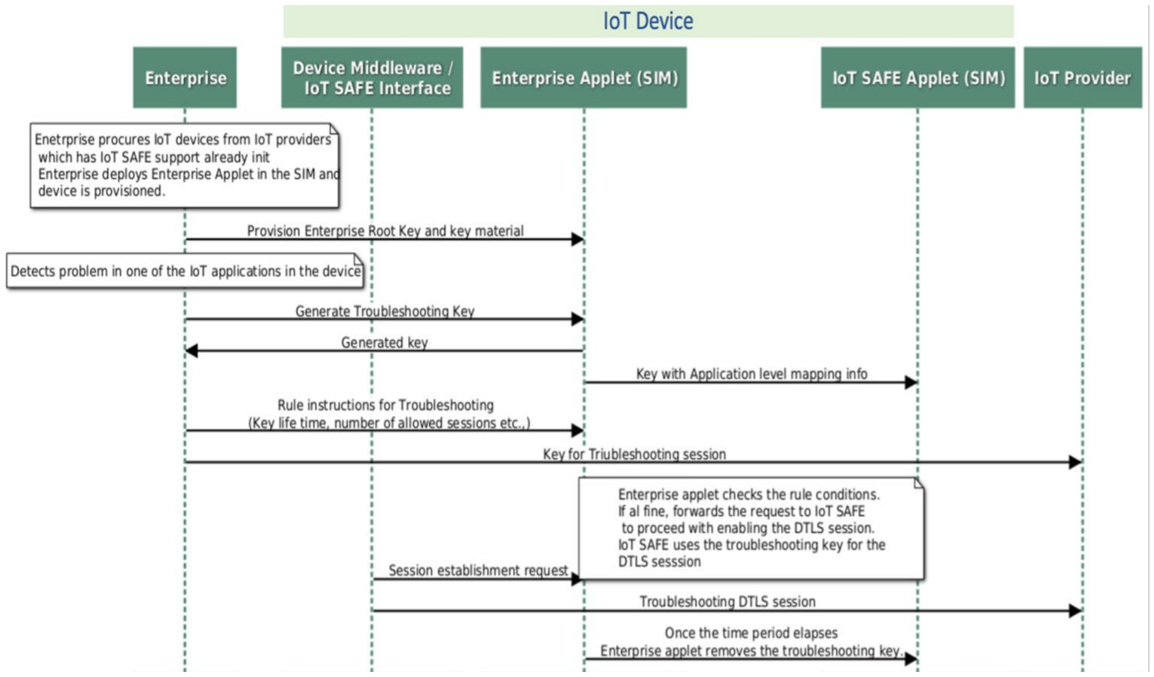
IoT Device
Enterprise
Device Middleware / IoT SAFE Interface
Enterprise Applet (SIM)
IoT SAFE Applet (SIM)
IoT Provider
Enetrprise procures IoT devices from IoT providers which has IoT SAFE support already init Enterprise deploys Enterprise Applet in the SIM and device is provisioned.
Provision Enterprise Root Key and key material
Detects problem in one of the IoT applications in the device
Generate Troubleshooting Key
Generated key
Key with Application level mapping info
Rule instructions for Troubleshooting (Key life time, number of allowed sessions etc.,)
Key for Triubleshooting session
Enterprise applet checks the rule conditions. If al fine, forwards the request to IoT SAFE to proceed with enabling the DTLS session. IoT SAFE uses the troubleshooting key for the DTLS sesssion
Session establishment request
Troubleshooting DTLS session
Once the time period elapses Enterprise applet removes the troubleshooting key

Figure 2: Exemplary Sequence Diagram

Figure 3, below, presents elements of an exemplary data collector telemetry architecture comprising integration with a cloud-based IoT services platform (which may be referred to herein as an IoTOD facility).

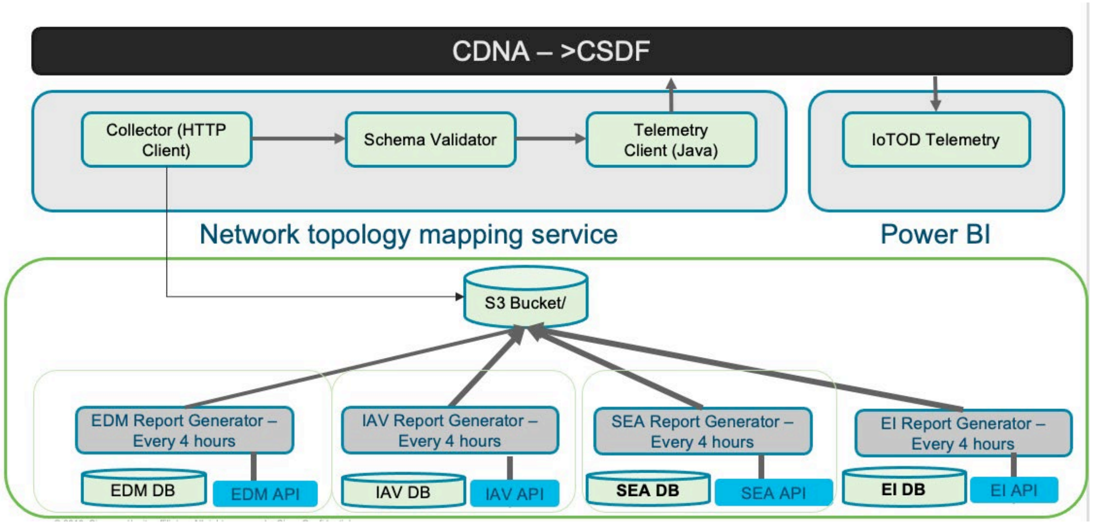
CDNA – >CSDF
Collector (HTTP Client)
Schema Validator
Telemetry Client (Java)
IoTOD Telemetry
Network topology mapping service
Power BI
S3 Bucket/
EDM Report Generator – Every 4 hours
IAV Report Generator – Every 4 hours
SEA Report Generator – Every 4 hours
EI Report Generator – Every 4 hours
EDM DB
EDM API
IAV DB
IAV API
SEA DB
SEA API
EI DB
EI API

Figure 3: Exemplary Data Collector Telemetry Architecture

Use of the techniques presented herein offers a number of advantages. First, the techniques are based on the IoT SAFE standard which is supported by eSIM and SIM card vendors, mobile service providers, module owners, and IoT providers. Consequently, there is no fragmentation or vendor lock-in. The novel Enterprise Agent (as described above) may be added to a SIM card and may provide a proxy layer for an IoT SAFE agent which is easy to manage and provides better agent- or controller-driven observability to enterprise and service provider domains.

Second, if a full-stack observability (FSO) approach is employed for observability detection then an IoTOD facility or a control center may dynamically provision an Enterprise Agent based on an enterprise-level rule or a service provider-hosted dynamic agent to provide a dynamic restriction policy and, at the same time, provide end-to-end IoT observability.

Third, the presented techniques enable any controller-based solution or IoT BU (IoTOD) controller and service provider (CC controller) customers to get their IoT observability data for doing observability root cause analysis (RCA) study and also can be monetized in a recurring software as a service (SaaS) business model.

The techniques presented herein may be employed in any number of contexts. As one possible example, consider a customer that has installed a vendor-supplied magnetic resonance imaging (MRI) machine which is connected through an out-of-band network interface, that is controlled through an eSIM, to manage different keys, certificates, and remote access workflows (such as collecting operational health data from the MRI machine) in different customer locations around the globe. Such an arrangement may be used in connection with predictive maintenance on the MRI machine for which a hosted application may be run on the machine. Additionally, the MRI machine may also have firmware from the embedded side of the device. With traditional OTel API-based techniques it is not possible to collect the firmware health data along with the MRI application state data. Under the presented techniques, an agent-based approach (comprising an eSIM and an IoT SAFE framework) provides a dedicated profile of the eSIM module with time-based rotating keys that are provisioned by a customer information technology (IT) network administrator. Such an approach allows the out-of-band LTE network connectivity link to collect all of the firmware health data along with the MRI application state data, through the dedicated eSIM profile, into the vendor's cloud application for operational equipment efficiency (OEE) assessment and predictive maintenance scheduling.

According the presented techniques, and in particular under the dynamic rule provisioning and management through an IoT SAFE agent and an eSIM that was described and illustrated above, an IoT application service and an IoT application client that are running in two private networks (which are managed by an enterprise's IT team) may securely communicate with each other, without exposing any port information on the Internet, through an end-to-end encrypted cloud relay link.

In summary, techniques have been presented herein that support a remote observability capability (comprising remote visibility, monitoring, and troubleshooting) for IoT devices that are connected behind an IoT gateway (by agnostic network connectivity) through a dynamically deployed Enterprise Agent that is able to monitor all of the underlying state changes (through a southbound interface in an OT protocol agnostic manner) in IoT devices and then relay the collected telemetry data to a cloud-based facility. Aspects of the presented techniques leverage an eSIM-based Java Card facility to collect telemetry data through a MELT framework and an OTel API along with a wireless modem. Further aspects of the presented techniques leverage an IoT SAFE agent.