Schneider Electric powers energy solutions on Azure Service Fabric

April 30, 2018

This post was authored by James Podgorski and Ed Price from Microsoft in conjunction with Noam Arbel, Stephen Berard, and Jarrett Campbell from Schneider Electric.

This post is part of a series (https://blogs.msdn.microsoft.com/azureservicefabric/tag/customer/) where we profile some of the customers who have been part of the Azure Service Fabric (https://azure.microsoft.com/en-us/services/service-fabric/) preview program over the last year and the applications they're now deploying into production. We look at why they chose to use Service Fabric and take a closer look at the design of their application, particularly focusing on a microservices design approach.

In this post, we're taking a look at Schneider Electric (http://www.schneider-electric.com/) and how they are using Service Fabric to provision customers with millions of connected field devices.

Schneider Electric is a global specialist in energy management and automation. They provide technological leadership in connectivity, sustainability, efficiency, and reliability in homes, cities, industries, buildings, and the cloud. The broader adoption of open connectivity standards and the technological advancements in data aggregation and data analysis have opened new opportunities for Schneider Electric to leverage their expertise in business process efficiencies to enable IOT-driven operational intelligence, energy management, and automation.

Schneider Electric delivers a large number of connected field devices such as sensors, drives, meters, programmable logic controllers, controls, and switchgear across a spectrum of business solutions including energy, home systems, automation and control, building management, power and cooling, data centers, electrical distribution, and more. They provide cloud services, middleware, remote monitoring, predictive analytics, and cloud analytics. This equates to millions of potentially connected devices with a diverse portfolio of systems and services for each of the vertical industrial markets that Schneider Electric serves.

Reducing cost and improving flexibility and extensibility

Schneider Electric found that they were repeatedly solving the same problems in machine-to-machine communication, data storage, mobile services and applications, and security. Instead, they wanted to focus innovation where it counts most, on services that are common and consistent for all Schneider Electric systems and devices, with inherent reuse and support for multiple tenants.

The challenge for Schneider Electric was to reduce the development and delivery cost for the service-level components common to each of their applications supporting their connected field devices. They were able to do this by building their EcoStruxture.io IoT-enabled platform, which provided the flexibility and extensibility for applications that serve their vertical markets.

For example, the platform must provide extensibility for executing custom business logic for those devices which cannot run such code locally due to constraints in either processing power or memory. The platform needs to be a broker of sorts for the devices when communication isn't available, or isn't viable because of expensive or slow links. To fulfill these challenges and more, Schneider Electric chose to use Azure Service Fabric to implement this part of their platform. Called Connect and Manage (CnM), it provides the flexibility and scalability to support millions of devices for telemetry, command/control, and other functionality with high reliability.

The core requirements of the Connect and Manage subsystem include:

- Registering and managing millions of devices
- Supporting multiple device devices communicating over numerous protocols (AMQP, LWM2M, MQTT)
- Providing multi-tenancy

- Querying device state and executing commands on a device
- Supporting a variety of connectivity models included devices that are intermittently connected
- Enabling per-device type pluggable logic
- Limiting overhead and maintaining low latency

Why did Schneider Electric choose Service Fabric?

Schneider Electric chose to use Service Fabric for CnM within EcoStruxture.io for a number of reasons. First, they saw the actor programming model as a natural way to represent devices. Second, they liked that Service Fabric ensured service resiliency and durability with a low cost in development and that it provided the ability to scale based on demand. Third, they believed that stateful services could ease development by providing a close connection between the logic and data, essentially eliminating the need for a backend data tier and allowing Schneider Electric to focus on adding business value rather than spending time and money on building a framework. Finally, Service Fabric provided Schneider Electric with a simplified operational life cycle including continuous deployment, elaborate configuration, rolling upgrades and package versioning. For a high-level look into why Schneider Electric chose Service Fabric and Azure IoT (https://azure.microsoft.com/en-us/services/iot-hub/) , see the video, Schneider Electric Leverages Microsoft Azure to Deliver IoT-Enabled Digital Services (https://www.youtube.com/watch?v=-BjV5Xf2QK4) .

Actor programming model

Schneider Electric needed a way to represent various device scenarios. They chose to use Service Fabric's reliable actors framework (https://azure.microsoft.com/en-us/documentation/articles/service-fabric-reliable-actors-introduction/) and created a model based on logical device actors, which expose interfaces to interact with applications, and physical device actors, which interact with the actual device. The various relationships are captured in Figure 1.

" Using Azure IoT with Service Fabric provides us with an open, flexible, and highly scalable platform. We can now talk to projects that might have a thousand devices or a million devices, and we're comfortable that the cloud technology can scale elastically to support those situations."

—John Brodeur: IoT Solutions Architect
Schneider Electric

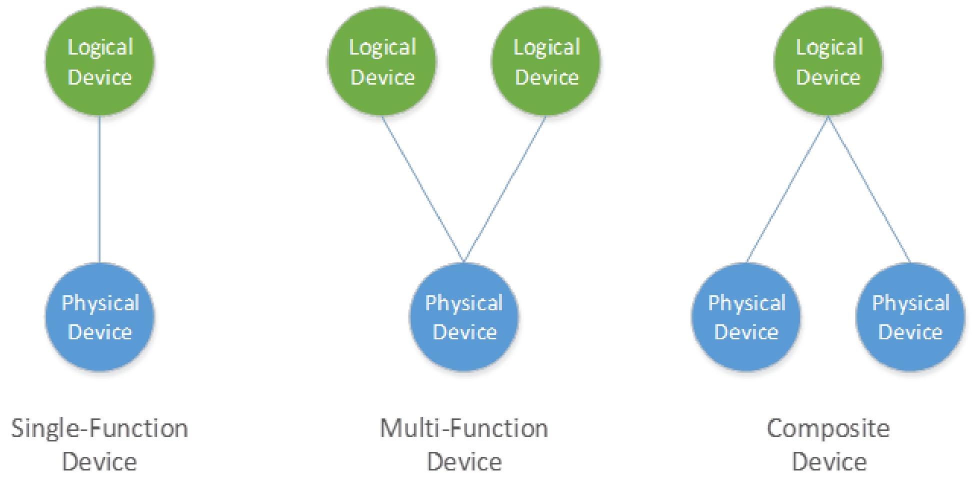

Figure 1: Connect and Manage model for devices

Actors in a pluggable way

The CnM capability abstracts the details of Service Fabric from the device manufacturers and requires that they implement two separate interfaces: the logical interface and a physical interface. Given the known interfaces, CnM can call into the device-specific implementation to interact with the appropriate device, leaving the details to the manufacturer. CnM achieves this by creating a common wrapping actor that loads the device-specific code to manage, query, control, and receive telemetry from said devices, all of which can be updated dynamically at run-time.

Service resiliency and durability

Schneider Electric used Service Fabric to build out an infrastructure with high service reliability and durability. They benefited from the agile development and delivery of the Service Fabric platform while concentrating on adding value to their underlying customers. Service Fabric reliable actors are automatically made durable by the platform with built-in partitioning, balancing, and scale. Building such capabilities in a traditional three-tier architecture with service resiliency, state durability, and partition-aware load balancing would have taken significant time and expense for development and operations teams.

Key reasons for Schneider Electric choosing the Service Fabric actor programming model were:

- Intuitive virtual actor API
- Built-in service discovery - clients don't need to figure out where a given actor is because Service Fabric does the addressing
- Natural state management - the platform does the job of reliably persisting the state
- Partitioned services by default - scalability is built in

Architectural Building Blocks

The CnM subsystem is composed of the following conceptual building blocks as shown in Figure 2. To simplify the discussion, Schneider Electric considers four upper level capabilities that make up CnM: Device Lifecycle, Logic & State, Device Hub and Device Gateway capabilities.

Device lifecycle deals with device registration, updates to device configuration, firmware upgrades and standard device management functionality. The device life cycle building block is accessed via a REST API and will save state, kick off the appropriate workflow executors, initiate control messages to the devices and update the configured topology.

Logic & State is where custom plugin logic is run, where state is cached and where data and control is requested from the devices. This conceptual building block is where the reliable actors reside and stateful topology information is kept.

The device hub building block is used to call the related gateway services which communicate with the actual device. Today there are two supported device gateways, one that supports OMA’s Lightweight M2M protocol (http://openmobilealliance.org/about-oma/work-program/m2m-enablers/) (LWM2M) and the other for Azure’s IoT Hub. It’s the device hub’s responsibility to know which device gateway to use when communicating with a specific device and which reliable actor, representing the device, needs to be called when a message is received from a device. Azure Service Bus (https://azure.microsoft.com/en-us/services/service-bus/) topics are used to receive requests from the reliable actors, such as “get the current temperature from a device”. Messages from devices manifest themselves as messages on an Azure Event Hub (https://azure.microsoft.com/en-us/services/event-hubs/) which are retrieved and forwarded to the appropriate reliable actor for processing by custom plugin logic.

Currently the device gateway supports two communications paths using similar, but different models. The first path supports the LWM2M protocol, using a service running on a virtual machine while the second path supports communication through Azure's IoT Hub. This latter path provides protocol support for AMQP, MQTT, HTTP, and is extendable via protocol gateways to support other protocols as needed.

" Service Fabric allowed us to focus on the key features of EcoStruxure.io, rather than building a distributed microservices architecture"

—Stephen Berard: Chief Architect - Digital Services Platform
Schneider Electric

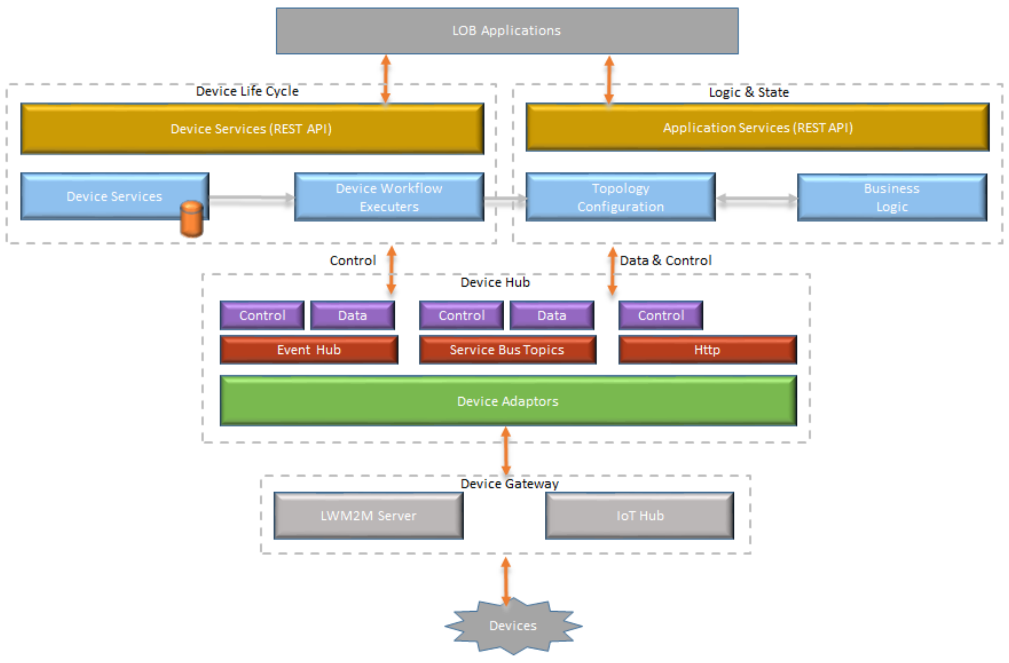

Figure 2: CnM conceptual architecture

Application design

The CnM capability is composed of a number of Service Fabric services that provide device management, topology configuration, device enrollment, client notifications, cloud device gateway/adaptors, device state, and custom pluggable logic. Each service can be independently versioned and upgraded.

The diagram below shows the various Azure Service Fabric and standard Azure services that make up CnM for the management and operation of devices. The application architecture diagram in Figure 3 is loosely grouped in the same way as the conceptual architecture shown in Figure 2, with subtle variances to simplify the layout of the services. A service type legend is provided with color coding to illustrate the types of services that make up the CnM. For instance, Schneider Electric built stateless Service Fabric Services for the API Gateway, the Ingress service and for the Device Adaptors, as denoted by the green service clouds.

“ We’re using Service Fabric in order to better process data, provide device logic, and provide our own device-connectivity framework.”

—Michael Mackenzie: Vice President of IoT Platform Delivery
Schneider Electric

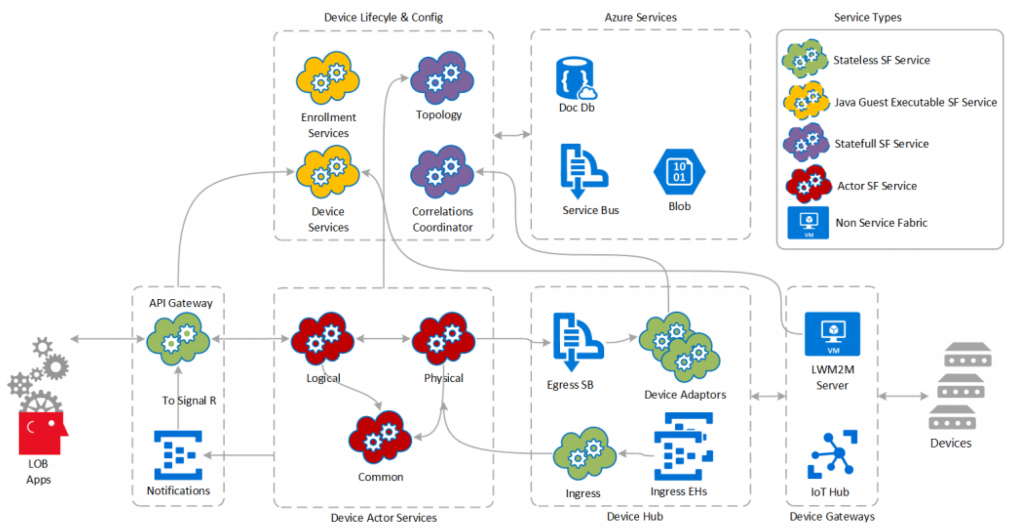

Figure 3: Schneider Electric microservices architecture using Azure Service Fabric

Let's take a detailed look at some of the key services:

API Gateway

The API gateway provides a RESTful interface between applications and devices in the CnM application and supports asynchronous notifications via ASP.NET SignalR. The API Gateway was built as a Service Fabric stateless service using OWIN to expose a Web API and SignalR. Actors can post messages to the Notifications Event Hub, which are then sent to any listeners via SignalR. For example, the application makes an inquiry about the temperature of a device, which happens to be offline. The application request is queued at the logical device actor and when the actor is back online it sends telemetry data to the CnM service, where placement in the Notifications Hub makes its way back to the application via the REST API Service and SignalR.

Device Actors Services

Schneider Electric designed its Device Actors Service to hold both logical and physical stateful Service Fabric device actors. The code for the device actors is registered by the Device Life Cycle & Configuration functional component and made available in a pluggable manner when a new device (actor) is created. The pluggable actor logic can be changed independently when new functionality is needed, or when a device is upgraded with new firmware, without any system downtime, and without the need for an upgrade of the services.

Logical device actors have both business logic and state. They respond to application requests for device state, device commands, and device push notifications. Logical device actors are ideally designed to hold quickly changing data, such as temperature, voltage, and so on.

Like logical device actors, physical device actors have both business logic and state. A physical device actor's primary responsibility is to receive telemetry from devices and send commands to the devices. The physical device actor translates the device-specific data format (integer, byte array, json, etc.). For example, a physical device actor can receive a byte array sent from the physical device, decodes the array, and forwards a higher-level message to the logical device actor, which is able to take the appropriate action based on the message.

The common actor service is used to provide common functionality to both the physical and logical device actors.

Device Hub Services

The Device Hub functional block is a messaging and routing layer that is composed of an Ingress Service, Device Adaptor services, a number of Egress Service Bus Topics and an Ingress Event Hub.

Physical device actors send control messages to devices. There are a series of topics, one per Device Adapter. The Device Adapter service retrieves messages from an egress topic and either sends the message to the appropriate protocol gateway, either LWM2M or Azure IoT Hub as described previously.

On the ingress side, there is a stateless Service Fabric Ingress service which reads device telemetry messages from the Ingress Event Hub and sends messages to the appropriate physical device actor for processing. The Ingress Azure Event Hub currently consists of two Azure Event Hubs, one for each Device Adapter.

Device Gateway

The LWM2M service currently runs as a classic Azure Cloud Service (https://azure.microsoft.com/en-us/services/cloud-services/) VM. Azure IoT Hub is deployed to work with AMQP devices and other protocols in the future through the use of Azure IoT Hub's protocol gateway support.

Device Life Cycle and Configuration

The device life cycle and configuration services are comprised of both stateful and stateless Service Fabric services together with guest executable services (https://azure.microsoft.com/en-us/documentation/articles/service-fabric-deploy-existing-app/) running Java applications. Schneider Electric migrated two existing Java applications for device services and enrollment management to be run in Service Fabric. These services provide the ability to register devices with the system, to onboard custom device logic plugins, and configure the cloud gateway/adaptor for new device types. Schneider Electric chose to build the device Topology and Correlations services using stateful Service Fabric services. Orchestration of the device registration and device management capabilities are executed through the combination of the Azure Service Bus for messaging, Azure Blob Storage (https://azure.microsoft.com/en-us/services/storage/blobs/) for firmware images and Azure DocumentDB for device registration specifics.

Summary

Schneider Electric wanted to reduce the development and delivery cost for the service-level components common to each of their systems, while providing flexibility and extensibility for applications that serve their markets. To achieve this, Schneider Electric leveraged Azure Service Fabric to build their EcoStruxure.io platform, which is both flexible and scalable. They created a common and consistent platform for connecting millions of devices to Schneider Electric applications.

“ Service Fabric reliable actors enabled us to build a scalable solution for implementing our device logic within EcoStruxure.io.”

—Stephen Berard: Chief Architect - Digital Services Platform
Schneider Electric