更に上のクオリティ
更に上のサービス!

問題集

ITEXAMPASS

https://www.itexampass.jp

１年で無料進級することに提供する

Exam : AZ-220

Title : Microsoft Azure IoT Developer

Version : DEMO

1. Topic 1, Contoso

Case Study
This is a case study. **Case studies are not timed separately. You can use as much exam time as you would like to complete each case**. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.

To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other question on this case study.

At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next sections of the exam. After you begin a new section, you cannot return to this section.

To start the case study
To display the first question on this case study, click the **Next** button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment, and problem statements. If the case study has an **All Information** tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the **Question** button to return to the question.

Existing Environment. Current State of Development
Contoso produces a set of Bluetooth sensors that read the temperature and humidity. The sensors connect to IoT gateway devices that relay the data.

All the IoT gateway devices connect to an Azure IoT hub named iothub1.

Existing Environment. Device Twin
You plan to implement device twins by using the following JSON sample.

```
{
  "deviceId": "device_n",
  "etag": "AAAAAAAAAAQ=",
  "deviceEtag": "NDcwMTU4Mzk=",
  "status": "enabled",
  "statusUpdateTime": "0001-01-01T00:00:00Z",
  "connectionState": "Disconnected",
  "lastActivityTime": "0001-01-01T00:00:00Z",
  "cloudToDeviceMessageCount": 0,
  "authenticationType": "sas",
  "x509Thumbprint": {
    "primaryThumbprint": null,
    "secondaryThumbprint": null
  },
  "version": 11,
  "properties": {
    "desired": {
      "fanSpeed": 70,
      "$metadata": {
        "$lastUpdated": "2019-10-16T09:43:42.2944169Z",
        "$lastUpdatedVersion": 4,
        "fanSpeed": {
          "$lastUpdated": "2019-10-16T09:43:42.2944169Z",
          "$lastUpdatedVersion": 4
        }
      },
      "$version": 4
    },
    "reported": {
      "fanSpeed": 80,
      "metadata": {
        "$lastUpdated": "2019-10-16T09:43:42.4035171Z",
        "fanSpeed": {
        "$lastUpdated": "2019-10-16T09:43:42.4035171Z"
        }
      },
      "$version": 7
    }
  },
  "capabilities": {
    "iotEdge": false
  }
}
```


Existing Environment. Azure Stream Analytics

Each room will have between three to five sensors that will generate readings that are sent to a single IoT gateway device. The IoT gateway device will forward all the readings to iothub1 at intervals of

between 10 and 60 seconds.

You plan to use a gateway pattern so that each IoT gateway device will have its own IoT Hub device identity.

You draft the following query, which is missing the GROUP BY clause.

```
SELECT
AVG(temperature),
System.TimeStamp() AS AsaTime
FROM
Iothub
```

You plan to use a 30-second period to calculate the average temperature reading of the sensors.

You plan to minimize latency between the condition reported by the sensors and the corresponding alert issued by the Stream Analytics job.

Existing Environment. Device Messages
The IoT gateway devices will send messages that contain the following JSON data whenever the temperature exceeds a specified threshold.

```
{
  "event": {
    "payload":"Temperature=26.23 Humidity = 78.70597746416186 Button = 0",
    "properties": {
      "application": {
        "level": "critical"
      }
    }
  }
}
```

The level property will be used to route the messages to an Azure Service Bus queue endpoint named criticalep.

Existing Environment. Issues
You discover connectivity issues between the IoT gateway devices and iothub1, which cause IoT devices to lose connectivity and messages.

Requirements. Planning Changes
Contoso plans to make the following changes:
- Use Stream Analytics to process and view data.
- Use Azure Time Series Insights to visualize data.
- Implement a system to sync device statuses and required settings.
- Add extra information to messages by using message enrichment.

- Create a notification system to send an alert if a condition exceeds a specified threshold.
- Implement a system to identify what causes the intermittent connection issues and lost messages.

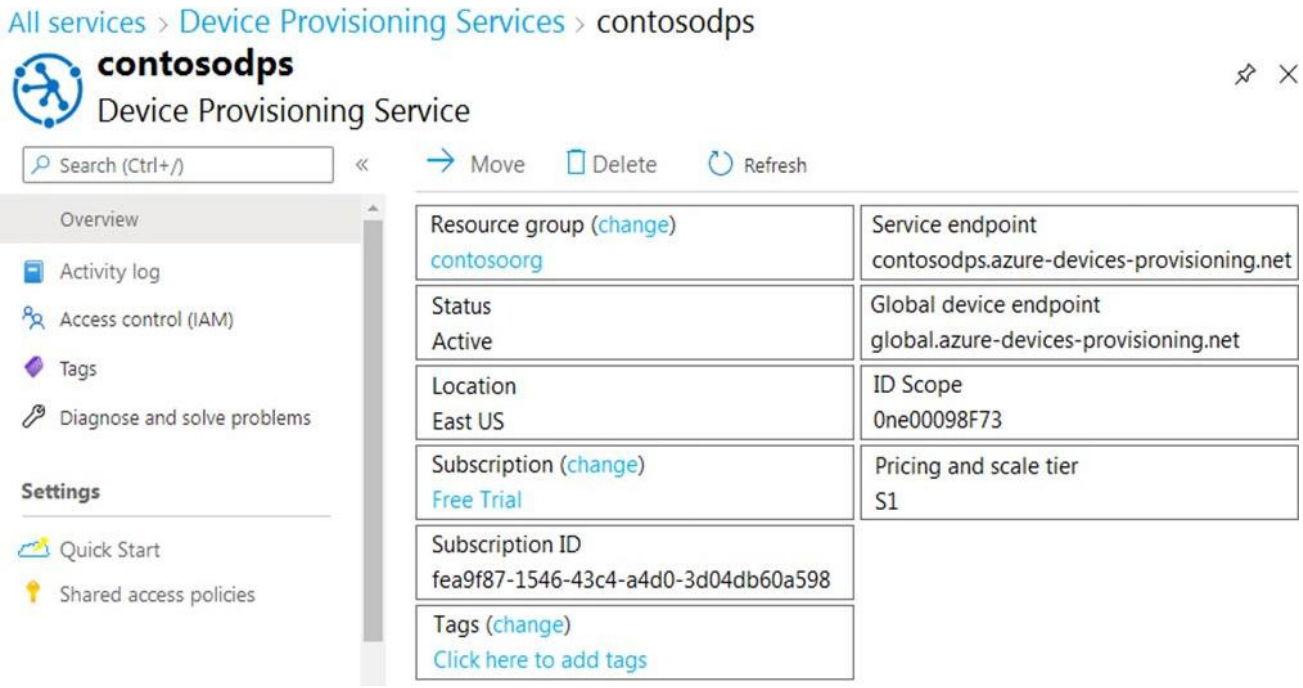
Requirements. Technical Requirements

Contoso must meet the following requirements:
- Use the built-in functions of IoT Hub whenever possible.
- Minimize hardware and software costs whenever possible.
- Minimize administrative effort to provision devices at scale.
- Implement a system to trace message flow to and from iothub1.
- Minimize the amount of custom coding required to implement the planned changes.
- Prevent read operations from being negatively affected when you implement additional services.

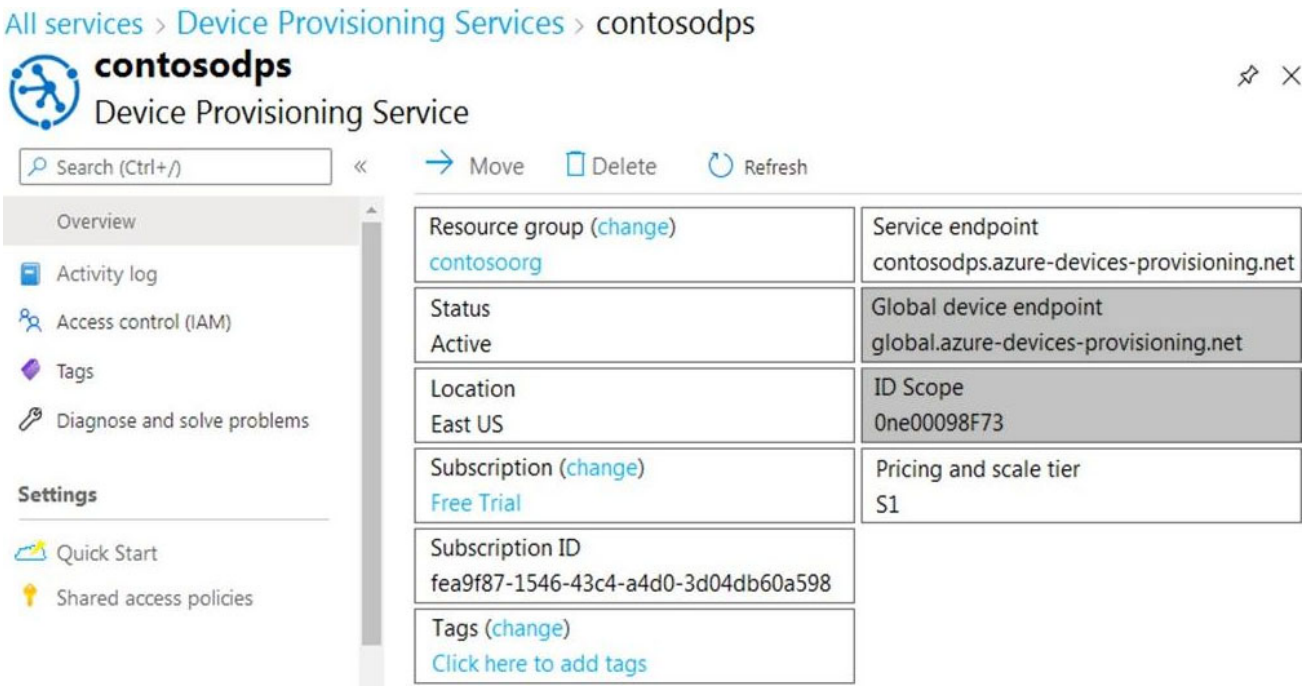
HOTSPOT

You are writing code to provision IoT devices by using the Device Provisioning Service.

Which two details from the Overview blade of the Device Provisioning Service are required to provision a new IoT client device? To answer, select the appropriate detail in the answer area. NOTE: Each correct selection is worth one point.



Answer:



Explanation:
Box 1: ID Scope
In the Azure portal, select the Overview blade for your Device Provisioning service and copy the ID Scope value. The ID Scope is generated by the service and guarantees uniqueness. It is immutable and used to uniquely identify the registration IDs.
Box 2: Global device endpoint
The global_prov_uri variable, which allows the IoT Hub client registration API IoTHubClient_LL_CreateFromDeviceAuth to connect with the designated Device Provisioning Service instance.
Example code:
static const char* global_prov_uri = "global.azure-devices-provisioning.net"; static const char* id_scope = "[ID Scope]";

2.How should you complete the GROUP BY clause to meet the Streaming Analytics requirements?
A. GROUP BY HoppingWindow(Second, 60, 30)
B. GROUP BY TumblingWindow(Second, 30)
C. GROUP BY SlidingWindow(Second, 30)
D. GROUP BY SessionWindow(Second, 30, 60)
Answer: B
Explanation:
Scenario: You plan to use a 30-second period to calculate the average temperature reading of the sensors.
Tumbling window functions are used to segment a data stream into distinct time segments and perform a function against them, such as the example below. The key differentiators of a Tumbling window are that they repeat, do not overlap, and an event cannot belong to more than one tumbling window.
InAnswers:
A: Hopping window functions hop forward in time by a fixed period. It may be easy to think of them as

Tumbling windows that can overlap, so events can belong to more than one Hopping window result set.
Reference: https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-window-functions

3.You plan to deploy Azure Time Series Insights.
What should you create on iothub1 before you deploy Time Series Insights?
A. a new message route
B. a new consumer group
C. a new shared access policy
D. an IP filter rule
Answer: B
Explanation:
Create a dedicated consumer group in the IoT hub for the Time Series Insights environment to consume from. Each Time Series Insights event source must have its own dedicated consumer group that isn't shared with any other consumer. If multiple readers consume events from the same consumer group, all readers are likely to exhibit failures.
Reference: https://docs.microsoft.com/en-us/azure/time-series-insights/time-series-insights-how-to-add-an-event-source- iothub

4.What should you do to identify the cause of the connectivity issues?
A. Send cloud-to-device messages to the IoT devices.
B. Use the heartbeat pattern to send messages from the IoT devices to iothub1.
C. Monitor the connection status of the device twin by using an Azure function.
D. Enable the collection of the Connections diagnostics logs and set up alerts for the connected devices count metric.
Answer: D
Explanation:
Scenario: You discover connectivity issues between the IoT gateway devices and iothub1, which cause IoT devices to lose connectivity and messages.
To log device connection events and errors, turn on diagnostics for IoT Hub. We recommend turning on these logs as early as possible, because if diagnostic logs aren't enabled, when device disconnects occur, you won't have any information to troubleshoot
the problem with.
Step 1:
1.Sign in to the Azure portal.
2.Browse to your IoT hub.
3.Select Diagnostics settings.
4.Select Turn on diagnostics.
5.Enable Connections logs to be collected.
6.For easier analysis, turn on Send to Log Analytics (see pricing).
Step 2:
Set up alerts for device disconnect at scale
To get alerts when devices disconnect, configure alerts on the Connected devices (preview) metric.
Reference: https://docs.microsoft.com/bs-cyrl-ba/azure/iot-hub/iot-hub-troubleshoot-connectivity

5.You need to enable telemetry message tracing through the entire IoT solution.
What should you do?
A. Monitor device lifecycle events.
B. Upload IoT device logs by using the File upload feature.
C. Enable the DeviceTelemetry diagnostic log and stream the log data to an Azure event hub.
D. Implement distributed tracing.
Answer: D
Explanation:
IoT Hub is one of the first Azure services to support distributed tracing. As more Azure services support distributed tracing, you'll be able trace IoT messages throughout the Azure services involved in your solution.
Note:
Enabling distributed tracing for IoT Hub gives you the ability to:
Precisely monitor the flow of each message through IoT Hub using trace context. This trace context includes correlation IDs that allow you to correlate events from one component with events from another component. It can be applied for a subset or all IoT device messages using device twin.
Automatically log the trace context to Azure Monitor diagnostic logs.
Measure and understand message flow and latency from devices to IoT Hub and routing endpoints. Start considering how you want to implement distributed tracing for the non-Azure services in your IoT solution.
Reference: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-distributed-tracing