


# **Nanoservice-enabled event-driven decision support REST API**

## **Demo (Hyponatremia)**

For more information about Nanoservice, please visit <https://csml.uconn.edu/>

Step 1: Go to <http://csmlab7.uconn.edu/swagger/>

 Select a spec NanoService API V1

# NanoService API v1

[/swagger/v1/swagger.json](#)

A simple example NanoService API

[Terms of service](#)

[Yong-Jun Shin - Website](#)

[Send email to Yong-Jun Shin](#)

[Open source GitHub Project \(MIT License\)](#)

## Actors

POST

**/createActor** Creates an actor.

POST

**/addVariable** Adds a variable to an actor.

POST

**/getValue** Gets the value of a variable.

POST


**/setValue** Sets the value of a variable.

# Step 2 [createActor]: Click

POST

/createActor

Creates an actor.

 swagger

Select a spec

NanoService API V1

## NanoService API <sup>v1</sup>

[/swagger/v1/swagger.json](#)

A simple example NanoService API

[Terms of service](#)

[Yong-Jun Shin - Website](#)

[Send email to Yong-Jun Shin](#)

[Open source GitHub Project \(MIT License\)](#)

Actors

POST

/createActor

Creates an actor.

POST

/addVariable

Adds a variable to an actor.

POST

/getValue

Gets the value of a variable.

POST

/setValue

Sets the value of a variable.

## Step 3 [createActor]: Follow the instruction and create a patient (actor) with a unique ID (actorId).

Sample request body (requestBody):

```
{  
  "actorId": "patient032904475"  
}
```

- "actionId": string

To try this service:

1. Click [Try it out] button (white).
2. Type your request body into "Example Vaule | Model" textbox (white). A sample request body is shown above.
3. Click [Execute] button (blue).
4. Check "Response body" (ignore "Code" for now). If you see "Actor created", your request is processed successfully. Otherwise, you will get an error message.

For more information about Service Fabric Actors, please see:

<https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-reliable-actors-introduction>

Server response

Code

Details

200

Response body

"Actor created"

## Service Fabric Actors

<https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-reliable-actors-introduction>

**Step 4 [addVariable]:** Add a variable (“bloodSodium”) to the previously created patient actor. Let the variable value be “unknown” initially

**POST**

**/addVariable** Adds a variable to an actor.

Sample request body (requestBody):

```
{
  "actorId": "patient032904475",
  "variable": "bloodSodium",
  "value": "unknown"
}
```

- "actionId": string
- "variable": string
- "value": string, int, float, bool

To try this service:

1. Click [Try it out] button (white).
2. Type your request body into “Example Vaule | Model” textbox (white). A sample request body is shown above.
3. Click [Execute] button (blue).
4. Check “Response body” (ignore “Code” for now). If you see "Variable added", your request is processed successfully. Otherwise, you will get an error message.

**Response body**

**"Variable added"**

## Step 5 [getValue]: Get the value of the variable “bloodSodium” which is “unknown”.

POST

/getValue Gets the value of a variable.

Sample request body (requestBody):

```
{
  "actorId": "patient032904475",
  "variable": "bloodSodium"
}
```

- "actionId": string
- "variable": string

To try this service:

1. Click [Try it out] button (white).
2. Type your request body into “Example Vaule | Model” textbox (white). A sample request body is shown above.
3. Click [Execute] button (blue).
4. Check “Response body” (ignore “Code” for now). If you see the value retrieved, your request is processed successfully. Otherwise, you will get an error message.

Response body

"unknown"

**Step 6 [setValue]:** Set the value of the variable “bloodSodium” to 109. The unit is mEq/L and not included.

**POST**

**/setValue** Sets the value of a variable.

Sample request body (requestBody):

```
{
  "actorId": "patient032904475",
  "variable": "bloodSodium",
  "value": 109
}
```

- "actionId": string
- "variable": string
- "value": string, int, float, bool

To try this service:

1. Click [Try it out] button (white).
2. Type your request body into “Example Vaule | Model” textbox (white). A sample request body is shown above.
3. Click [Execute] button (blue).
4. Check “Response body” (ignore “Code” for now). If you see "Value set", your request is processed successfully. Otherwise, you will get an error message.

Response body

**"Value set"**

Note this new value is also published to **Azure Event Grid**.

## Step 7 [getValue]: Get the value of the variable “bloodSodium” which is now changed to 109.

POST

**/getValue** Gets the value of a variable.

Sample request body (requestBody):

```
{
  "actorId": "patient032904475",
  "variable": "bloodSodium"
}
```

- "actionId": string
- "variable": string

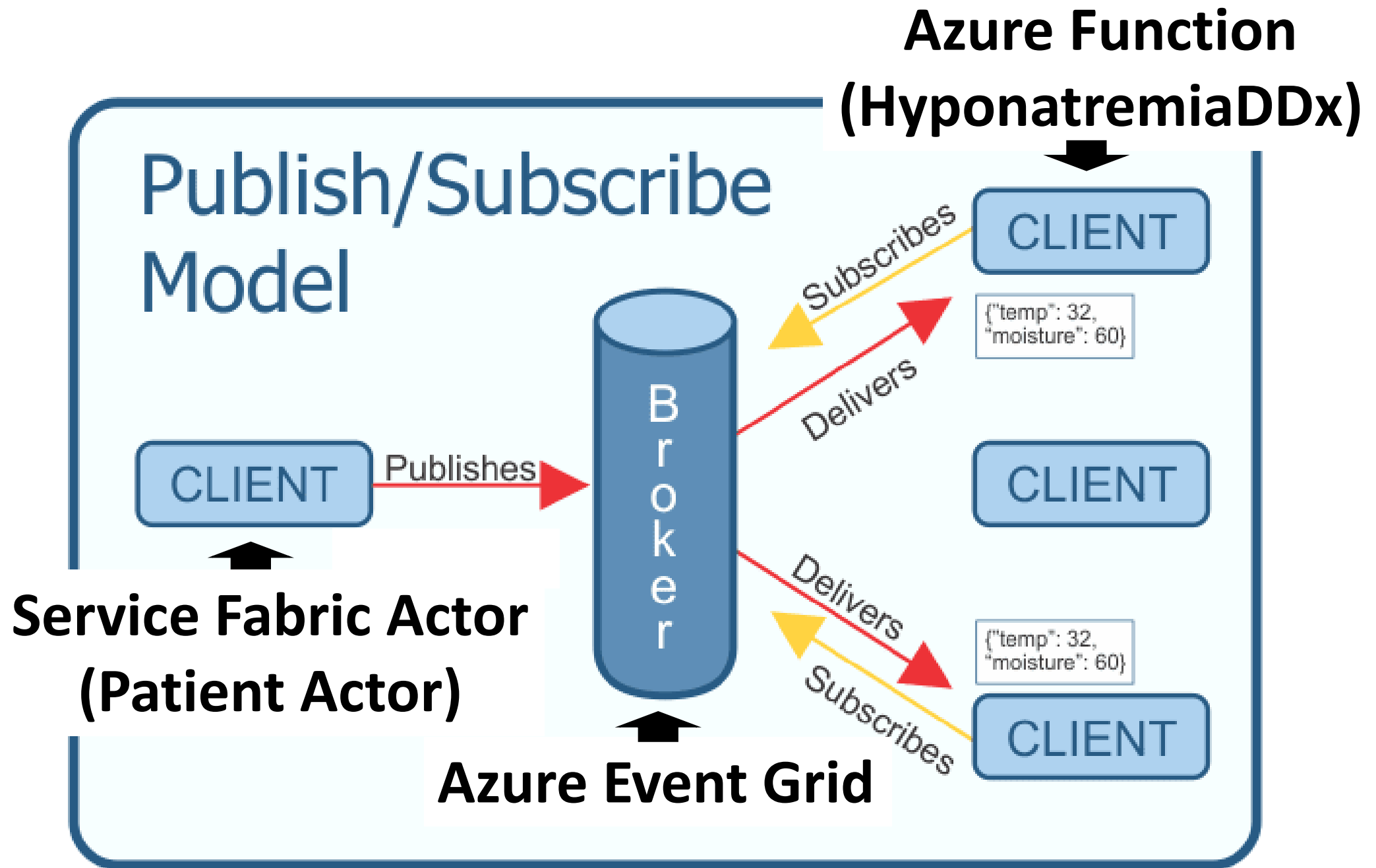
To try this service:

1. Click [Try it out] button (white).
2. Type your request body into “Example Vaule | Model” textbox (white). A sample request body is shown above.
3. Click [Execute] button (blue).
4. Check “Response body” (ignore “Code” for now). If you see the value retrieved, your request is processed successfully. Otherwise, you will get an error message.

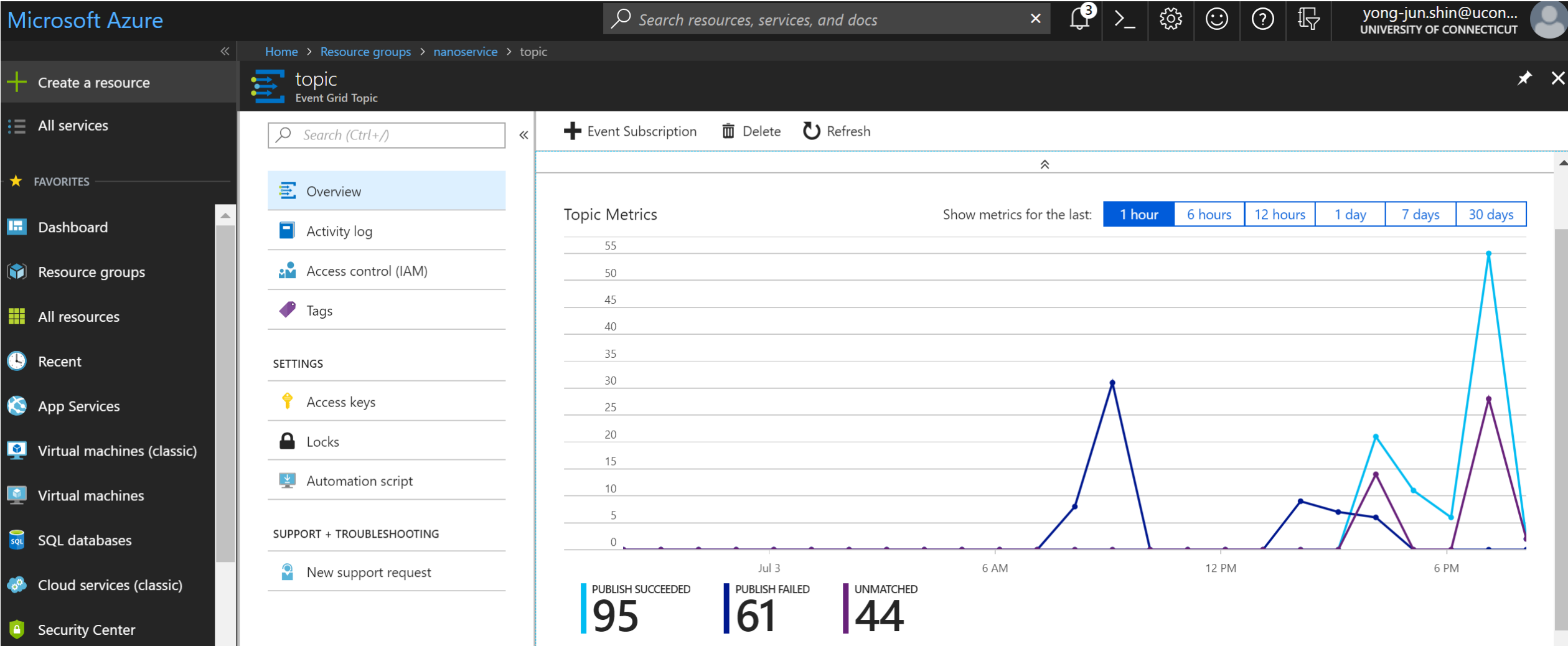
Response body

109





# Azure Event Grid <https://azure.microsoft.com/en-us/services/event-grid/>



# Code for publishing an event to Azure Event Grid

```
public async Task<string> PublishToAzureEventGridAsync(JObject data)
{
    string AzureEventGridTopicEndPoint = "https://topic.eastus-1.eventgrid.azure.net/api/events?api-version=2018-01-01";
    string AzureEventGridTopicAccessKey = "9UGRYFbXX3Pqr8yTp2vvhgvNBr8H00HSWza/PMdxu/0=";
    string uri = AzureEventGridTopicEndPoint;
    string topicSubject = (string)data.SelectToken("variable");

    // Event data schema (Azure Event Grid)
    // https://docs.microsoft.com/en-us/azure/event-grid/post-to-custom-topic#event-data

    dynamic requestBody = new ExpandoObject();
    requestBody.id = "notSet";
    requestBody.eventType = "notSet";
    requestBody.subject = topicSubject; // e.g., bloodSodium
    requestBody.eventTime = DateTime.Now;
    requestBody.data = data;
    requestBody.dataVersion = "v1";

    List<dynamic> requestBodyArray = new List<dynamic>();
    requestBodyArray.Add(requestBody);

    HttpClient client = new HttpClient();
    client.DefaultRequestHeaders.Add("aeg-sas-key", AzureEventGridTopicAccessKey);
    var request = new HttpRequestMessage(HttpMethod.Post, uri);
    string jsonRequestBody = JsonConvert.SerializeObject(requestBodyArray);
    request.Content = new StringContent(jsonRequestBody, Encoding.UTF8, "application/json");
    var response = await client.SendAsync(request);
    string stringResponseContent = await response.Content.ReadAsStringAsync();
    return stringResponseContent;
}
```

Source code available at

<https://github.com/yshin1209/Nanoservice>

# Azure Function <https://azure.microsoft.com/en-us/services/functions/>

The screenshot displays the Azure Functions portal interface. At the top, the breadcrumb navigation shows 'Home > All resources > Nanoservices - HyponatremiaDDx'. The main header area contains the title 'Nanoservices - HyponatremiaDDx' and 'Function Apps'. Below this, there's a search bar with 'Nanoservices' and a red 'X' icon. The left sidebar shows a tree view with 'Function Apps', 'Nanoservices', 'Functions', 'DecideHyponatremia', 'Integrate', 'Manage', 'Monitor', 'HyponatremiaDDx', 'Proxies', and 'Slots (preview)'. The main content area shows the code editor for 'run.csx'. The code is as follows:

```
1 #r "Newtonsoft.Json"
2
3 using Newtonsoft.Json;
4 using Newtonsoft.Json.Linq;
5 using System.Net;
6 using System.Dynamic;
7 using System.Net.Http;
8 using System.Text;
9
10 public async static void Run(JObject eventGridEvent, TraceWriter log)
11 {
12     log.Info(eventGridEvent.ToString(Formatting.Indented));
13     var jsonEventGridEvent = eventGridEvent.ToString();
14     JObject jobj = JObject.Parse (jsonEventGridEvent);
15     double bloodSodiumValue = (double)jobj["data"]["value"];
16     string actorId = (string)jobj["data"]["actorId"];
17     //log.Info("bloodSodiumValue: " + bloodSodiumValue);
18     //log.Info("(string)jobj[actorId]: " + actorId);
19     string decision = "unknown";
20     if (bloodSodiumValue < 135) {decision = "true";}
21     if (bloodSodiumValue >= 135) {decision = "false";}
22     HttpClient client = new HttpClient();
23     string addVariableUri = "http://csm1lab7.uconn.edu/addVariable";
24     var firstRequest = new HttpRequestMessage(HttpMethod.Post, addVariableUri);
25     dynamic firstRequestBody = new ExpandoObject();
26     firstRequestBody.actorId = actorId;
```

Below the code editor, there are tabs for 'Logs' and 'Errors and warnings', both with upward-pointing arrows.

This Azure Function (HyponatremiaDDx) subscribes to Azure Event Grid and is triggered whenever a new bloodSodium value is published (new event).

## Azure Function <https://azure.microsoft.com/en-us/services/functions/>

```
string decision = "unknown";  
if (bloodSodiumValue < 135) {decision = "true";}   
if (bloodSodiumValue >= 135) {decision = "false";}   
HttpClient client = new HttpClient();
```

```
string addVariableUri = "http://csmlab7.uconn.edu/addVariable";  
var firstRequest = new HttpRequestMessage(HttpMethod.Post, addVariableUri);  
dynamic firstRequestBody = new ExpandoObject();  
    firstRequestBody.actorId = actorId;  
    firstRequestBody.variable = "hyponatremia";
```

```
string setValueUri = "http://csmlab7.uconn.edu/setValue";  
var secondRequest = new HttpRequestMessage(HttpMethod.Post, setValueUri);  
    secondRequest.Headers.Add("Content-Type", "application/json");
```

The HyponatremiaDDx function 1) decides whether it is hyponatremia or not, 2) adds a new variable “hyponatremia” to the patient actor in case the variable is not yet added, and 3) sets the value to true (hyponatremia) or false (not hyponatremia).

**Step 8 [getValue]:** Get the value of the variable “hyponatremia” which is now “true” (because  $109 < 135$ )

Example Value | Model

```
{  
  "actorId": "patient032904475",  
  "variable": "hyponatremia"  
}
```

Response body

"true"

**Step 9 [setValue]:** Set the value of the variable “bloodSodium” to 235.

Example Value | Model

```
{  
  "actorId": "patient032904475",  
  "variable": "bloodSodium",  
  "value": 235  
}
```

**Step 10 [getValue]:** Get the value of the variable “hyponatremia” which is now “false” (because  $235 > 135$ )

Response body

"false"

## Step 11 [removeVariable]: Remove the variable “bloodSodium”.

POST

**/removeVariable** Removes a variable from an actor.

Sample request body (requestBody):

```
{
  "actorId": "patient032904475",
  "variable": "bloodSodium"
}
```

- "actionId": string
- "variable": string

To try this service:

1. Click [Try it out] button (white).
2. Type your request body into “Example Vaule | Model” textbox (white). A sample request body is shown above.
3. Click [Execute] button (blue).
4. Check “Response body” (ignore “Code” for now). If you see "Variable removed", your request is processed successfully. Otherwise, you will get an error message.

Response body

"Variable removed"

For more information about Nanoservice, please visit <https://csml.uconn.edu/>