What is the Azure Sphere Public API?

- 03/23/2020

Azure Sphere provides a public API through which you can request and receive data from devices in the field. This API can be useful for the following tasks:

- Provisioning
- Configuration
- Containerization
- Deployment
- Orchestration
- Automation
- Compliance

The Azure Sphere public API uses the REST (REpresentational State Transfer) HTTP protocol to request and receive data formatted in JSON (JavaScript Object Notation).

Components of a REST API request/response pair

A REST API request/response pair has five components:

1. The request URI, in the following form:

   ```
   VERB
   https://prod.core.sphere.azure.net/v{version}/{collection}/{id}…[{resourc
   eId} | {collection}]
   ```

   Parameters:

   - *collection*: One or more collections. Multiple nested collections are supported, so relative paths can include /collection/id/collection/id …

     Example: `/v2/tenants/{tenantId}/devices/{deviceId}/images`

   - *resourceId*: The ID of a specific resource, which enables access to specific resources within a collection.

     Example: `/v2/tenants/{tenantId}/devicegroups/{devicegroupid}`

    - *version*: The API version, which identifies the version of the API. Every API request should include an api-version to avoid having your app or service break as APIs evolve.

      Example: `/v2`

2. HTTP request message header fields:
    - A required HTTP method (also known as an operation or verb), which tells the service what type of operation you are requesting.
    - Additional header fields, as required by the specified URI and HTTP method. Specifically, an authorization header which provides a bearer token containing client authorization token for the request.
3. Optional HTTP request message body fields to support the URI and HTTP operation.
    - For HTTP POST or PUT operations, the body should be specified in the Content-Type request header as well as application/json.
4. HTTP response message header fields:
    - An HTTP status code. Successful calls return 2xx codes; 4xx and 5xx codes are error statuses. Alternatively, a service-defined status code may be returned, as indicated in the API documentation.
    - Optional additional header fields as required to support the response to the request, such as a Content-Type response header.
5. Optional HTTP response message body fields:
    - MIME-encoded response objects may be returned in the HTTP response body, such as a response from a GET method that is returning data. These objects are always returned in a structured JSON format, as indicated by the Content-Type response header.

Authenticate a request

Before you can make a valid request, your application or service must be authenticated with the Azure Sphere public API. The following table shows some of the ways you can authenticate.

Type of application	Description	Example	Authentication mechanism
Interactive client-side	GUI based client side application	Windows app enumerating devices	Microsoft Authentication Library (MSAL)
Interactive JavaScript	GUI based JavaScript application	AngularJS single page app displaying deployments for a device group.	MSAL
Interactive web	GUI based web application	Custom Web dashboard displaying build summaries	OAuth 2

Note

The **Azure Active Directory platform** is evolving into the **Microsoft Identity platform**.

Authentication library values

If you call the Active Directory Libraries to acquire a Bearer Token for use authentication, you must supply four values:

- Azure Sphere Client Application ID: "0B1C8F7E-28D2-4378-97E2-7D7D63F7C87F" This Client Application ID is required for successful authentication.
- The scope for the user: https://sphere.azure.net/api/user_impersonation
- Azure Sphere Tenant ID: 7d71c83c-ccdf-45b7-b3c9-9c41b94406d9
- Azure Sphere API Endpoint: https://prod.core.sphere.azure.net/

You can find more information on authentication by reviewing Azure Active Directory Authentication Libraries and Authentication flows.

Make a request

After you have authenticated with Azure Sphere, you can make requests and receive responses.

The following C# example uses the HttpClient class to make a request.

C#Copy

```
namespace SpherePublicAPISample
{
    using System;
    using System.Collections.Generic;
```

```
    using System.Net.Http;
    using System.Net.Http.Headers;
    using System.Threading;
    using System.Threading.Tasks;
    // You install the Microsoft.Identity.Client reference by using Nuget,
    // starting at https://www.nuget.org/packages/Microsoft.Identity.Client.
    // Follow the instructions to install using Package Manager.
    using Microsoft.Identity.Client;

    class Program
    {
        /// <summary>
        /// Azure Sphere Public API resource URI
        /// </summary>
        private readonly List<string> Scopes = new List<string>() {
"https://sphere.azure.net/api/user_impersonation" };
        /// <summary>
        /// Azure Sphere Public API client application ID.
        /// </summary>
        private const string ClientApplicationId = "0B1C8F7E-28D2-4378-97E2-
7D7D63F7C87F";
        /// <summary>
        /// Azure Sphere Tenant ID.
        /// </summary>
        public const string Tenant = "7d71c83c-ccdf-45b7-b3c9-9c41b94406d9";
        /// <summary>
        /// Azure Sphere Public API URI
        /// </summary>
        private static readonly Uri AzureSphereApiUri = new
Uri("https://prod.core.sphere.azure.net/");
        /// <summary>
        /// Program entry-point.
        /// </summary>
        /// <returns>Zero on success, otherwise non-zero.</returns>
        private static int Main()
        {
            try
            {
                CancellationTokenSource cancellationTokenSource = new
CancellationTokenSource();

                Program program = new Program();

                program.ExecuteAsync(cancellationTokenSource.Token)
                    .GetAwaiter()
```

```
                .GetResult();

            Console.ReadLine();
        }
        catch (Exception ex)
        {
            Console.Error.WriteLine(ex.ToString());
            return -1;
        }
        return 0;
    }

    private async Task ExecuteAsync(CancellationToken cancellationToken)
    {
        IPublicClientApplication publicClientApp = PublicClientApplicationBuilder
            .Create(ClientApplicationId)
            .WithAuthority(AzureCloudInstance.AzurePublic, Tenant)
            .WithRedirectUri("http://localhost")
            .Build();

        AuthenticationResult authResult = await
publicClientApp.AcquireTokenInteractive(Scopes)
            .ExecuteAsync();

        string accessToken = authResult.AccessToken;

        // Call the Azure Sphere API to get tenants available to the
authenticated user.
        string result = await GetAsync(accessToken, "v2/tenants",
cancellationToken);

        Console.WriteLine(result);
    }

    private async Task<string> GetAsync(string accessToken, string relativeUrl,
CancellationToken cancellationToken)
    {
        using (HttpClient client = new HttpClient())
        {
            client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", accessToken);

            Uri uri = new Uri(AzureSphereApiUri, relativeUrl);
```

```
                using (HttpResponseMessage response = await client.GetAsync(uri,
cancellationToken))
                {
                    response.EnsureSuccessStatusCode();
                    return await response.Content.ReadAsStringAsync();
                }
            }
        }
    }
}
```

Receive a response

Each call returns a JSON response in the following format:

JSONCopy

```
[{"Id":"{tenantId}","Name":"Contoso","Roles":null}]
```

Each call also returns a status code:

- 200-299 = success
- 400-499 = client error
- 500-599 = server error

RFC 7231 provides more information on specific error codes.

Azure Sphere public API error codes

The Azure Sphere public API returns the following error codes:

- 400 - Bad request
- 404 - Not Found
- 409 - Conflict
- 410 - Gone
- 500 - Internal Server Error

General guidance for 4xx errors

Check your request syntax and the data being passed. Malformed requests can lead to errors.

General guidance for 500 errors

Occasionally, transient server errors may occur. If you receive a server error, retry the request a few times to see if the error goes away.

CORS support

Cross-Region Origin Sharing (CORS) is not supported in this release.