#GlobalAzure

#CloudGenVerona

@cloudgen_verona

Thanks to all the sponsors

PREMIUM SPONSOR

4WARD

KIRATECH®
WE DEVOPS IT

BASIC SPONSOR

CODICEPLASTICO

dgroove
Tecnologia emozionale.

TOPIC

Real-time mobile apps with Xamarin and Azure SignalR Service

Who I am

Andrea Ceroni

@andrekiba

andrea.ceroni@gmail.com

https://github.com/andrekiba

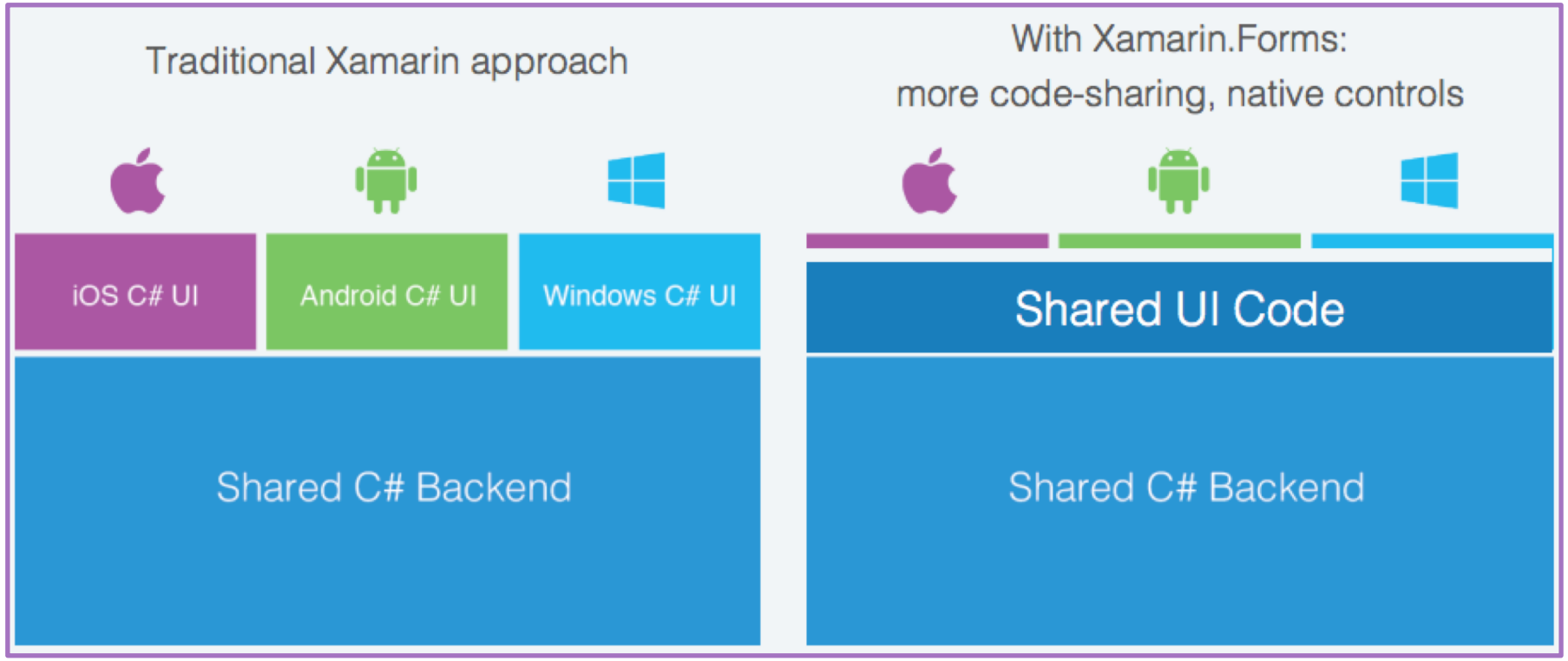
https://www.linkedin.com/in/andreaceroni/

elfo.net

klabcommunity.org

Real-time apps with Xamarin and Azure SignalR

Xamarin.Forms



Azure SignalR Service

What is Azure SignalR Service?

03/01/2019 • 2 minutes to read • Contributors

Azure SignalR Service simplifies the process of adding real-time web functionality to applications over HTTP. This real-time functionality allows the service to push content updates to connected clients, such as a single page web or mobile application. As a result, clients are updated without the need to poll the server, or submit new HTTP requests for updates.

This article provides an overview of Azure SignalR Service.

What is Azure SignalR Service used for?

There are many application types that require real-time content updates. The following examples are good candidates for using Azure SignalR Service:

- Apps that require high frequency updates from the server. Examples are gaming, voting, auction, maps, and GPS apps.
- Dashboards and monitoring apps. Examples include company dashboards and instant sales updates.
- Collaborative apps. Whiteboard apps and team meeting software are examples of collaborative apps.
- Apps that require notifications. Social networks, email, chat, games, travel alerts, and many other apps use notifications.

SignalR provides an abstraction over a number of techniques used for building real-time web applications. WebSockets is the optimal transport, but other techniques like Server-Sent Events (SSE) and Long Polling are used when other options aren't available. SignalR automatically detects and initializes the appropriate transport based on the features supported on the server and client.

In addition, SignalR provides a programming model for real-time applications that allows the server to send messages to all connections, or to a subset of connections that belong to a specific user or have been placed in an arbitrary group.

https://docs.microsoft.com/en-us/azure/azure-signalr/signalr-overview

Demo

Use SignalR Service

Scale an ASP.NET Core SignalR App

integrato con ASP.NET Core (basato su questo) permette di fare scale-out sulle connessioni

Build serveless real-time apps

utilizzando i bindings per Azure Functions (v2) è possibile "reagire" a tutta una serie di eventi (storage, queue, http trigger...)

Send messages from server to client with the REST API

integrazione con altre app che possono chiamare le API e "pushare" informazioni ai client

Concepts

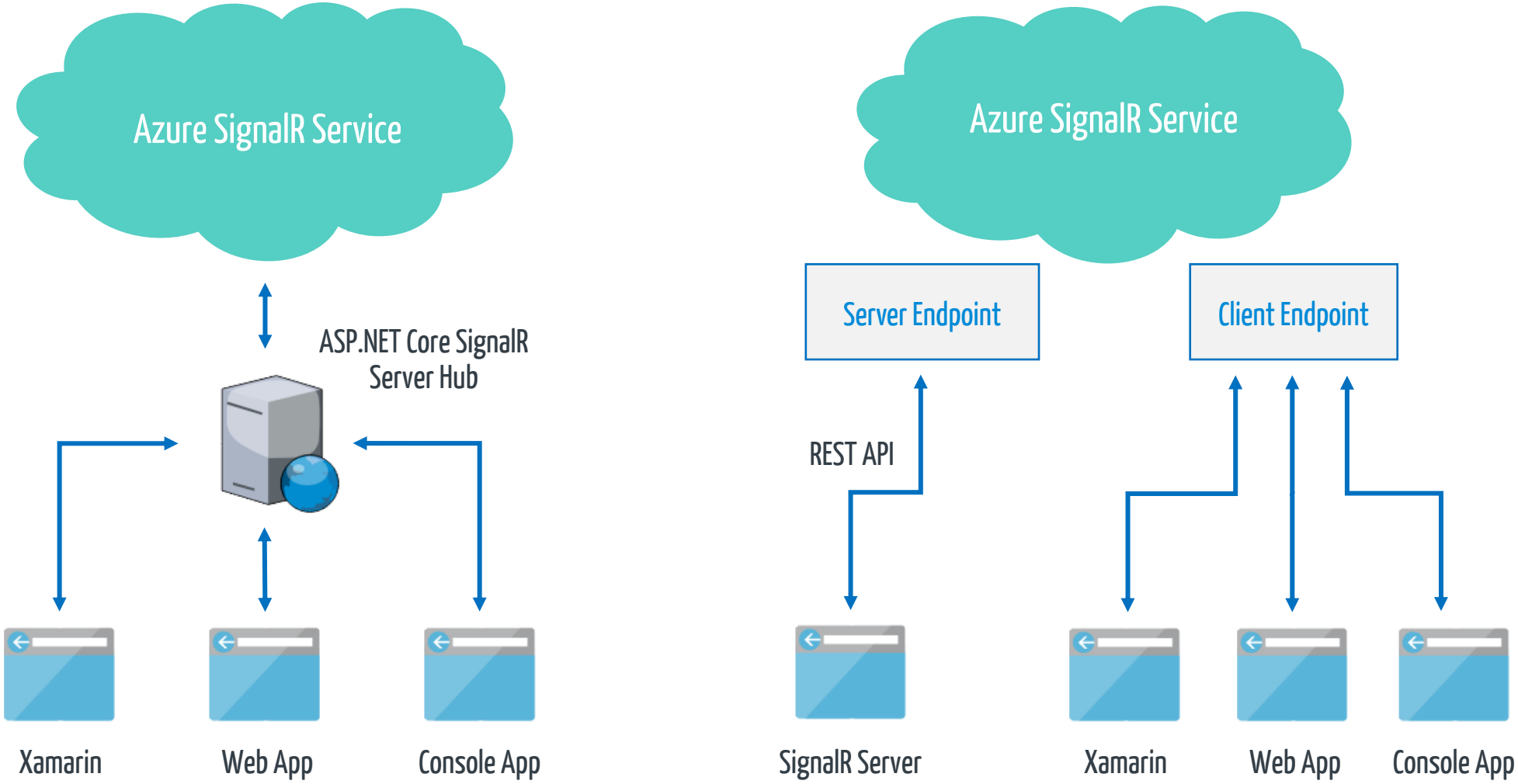
Transports

in base alle capacità di server e client SignalR sceglie il traporto migliore tra WebSocket, Server-Sent Events e Long Polling

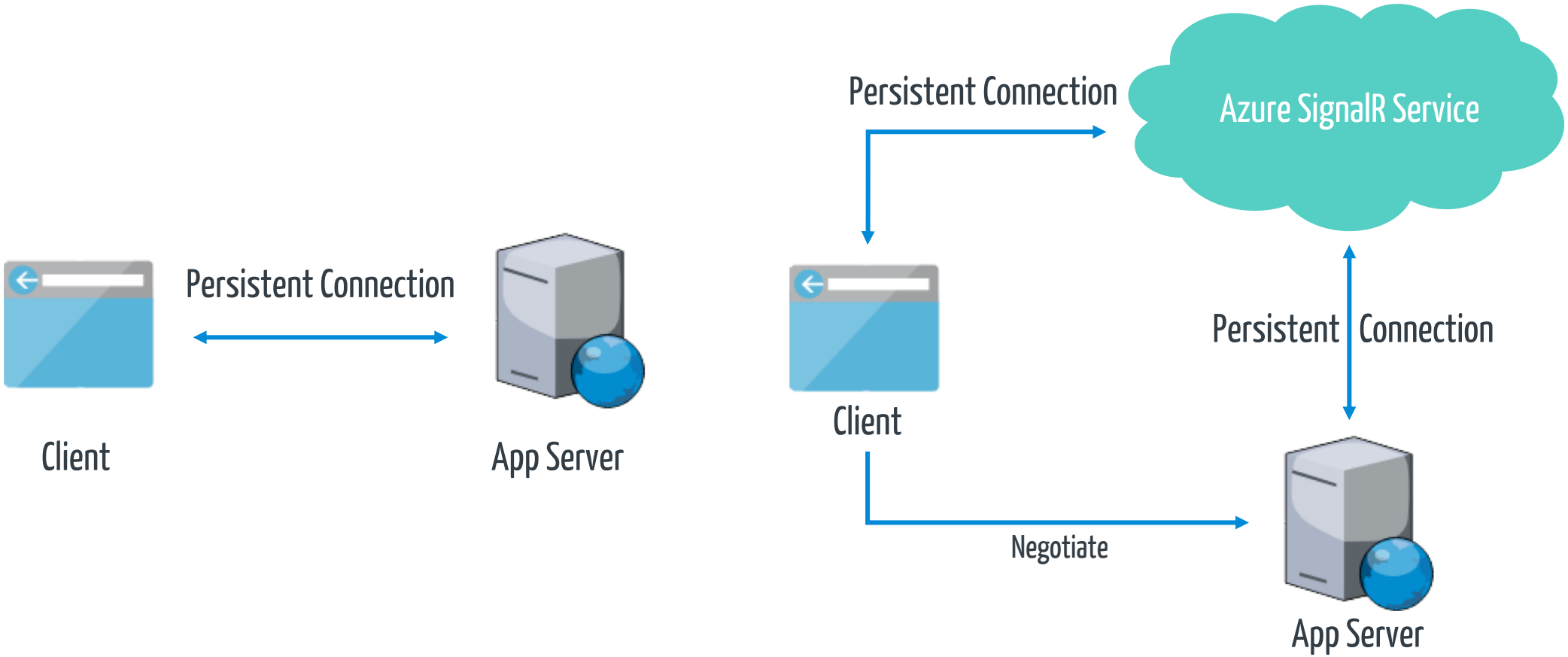
Hubs

un hub è una pipeline di alto livello che permette a client e server di chiamarsi a vicenda, passando parametri "strongly-typed" ai metodi.
Il messaggio contiene il nome del metodo da invocare e i relativi parametri
Esiste quindi una sorta di "contratto" vero e proprio tra client e server

With our without ASP.NET Core SignalR



ASP.NET Core SignalR vs SignalR Service



Different connections

Server connections

tra SignalR Service e SignalR Server vengono aperte connessioni websocket persistenti (5 di default), utilizzate per la comunicazione bidirezionale con i client

Client connections

i client vengono connessi direttamente a SignalR Service, url e token di accesso vengono risolti dall'endpoint "negotiate"

Data transmit

1 to 1 mapping tra connessione client e relativa connessione server che la prende in carico
Di fatto quindi SignalR Service fa da layer di trasporto tra server e client, gestendo completamente le connessioni (scalabilità)

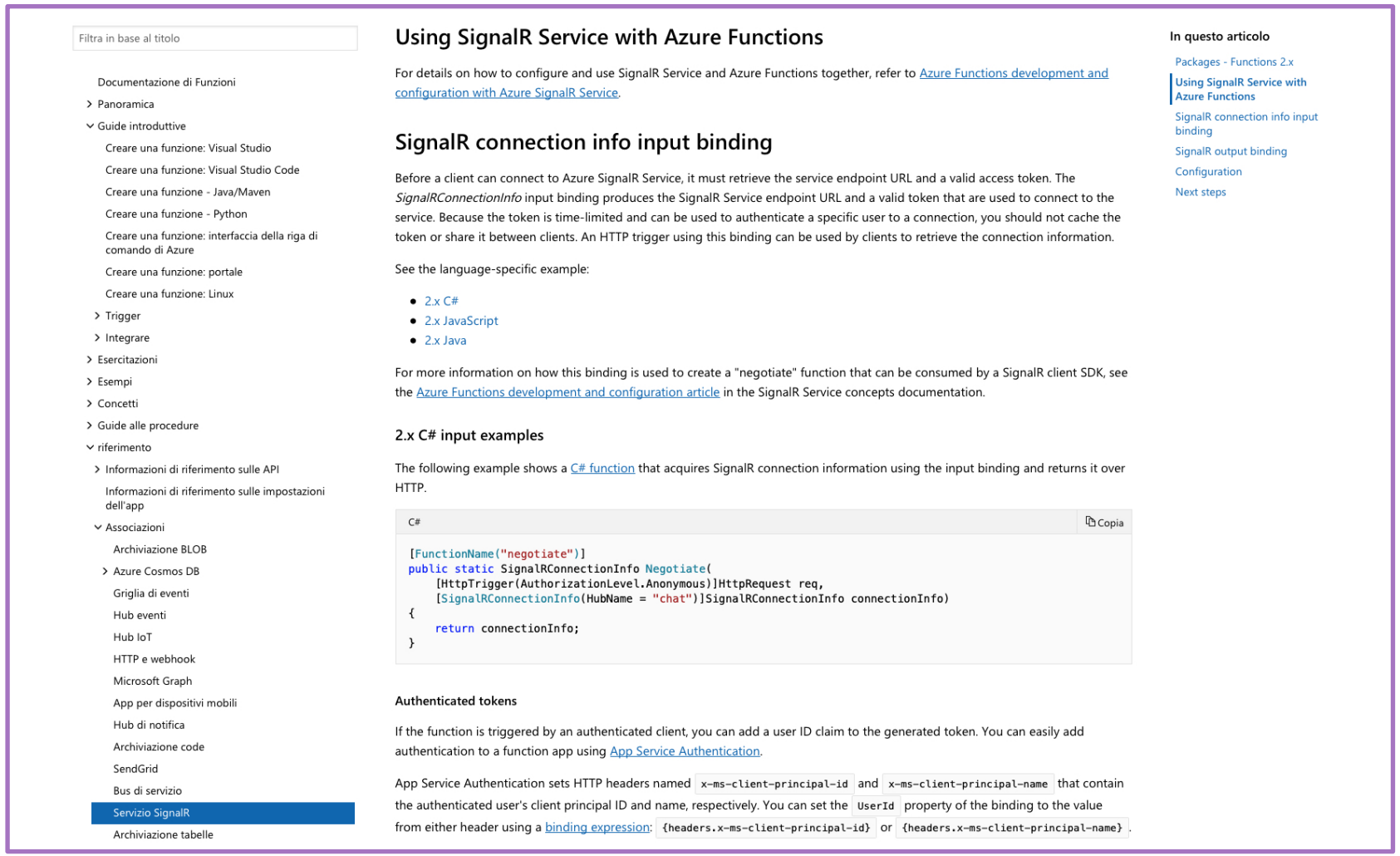
Meeting Rooms

Meeting
Room API

Azure SignalR

Microsoft
Graph

SignalR Service Bindings for Azure Functions



Filtra in base al titolo

- Documentazione di Funzioni
- Panoramica
- Guide introduttive
  - Creare una funzione: Visual Studio
  - Creare una funzione: Visual Studio Code
  - Creare una funzione - Java/Maven
  - Creare una funzione - Python
  - Creare una funzione: interfaccia della riga di comando di Azure
  - Creare una funzione: portale
  - Creare una funzione: Linux
  - Trigger
  - Integrare
- Esercitazioni
- Esempi
- Concetti
- Guide alle procedure
- riferimento
  - Informazioni di riferimento sulle API
  - Informazioni di riferimento sulle impostazioni dell'app
  - Associazioni
    - Archiviazione BLOB
    - Azure Cosmos DB
    - Griglia di eventi
    - Hub eventi
    - Hub IoT
    - HTTP e webhook
    - Microsoft Graph
    - App per dispositivi mobili
    - Hub di notifica
    - Archiviazione code
    - SendGrid
    - Bus di servizio
    - Servizio SignalR
    - Archiviazione tabelle

Using SignalR Service with Azure Functions

For details on how to configure and use SignalR Service and Azure Functions together, refer to Azure Functions development and configuration with Azure SignalR Service.

SignalR connection info input binding

Before a client can connect to Azure SignalR Service, it must retrieve the service endpoint URL and a valid access token. The *SignalRConnectionInfo* input binding produces the SignalR Service endpoint URL and a valid token that are used to connect to the service. Because the token is time-limited and can be used to authenticate a specific user to a connection, you should not cache the token or share it between clients. An HTTP trigger using this binding can be used by clients to retrieve the connection information.

See the language-specific example:

- 2.x C#
- 2.x JavaScript
- 2.x Java

For more information on how this binding is used to create a "negotiate" function that can be consumed by a SignalR client SDK, see the Azure Functions development and configuration article in the SignalR Service concepts documentation.

2.x C# input examples

The following example shows a C# function that acquires SignalR connection information using the input binding and returns it over HTTP.

C# Copia

```
[FunctionName("negotiate")]
public static SignalRConnectionInfo Negotiate(
    [HttpTrigger(AuthorizationLevel.Anonymous)]HttpRequest req,
    [SignalRConnectionInfo(HubName = "chat")]SignalRConnectionInfo connectionInfo)
{
    return connectionInfo;
}
```

Authenticated tokens

If the function is triggered by an authenticated client, you can add a user ID claim to the generated token. You can easily add authentication to a function app using App Service Authentication.

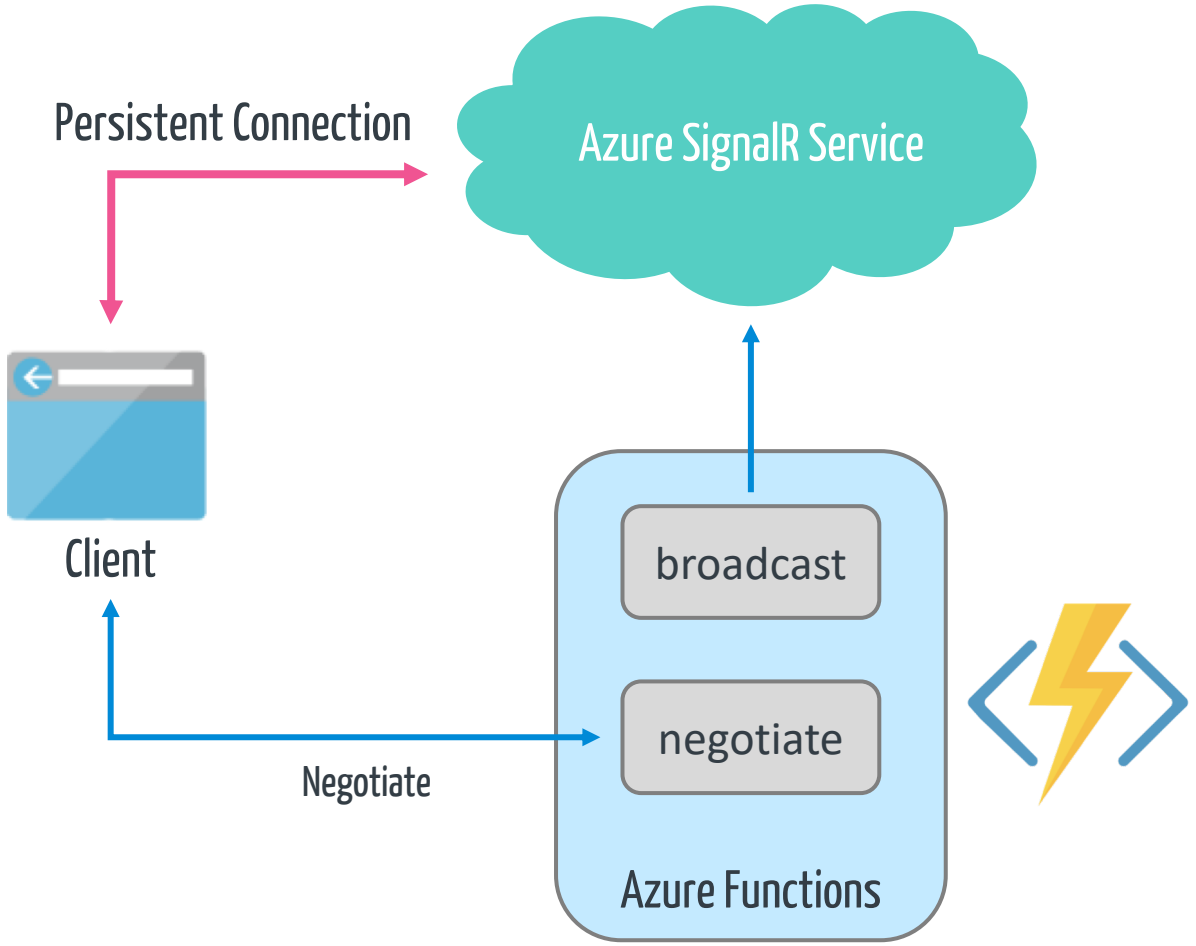
App Service Authentication sets HTTP headers named `x-ms-client-principal-id` and `x-ms-client-principal-name` that contain the authenticated user's client principal ID and name, respectively. You can set the `UserId` property of the binding to the value from either header using a binding expression: `{headers.x-ms-client-principal-id}` or `{headers.x-ms-client-principal-name}`.

In questo articolo

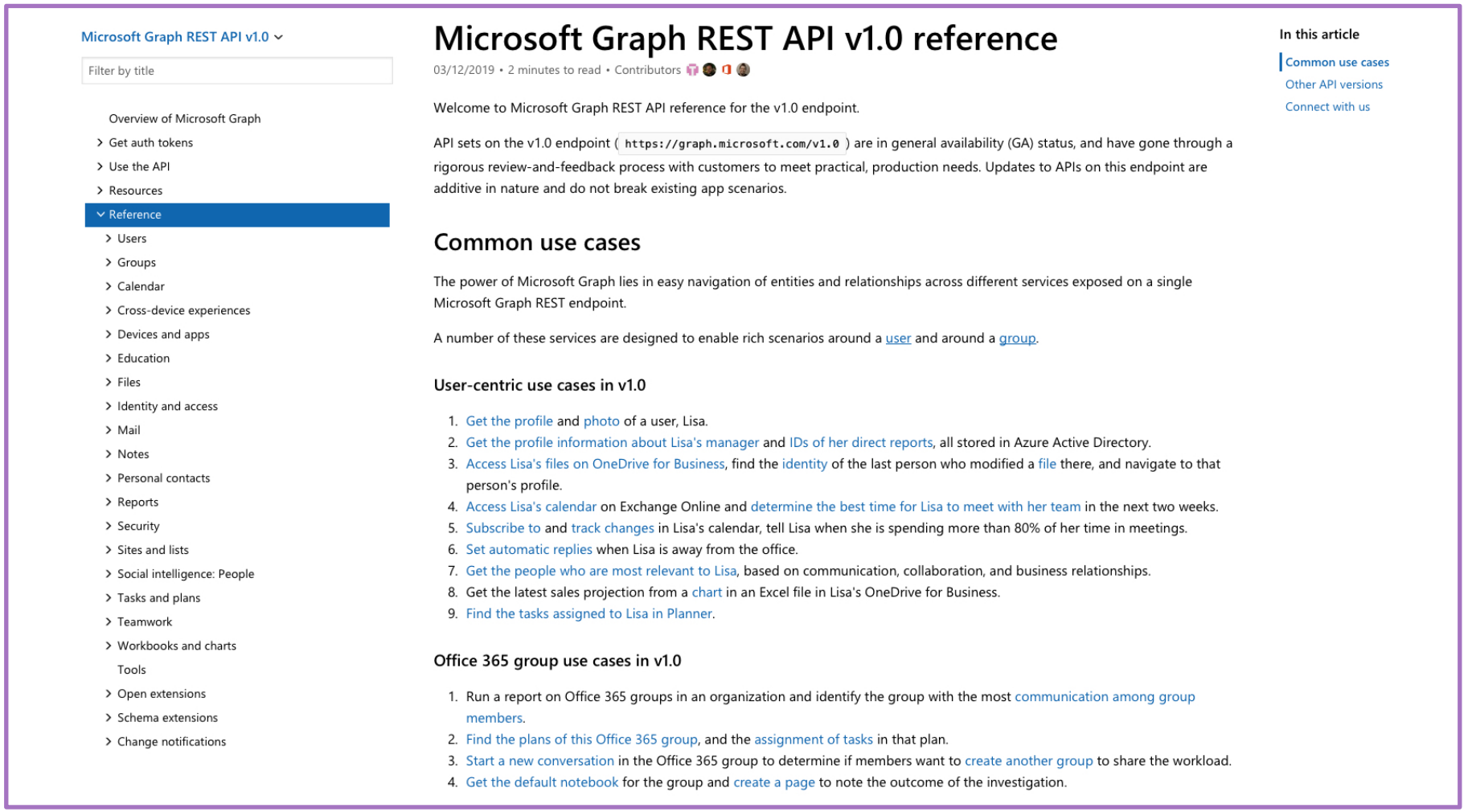
- Packages - Functions 2.x
- Using SignalR Service with Azure Functions
- SignalR connection info input binding
- SignalR output binding
- Configuration
- Next steps

https://docs.microsoft.com/it-it/azure/azure-functions/functions-bindings-signalr-service#2x-c-input-examples

Server-less Architecture with Azure Functions



Microsoft Graph API



Microsoft Graph REST API v1.0

Filter by title

- Overview of Microsoft Graph
- Get auth tokens
- Use the API
- Resources
- Reference
  - Users
  - Groups
  - Calendar
  - Cross-device experiences
  - Devices and apps
  - Education
  - Files
  - Identity and access
  - Mail
  - Notes
  - Personal contacts
  - Reports
  - Security
  - Sites and lists
  - Social intelligence: People
  - Tasks and plans
  - Teamwork
  - Workbooks and charts
  - Tools
  - Open extensions
  - Schema extensions
  - Change notifications

Microsoft Graph REST API v1.0 reference

03/12/2019 • 2 minutes to read • Contributors

Welcome to Microsoft Graph REST API reference for the v1.0 endpoint.

API sets on the v1.0 endpoint (`https://graph.microsoft.com/v1.0`) are in general availability (GA) status, and have gone through a rigorous review-and-feedback process with customers to meet practical, production needs. Updates to APIs on this endpoint are additive in nature and do not break existing app scenarios.

Common use cases

The power of Microsoft Graph lies in easy navigation of entities and relationships across different services exposed on a single Microsoft Graph REST endpoint.

A number of these services are designed to enable rich scenarios around a user and around a group.

User-centric use cases in v1.0

1. Get the profile and photo of a user, Lisa.
2. Get the profile information about Lisa's manager and IDs of her direct reports, all stored in Azure Active Directory.
3. Access Lisa's files on OneDrive for Business, find the identity of the last person who modified a file there, and navigate to that person's profile.
4. Access Lisa's calendar on Exchange Online and determine the best time for Lisa to meet with her team in the next two weeks.
5. Subscribe to and track changes in Lisa's calendar, tell Lisa when she is spending more than 80% of her time in meetings.
6. Set automatic replies when Lisa is away from the office.
7. Get the people who are most relevant to Lisa, based on communication, collaboration, and business relationships.
8. Get the latest sales projection from a chart in an Excel file in Lisa's OneDrive for Business.
9. Find the tasks assigned to Lisa in Planner.

Office 365 group use cases in v1.0

1. Run a report on Office 365 groups in an organization and identify the group with the most communication among group members.
2. Find the plans of this Office 365 group, and the assignment of tasks in that plan.
3. Start a new conversation in the Office 365 group to determine if members want to create another group to share the workload.
4. Get the default notebook for the group and create a page to note the outcome of the investigation.

In this article

- Common use cases
- Other API versions
- Connect with us

https://docs.microsoft.com/en-us/graph/api/overview?view=graph-rest-1.0

Demo

Q & A


```
[FunctionName("Question")]
public async Task<IActionResult> Question(
        [HttpTrigger(AuthorizationLevel.Anonymous, "post")] Question question,
        [SignalR(HubName = "Q&A")] IAsyncCollector<SignalRMessage> answers)
{
        var answer = iKnowTheAnswer ☺ ? question.GetAnswer() : ... ☹ ...
        return await answers.AddAsync(new SignalRMessage {
                Target = "Answer",
                Arguments = new [ ] { answer }
        });
}
```

Links

https://github.com/andrekiba/Gab19

https://docs.microsoft.com/en-us/azure/azure-signalr/signalr-overview

https://github.com/Azure/azure-signalr/blob/dev/docs/rest-api.md

https://docs.microsoft.com/it-it/azure/azure-functions/functions-bindings-signalr-service#2x-c-input-examples

https://github.com/Azure/azure-signalr/blob/dev/docs/faq.md#service-mode

https://docs.microsoft.com/en-us/graph/api/overview?view=graph-rest-1.0

Thanks!

andrea.ceroni@elfo.net

andrea.ceroni@gmail.com

@andrekiba

http://github.com/andrekiba

http://www.linkedin.com/in/andreaceroni