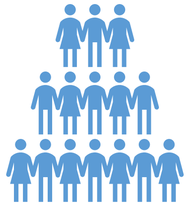
Introduction

Freelance Technical Consultant & Trainer

Microsoft MVP Business Applications

Experience with Dynamics NAV / Business Central since 2002



Co-founder Dutch Dynamics Community

http://kauffmann.nl

Email: aj@kauffmann.nl
Twitter: @ajkauffmann

Creating custom APIs with AZ AL Dev Tools

- New AL File wizard to add a new Page
  - Set page type to API
  - API properties automatically filled in (setting)
  - Add multiple fields
  - No ApplicationArea
  - No Captions (setting)
  - Automatically convert names

- Tip:
  - Think about the order of fields
  - ODataKeyFields must be added afterwards

```
"alOutline.defaultApiPublisher": "cronus",
"alOutline.defaultApiGroup": "data",
"alOutline.defaultApiVersion": "v1.0",
"alOutline.createApiFieldsCaptions": false,
"alOutline.apiFieldNamesConversion": [
    {
        "searchRegExp": "^lineNo$",
        "newValue": "sequence"
    },
    {
        "searchRegExp": "^no$",
        "newValue": "number"
    },
    {
        "searchRegExp": "No$",
        "newValue": "Number"
    },
    {
        "searchRegExp": "^systemId$",
        "newValue": "id"
    },
    {
        "searchRegExp": "^systemModifiedAt$",
        "newValue": "lastModifiedDateTime"
    }
],
```

API page triggers

Different flow per data operation

API page triggers – GET operation

API page triggers – POST operation

API page triggers – PATCH operation

API page triggers – DELETE operation

Turn POST into PATCH

- When the record is read from the OnValidate trigger:
  - OnNewRecord still executes
  - OnInsertRecord is skipped
  - OnModifyRecord is executed instead

```
field(id; Rec.SystemId)
{
    trigger OnValidate()
    begin
        Rec.GetBySystemId(Rec.SystemId);
    end;
}
0 references
field(number; Rec."No.")
{
    trigger OnValidate()
    begin
        Rec.Get(Rec."No.");
    end;
}
```

Using temporary tables

- Potential performance hit
  - Large tables
  - Paging
  - No caching

- Tips:
  - Fill data with a query
  - Reduce data by filtering
  - Force filterin in code

> Don't use temp tables as a source if you have many records. Temp tables that are based on APIs are a performance hit. The server has to fetch and insert every record, and there's no caching on data in temp tables. Paging becomes difficult to do in a performant manner. A rule of thumb is if you have more than 100 records, don't use temp tables.

Using temporary tables

- Potential performance hit
  - Large tables
  - Paging
  - No caching

- Tips:
  - Fill data with a query
  - Reduce data by filtering
  - Force filtering in code

```
trigger OnOpenPage()
var
    GLEntriesAPIMgt: Codeunit GLEntriesAPIMgt;
begin
    GLEntriesAPIMgt.GetIncomeTransactions(Rec);
end;
```

```
procedure GetIncomeTransactions(var GLEntryBuffer: Record GLEntryBuffer)
var
    Transactions: Query GLEntries;
begin
    Transactions.SetRange(GlobalDimension1Code, 'QR00000', 'QR99999');
    Transactions.SetFilter(ShortcutDimension4Code, '%1|%2', 'XPRA01', '#');
    Transactions.SetRange(BalanceAccountType, "Gen. Journal Account Type"::Customer);
    Transactions.SetRange(IncomeBalance, Transactions.IncomeBalance::"Balance Sheet");
    Transactions.SetRange(AccountCategory, "G/L Account Category"::Assets);
    Transactions.SetRange(AccountSubcategory, 'Bank and Cash');
    Transactions.SetRange(AccountType, "G/L Account Type"::Posting);
    Transactions.SetRange(PostingDate, GLEntryBuffer.GetMinDateFilter(), GLEntryBuffer.GetMaxDateFilter());

    Transactions.Open();

    while Transactions.Read() do
        GLEntryBuffer.CopyFromTransaction(Transactions);

    Transactions.Close();
end;
```

Bound Action return value

Option 1: Location Header

Option 1 – Copy all APIs

Relationships between APIs

- Two types:
  - 1:1
  - 1:n
- Defined in page part
- Automatic relationship is created under certain conditions:
  - TableRelation specified
  - API exists for both tables

```
part(itemVariants; "APIV2 - Item Variants")
{
    Caption = 'Variants';
    EntityName = 'itemVariant';
    EntitySetName = 'itemVariants';
    SubPageLink = "Item Id" = field(SystemId);
}
```

```
part(baseUnitOfMeasure; "APIV2 - Units of Measure")
{
    Caption = 'Unit Of Measure';
    Multiplicity = ZeroOrOne;
    EntityName = 'unitOfMeasure';
    EntitySetName = 'unitsOfMeasure';
    SubPageLink = SystemId = Field("Unit of Measure Id");
}
```

Partial records

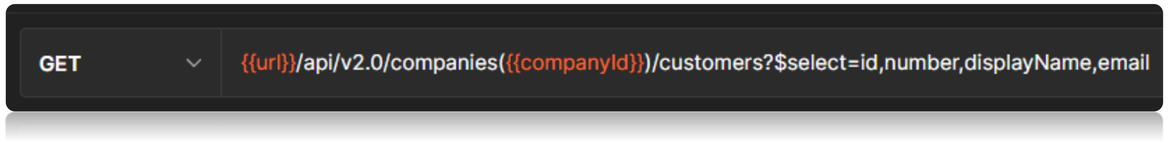
- Partial records is automatically enabled for API pages
- Fields defined in the repeater are automatically loaded
- Any other field that is accessed from code causes a JIT (Just In Time) load
- Avoid JIT loads by using Rec.AddLoadFields

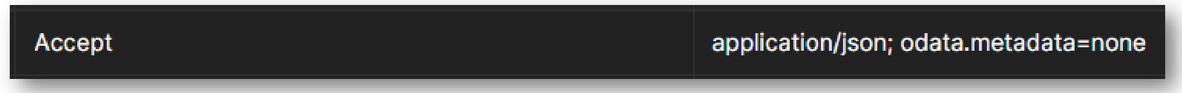
```
trigger OnOpenPage()
begin
    Rec.AddLoadFields("Value Id", "Dimension Value Code");
end;
```

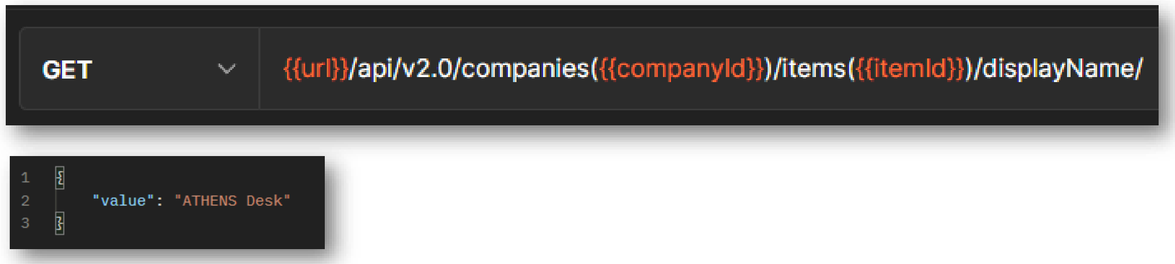
```
local procedure SetCalculatedFields()
begin
    GlobalDimensionValueId := "Value Id";
    GlobalDimensionValueCode := "Dimension Value Code";
end;
```

Reducing JSON payload

- $select= to reduce the JSON payload
- Do not use $top and $skip. Instead use server-drive paging with header odata.maxpagesiz
- Accept header to manage OData tags
- Navigate to a single property







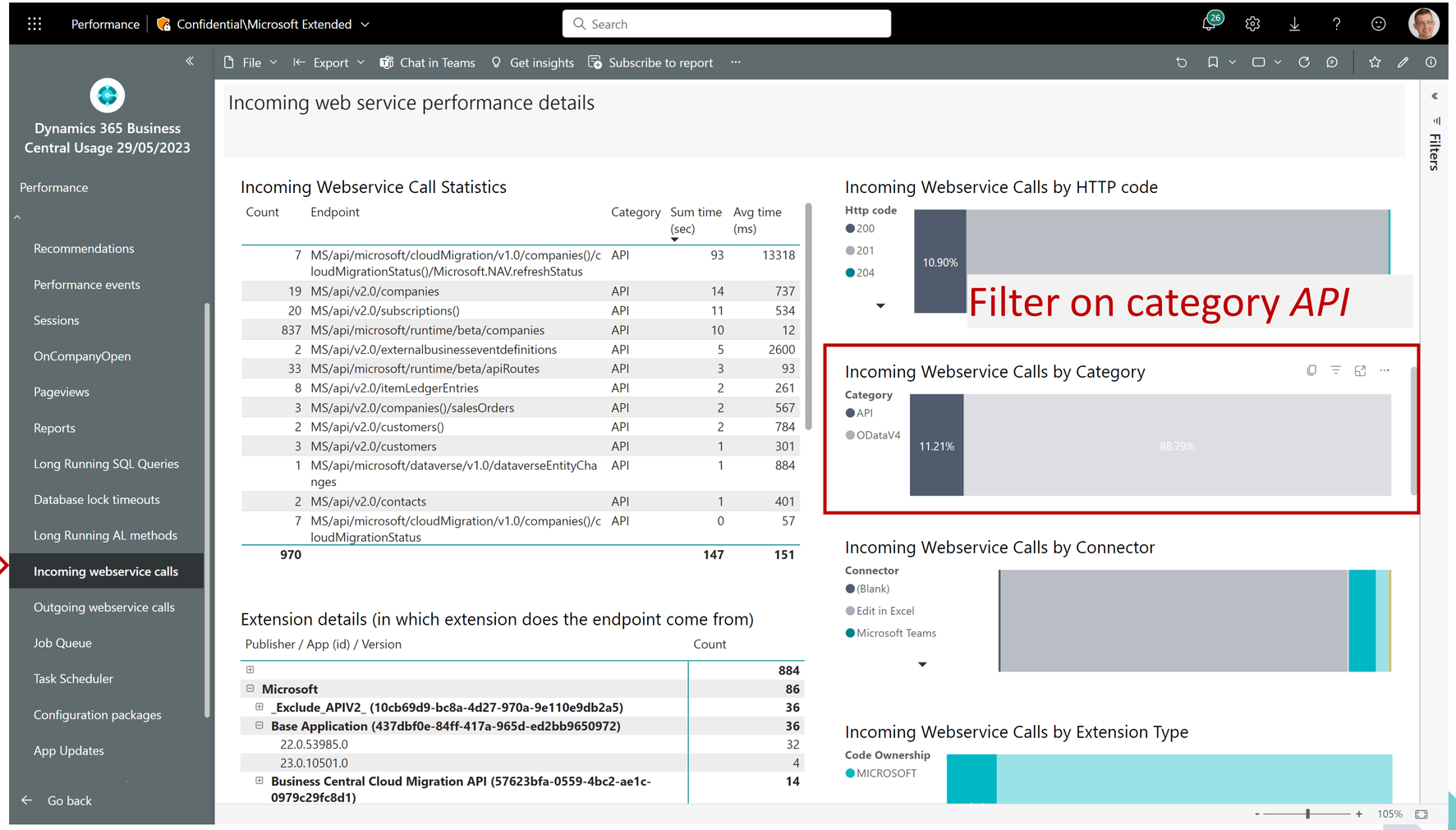
Monitoring incoming web service calls

Four things to consider:

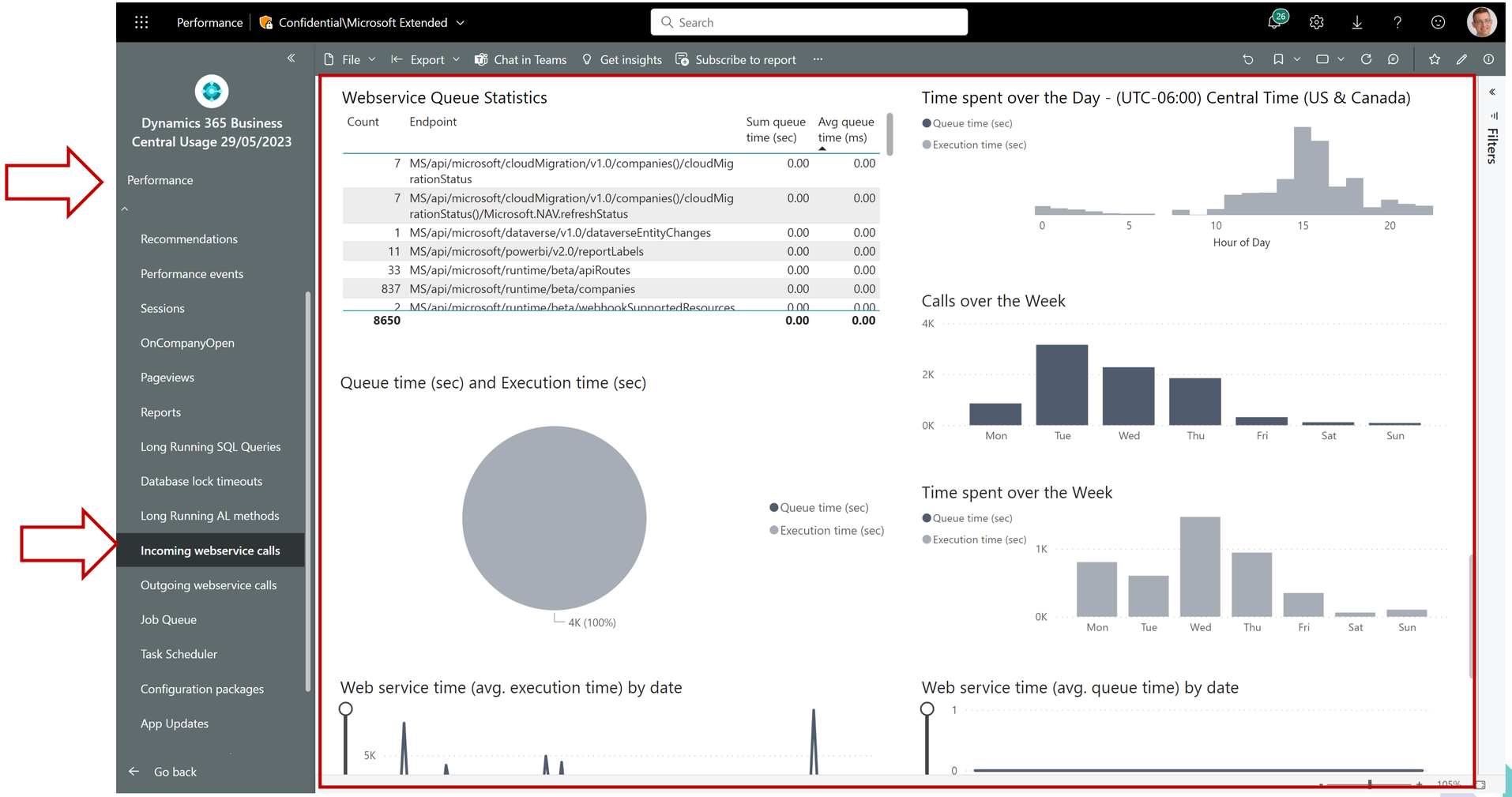
1. Use APIs over OData/SOAP on UI pages
2. Reduce aggressive calls
3. Check time spend in queues
4. Fix non-200 HTTP errors (these just waste resources)

5. For more tips check:
   http://aka.ms/bcperformance
   https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/api-reference/v2.0/dynamics-error-codes

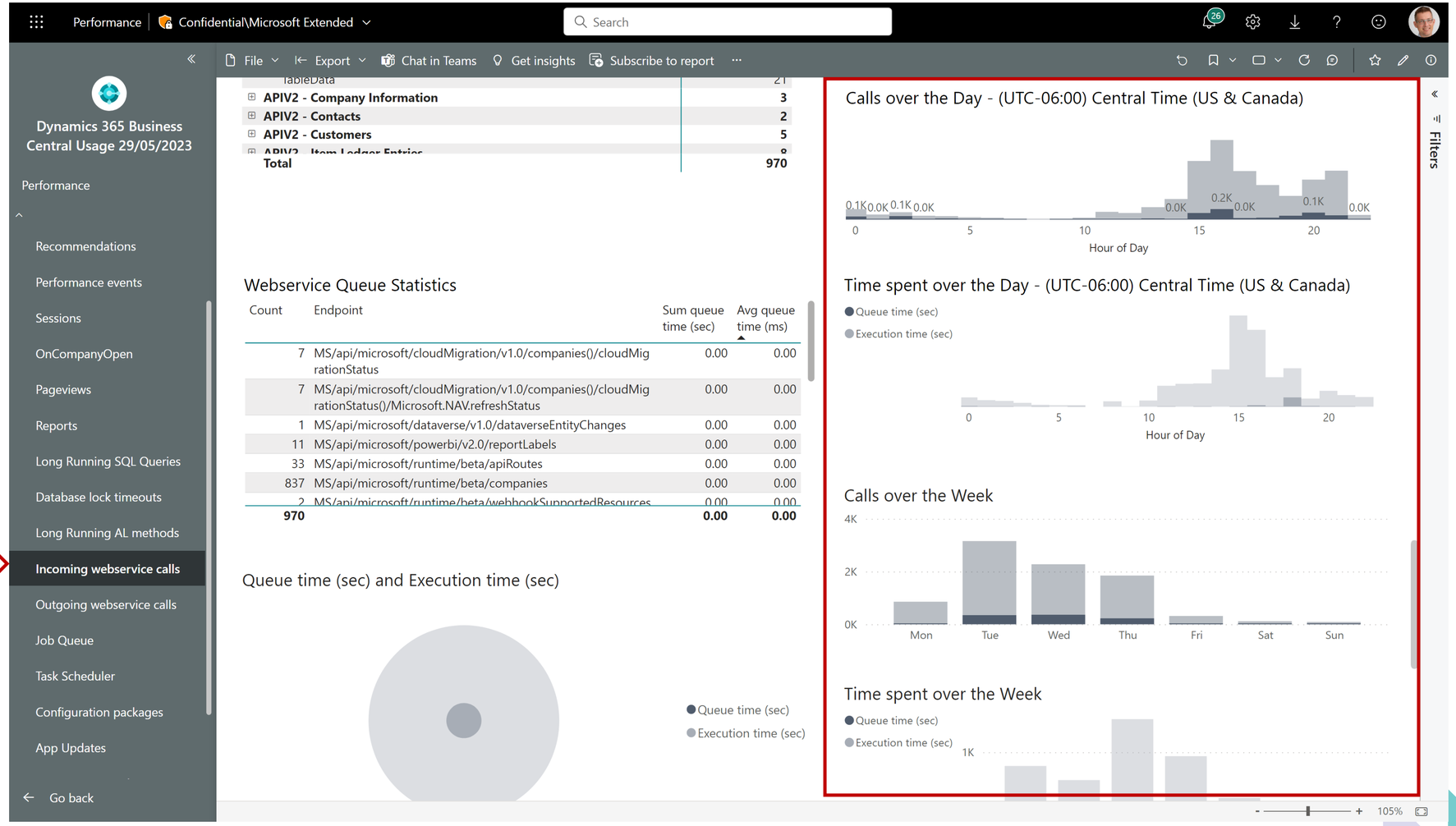
Is your API code fast?



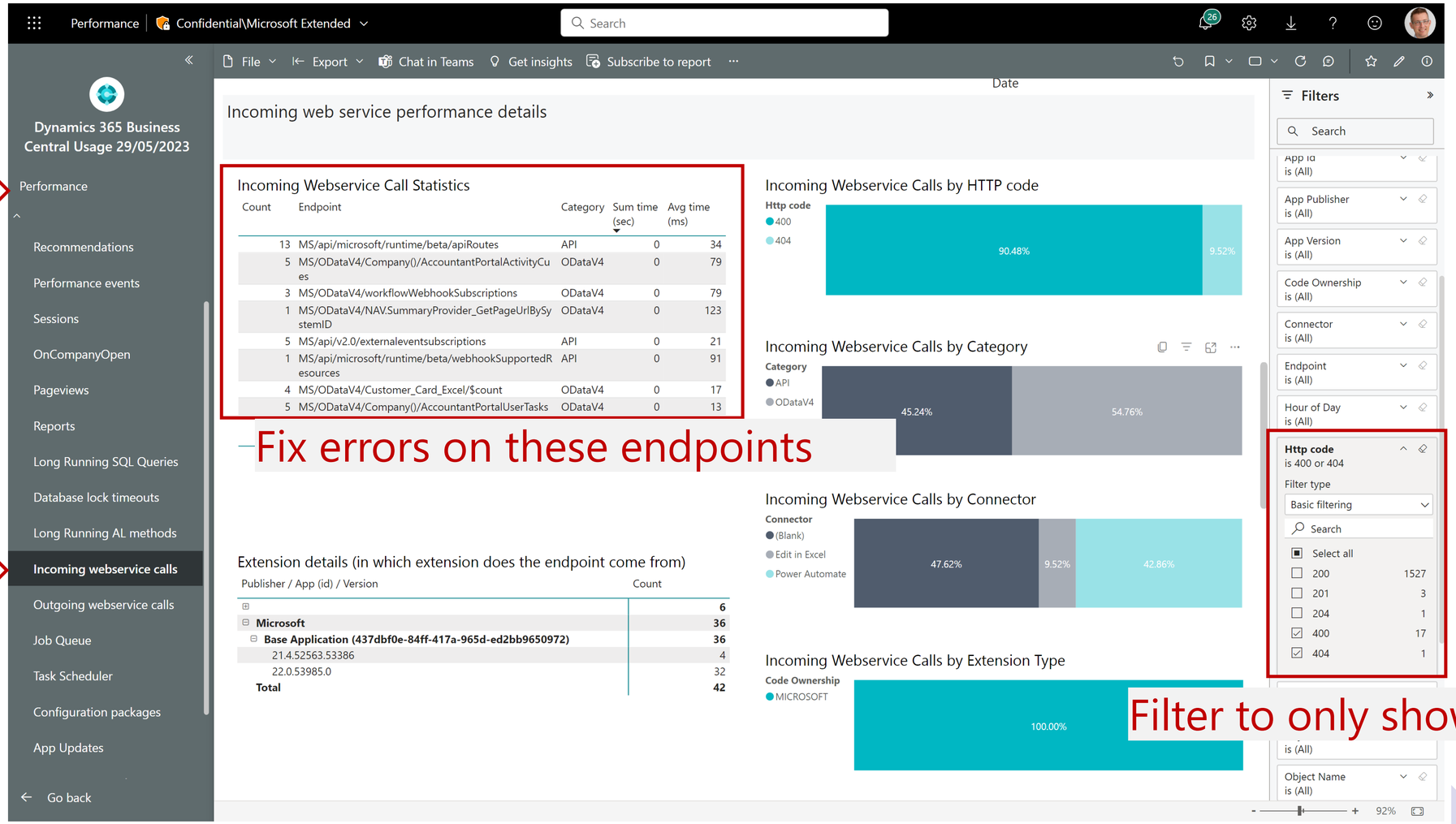
Do your API calls get queued up?



Are your APIs called aggressively?

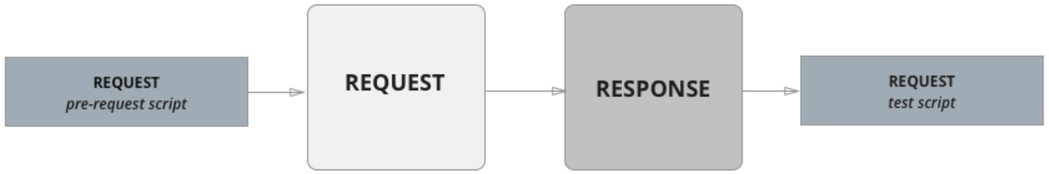


Incoming web service performance – non-200 calls



Postman test script

- Postman runtime is based on Node.js
- Add dynamic behavior to requests
  - Pre-request scripts
  - Test scripts
  - Dynamics parameters
  - Pass data between requests
- Javascript code



POST {{url}}/api/v2.0/companies({{companyId}})/customers

Params Authorization Headers (11) Body Pre-request Script Tests Settings

```
pm.test("Status code is 201", function () {
  pm.response.to.have.status(201);
});

const jsonData = pm.response.json();

pm.test("Response is a json object", () => {
    pm.expect(jsonData).to.be.an("object");
});

const schema = {…
}

pm.test("Response schema is valid", () => {
    pm.expect(tv4.validate(jsonData, schema)).to.be.true;
});

pm.test("The response has all properties", () => {
    pm.expect(jsonData.displayName).to.eql('BC Techdays 2023');
    pm.expect(jsonData.city).to.eql('Antwerp');
    pm.expect(jsonData.country).to.eql('BE');
    pm.expect(jsonData.type).to.eql('Company');
    pm.expect(jsonData.currencyCode).to.eql('EUR');
});
```

Body Cookies Headers (11) Test Results (4/4)

All Passed Skipped Failed

PASS Status code is 201

PASS Response is a json object

PASS Response schema is valid

PASS The response has all properties

Consuming APIs with C# - Authentication

- Add NuGet package Microsoft.Identity.Client
- Use the package for retrieving OAuth access tokens

```
static IPublicClientApplication publicApp =
    PublicClientApplicationBuilder
        .Create(clientId)
        .WithTenantId(azureTenant)
        .WithDefaultRedirectUri()
        .Build();
```

```
private static AuthenticationResult ExecuteAuthorizationCodeGrantFlow()
{
    AuthenticationResult result;

    var accounts = publicApp.GetAccountsAsync().Result;

    if (accounts.Any())
    {
        try
        {
            result = publicApp.AcquireTokenSilent(scopesDynamic, accounts.FirstOrDefault())
                              .ExecuteAsync()
                              .Result;
        }
        catch (MsalUiRequiredException)
        {
            result = publicApp.AcquireTokenInteractive(scopesDynamic)
                              .ExecuteAsync()
                              .Result;
        }
    }
    else
    {
        result = publicApp.AcquireTokenInteractive(scopesDynamic)
                          .ExecuteAsync()
                          .Result;
    }

    return result;
}
```

Consuming APIs with C# - Authentication

- Add NuGet package Microsoft.Identity.Client
- Use the package for retrieving OAuth access tokens

```
static IConfidentialClientApplication confidentialApp =
    ConfidentialClientApplicationBuilder
        .Create(clientId)
        .WithClientSecret(clientSecret)
        .WithAuthority(AzureCloudInstance.AzurePublic, AadAuthorityAudience.AzureAdMultipleOrgs)
        .Build();
```

```
private static AuthenticationResult ExecuteClientCredentialsFlow()
{
    AuthenticationResult result;

    result = confidentialApp.AcquireTokenForClient(scopesStatic)
                            .WithAuthority(AzureCloudInstance.AzurePublic, azureTenant)
                            .ExecuteAsync()
                            .Result;

    return result;
}
```

Consuming APIs with C# - Data

- Create a class with properties for each field in the Json payload
- Map each property to the corresponding Json key by using JsonPropertyName
- Do the same for the value property in the response

```
using System.Text.Json.Serialization;

namespace DemoOAuth.BusinessCentral
{
    1 reference
    public class Customers
    {
        [JsonPropertyName("value")]
        1 reference
        public Customer[] Value { get; set; }
    }
}
```

```
using System.Text.Json.Serialization;

namespace DemoOAuth.BusinessCentral
{

    5 references
    public class Customer
    {

        [JsonPropertyName("id")]
        0 references
        public string Id { get; set; }

        [JsonPropertyName("number")]
        2 references
        public string Number { get; set; }

        [JsonPropertyName("displayName")]
        3 references
        public string Name { get; set; }

        [JsonPropertyName("addressLine1")]
        0 references
        public string Address { get; set; }

        [JsonPropertyName("addressLine2")]
        0 references
        public string Address2 { get; set; }

        [JsonPropertyName("city")]
        3 references
        public string City { get; set; }

        [JsonPropertyName("postalCode")]
        0 references
        public string PostalCode { get; set; }

        [JsonPropertyName("country")]
        2 references
        public string Country { get; set; }
    }
}
```

Consuming APIs with C# - Data

- Use the HttpClient extension method **GetFromJsonAsync** to automatically convert the Json response into an object

```
var customers = client.GetFromJsonAsync<Customers>($@"/v2.0/{environmentName}/api/v2.0/companies({companyId})/customers").Result;

Console.Clear();
foreach (var customer in customers.Value)
{
    Console.WriteLine($"{customer.Number} {customer.Name} {customer.City}");
}
```

Consuming APIs with C# - Data

- Use the HttpClient extension method **PostAsJsonAsync** to automatically convert an object into a Json request

- Use the HttpContent extension method **ReadFromJsonAsync** to automatically convert the Json response into an object

```
Customer customer = new()
{
    Name = "BC Techdays 2023",
    City = "Antwerp",
    Country = "BE"
};
JsonSerializerOptions options = new()
{
    DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull
};

string url = $@"/v2.0/{environmentName}/api/v2.0/companies({companyId})/customers";

var result = client.PostAsJsonAsync<Customer>(url, customer, options).Result;
customer = result.Content.ReadFromJsonAsync<Customer>().Result;

Console.Clear();
Console.WriteLine($"Customer created");
Console.WriteLine($"No.     : {customer.Number}");
Console.WriteLine($"Name    : {customer.Name}");
Console.WriteLine($"City    : {customer.City}");
Console.WriteLine($"Country: {customer.Country}");
Console.ReadKey();
```

BC/NAV TECH DAYS 2023
mibuso.com

Q&A

Any Questions?

BC/NAV TECH DAYS 2023
mibuso.com

Thank You!