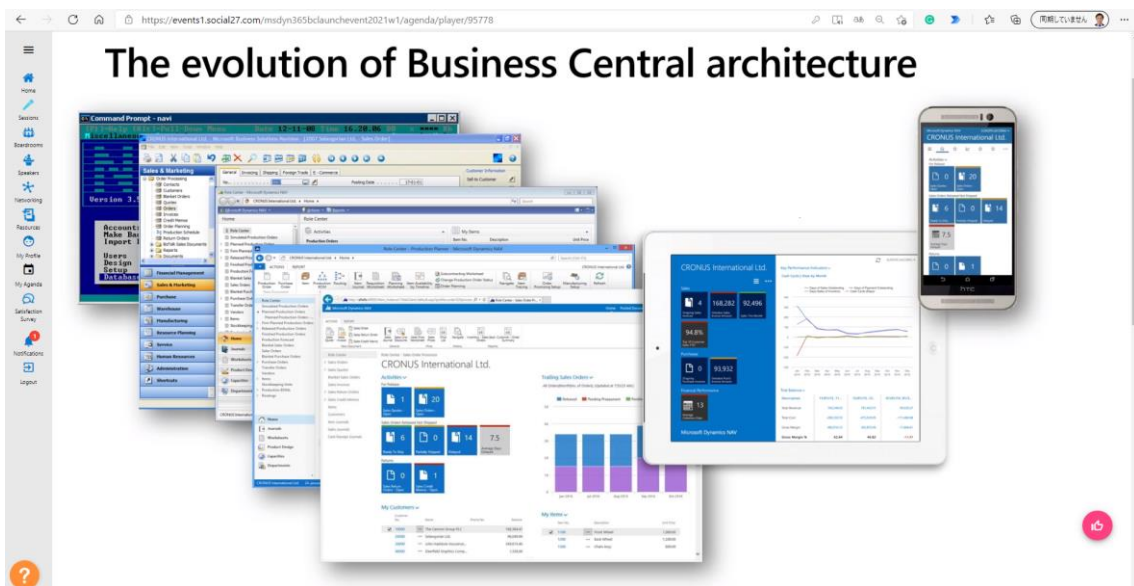
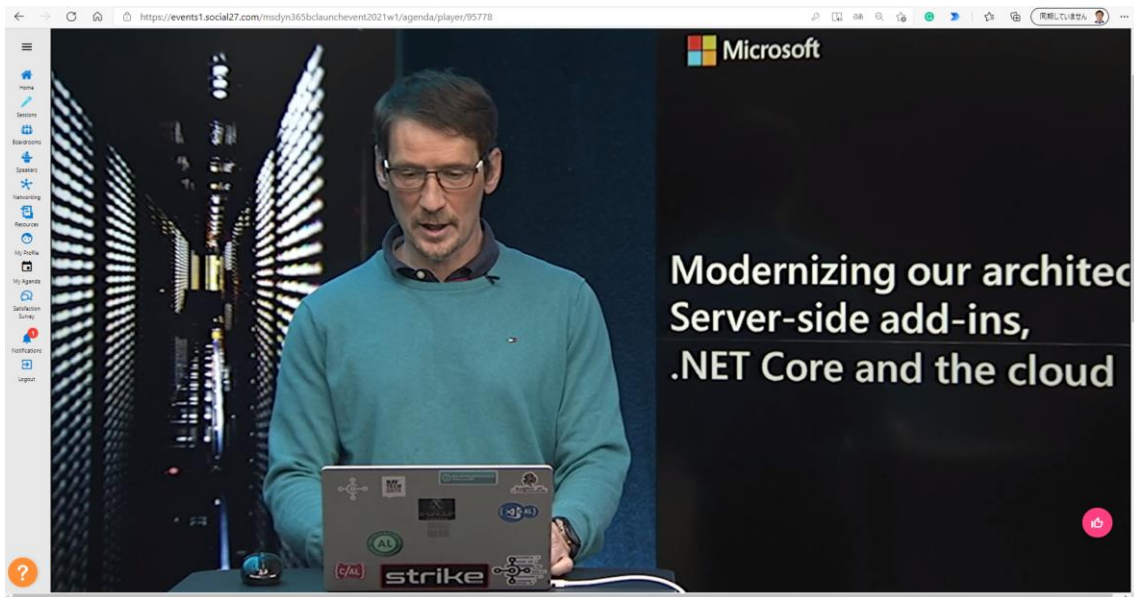
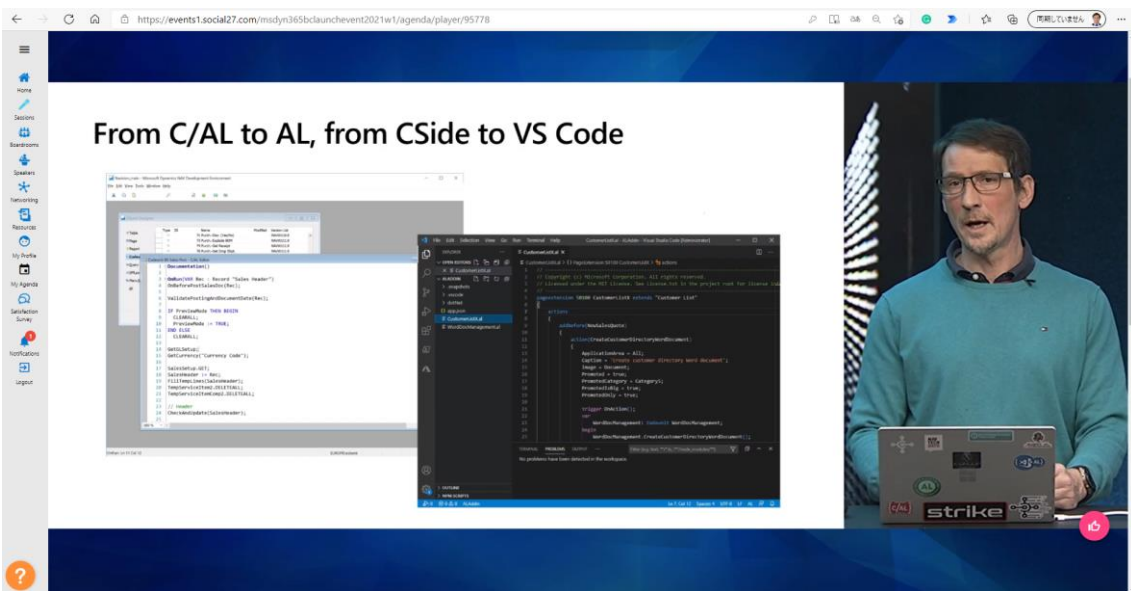
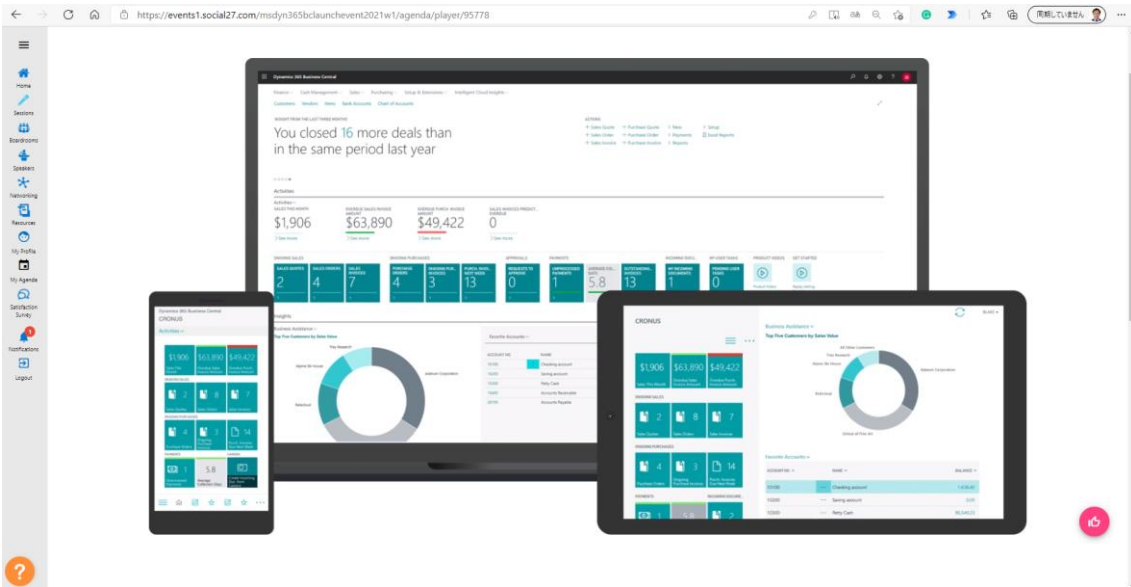


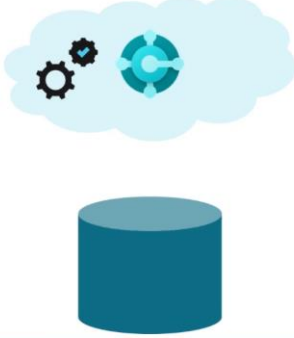
# Modernizing our architecture






https://events1.social27.com/msdyn365bclaunchevent2021w1/agenda/player/95778

## From on-premises to the cloud




A diagram illustrating the transition from on-premises to the cloud. It features a light blue cloud containing two black gears and a teal circular arrow icon. Below the cloud is a teal server cylinder.



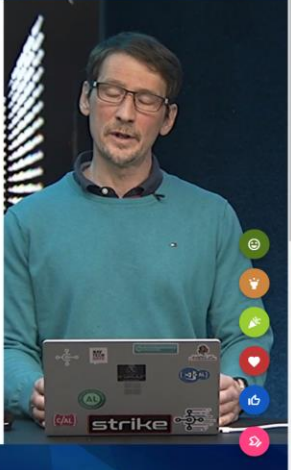
A speaker in a teal sweater is shown in a video feed, holding a laptop with 'strike' branding. The video feed includes standard social media interaction icons like like, share, and comment.

https://events1.social27.com/msdyn365bclaunchevent2021w1/agenda/player/95778

## Migrating to .NET Core



The .NET Core logo, consisting of a purple circle with the text '.NET Core' in white.




A speaker in a teal sweater is shown in a video feed, holding a laptop with 'strike' branding. The video feed includes standard social media interaction icons like like, share, and comment.

https://events1.social27.com/msdyn365bclaunchevent2021w1/agenda/player/95778

## Why moving to .NET Core

- .NET 4.8 is in sustained engineering
- No more improvements in .NET 4.8
- All new libraries are now built upon .NET Core
  - => potentially impossible to adopt
- .NET core
  - is more lightweight (= > smaller container images)
  - is more modular (only the needed modules are loaded)
  - has a more performant runtime
  - can run on Nano Server
  - can run in Azure Functions
  - can run in Service Fabric mesh (lightweight Service Fabric)




A speaker in a teal sweater is shown in a video feed, holding a laptop with 'strike' branding. The video feed includes standard social media interaction icons like like, share, and comment.

https://events1.social27.com/msdyn365bclaunchevent2021w1/agenda/player/95778

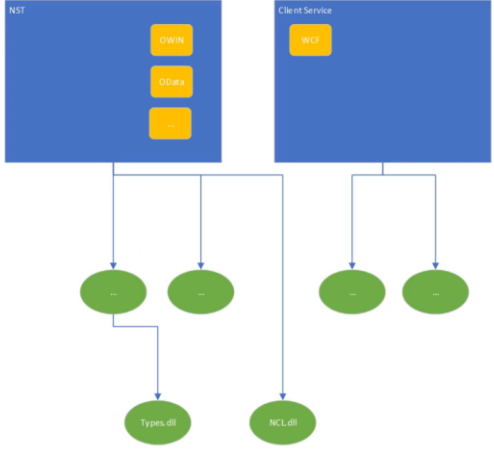
## Performance improvements

- Various products have evidence of 10% performance improvements when moving from 2.1 to 3.1
- Future performance improvements are to be expected



https://events1.social27.com/msdyn365bclaunchevent2021w1/agenda/player/95778

## Bottom-up & Top-down approach

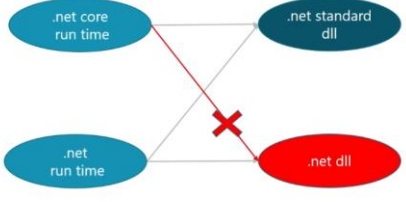


```

graph TD
    subgraph NSI
        CRM1[CRM]
        CRM2[CRM]
        CRM3[CRM]
    end
    subgraph Client_Service [Client Service]
        WEB[WEB]
    end
    NSI --- C1(( ))
    NSI --- C2(( ))
    NSI --- C3(( ))
    Client_Service --- C4(( ))
    Client_Service --- C5(( ))
    C1 --- Types[Types.dll]
    C2 --- Types
    C3 --- Types
    C4 --- NCL[NCL.dll]
    C5 --- NCL
  
```


https://events1.social27.com/msdyn365bclaunchevent2021w1/agenda/player/95778

## .NET, .NET Core and .NET Standard



```

graph TD
    A([.net core run time]) --- B([.net standard dll])
    A --- C([.net run time])
    B --- C
    C --- D([.net dll])
    B -.- X([X]) --- D
  
```



https://events1.social27.com/msdyn365blaunchevent2021w1/agenda/player/95778

## Server-side add-in architecture

AL extension

Base app

System app

NST

.NET Interop

.NET server-side add-in

3<sup>rd</sup> party library

Process boundary

Running within the .NET 4.8 runtime

https://events1.social27.com/msdyn365blaunchevent2021w1/agenda/player/95778

## Server-side add-in architecture

AL extension

Base app

System app

NST

.NET Interop

~~.NET server-side add-in~~

~~3<sup>rd</sup> party library~~

Process boundary

Running within the .NET Core runtime

https://events1.social27.com/msdyn365blaunchevent2021w1/agenda/player/95778

## Server-side add-in architecture

AL extension

Base app

System app

NST

.NET Interop

~~.NET server-side add-in~~

~~3<sup>rd</sup> party library~~

Process boundary

Running within the .NET Core runtime

**Solution #1: Migrate to .NET Core**

... but what about the third-party libraries  
...and what about the cloud!

**Solution #2: Move to a cloud-ready architecture**

https://events1.social27.com/msdyn365bclaunchevent2021w1/agenda/player/95778

## .NET Core and cloud-ready architecture

AL extension

Base app

System app

NST

Process boundary  
Running within the .NET Core runtime

HTTP Request

NET server-side add-in code

3<sup>rd</sup> party library

Function App in Azure Cloud

https://events1.social27.com/msdyn365bclaunchevent2021w1/agenda/player/95778

## An example: The Word Document Manager

https://events1.social27.com/msdyn365bclaunchevent2021w1/agenda/player/95778

CRONUS International Ltd. | Sales | Purchasing | Inventory | Posted Documents | Setup & Extensions

Customers: All | Search | + New | Delete | Process | Report | New Document | Customer | ...

Reminder: your work date is 1/26/2023 Use today | Chart | Create customer directory Word document

No. ↑	Name	Responsibility Center	Color	Owner
01121212	Spotsmeyer's Furnishings			
01445544	Progressive Home Furnishings			
01454545	New Concepts Furniture			
01905893	Candoxy Canada Inc.			
01905899	Elkhorn Airport		YELLOW	Mr. Rya
01905902	London Candoxy Storage Cam...		YELLOW	Mr. Johi
10000	The Cannon Group PLC	BIRMINGHAM	BLUE	Mr. Anc
20000	Selangorian Ltd.			Mr. Mar
20309920	Metatorad Malaysia Sdn Bhd		YELLOW	Mrs. Azi
20312912	Highlights Electronics Sdn Bhd		GREEN	Mr. Mar
20339921	Trax Tonic Sdn Bhd		YELLOW	Mrs. Ru
21233572	Somadis		YELLOW	M. Syed
21245278	Maronegoce		BLUE	Mme. F

Details | Attachments (0)

### Sell-to Customer Sales History

0	0	0
Ongoing Sales Quotes	Ongoing Sales Blanket Orders	Ongoing Sales Orders
0	0	0
Ongoing Sales Invoices	Ongoing Sales Return Orders	Ongoing Sales Credit Memos
0	0	0
Posted Sales Shipments	Posted Sales Invoices	Posted Sales Return Receipts
0		

https://events1.social27.com/msdyn365bclaunchevent2021w1/agenda/player/95778

Go to Agenda | My Agenda

Session: Modernizing our architecture

Dynamics 365 Business Central

CRONUS International Ltd. | Sales | Purchasing | Inventory | Posted

Customers: All | Search | + New | Delete | Process | Report | New D

Reminder: your work date is 1/26/2023 Use today | Change to... | Turn off reminder

No.	Name	Responsibility Center	Location Code	Phone No.	Contact
01121212	Spotsmeyer's Furnishings		YELLOW		Mr. Mike
01445544	Progressive Home Furnishings		YELLOW		Mr. Scott
01454545	New Concepts Furniture				
01905893	Candoxy Canada Inc.				
01905899	Elkhorn Airport				
01905902	London Candoxy Storage Campus				
10000	The Cannon Group PLC				
20000	Selangorian Ltd.				Mr. Mark
20309920	Metatorad Malaysia Sdn Bhd		YELLOW		Mrs. Azlee
20312912	Highlights Electronics Sdn Bhd		GREEN		Mr. Mark

Customer list created

OK

Downloads

- CustomerList.docx
- CustomerList.docx

See more

Sell-to Customer Sales History

	Ongoing Sales	Ongoing Sales Blanket Orders	Ongoing Sales Orders
	0	0	0
	0	0	0
	0	0	0
	0	0	0

https://events1.social27.com/msdyn365bclaunchevent2021w1/agenda/player/95778

Go to Agenda | My Agenda

Session: Modernizing our architecture

AutoSave | CustomerList - Compatibility Mode - Saved to this PC

File Home Insert Draw Design Layout References Mailings Review View Help

Calibri (Body) 11 A A As

1 Normal 1 No Spac... Heading 1 Heading 2

Find Replace Select Dictate Voice Sensitivity Editor Reuse Files

CustomerList

New Concepts Furniture  
705 West Peachtree Street  
US-GA 31772  
Atlanta

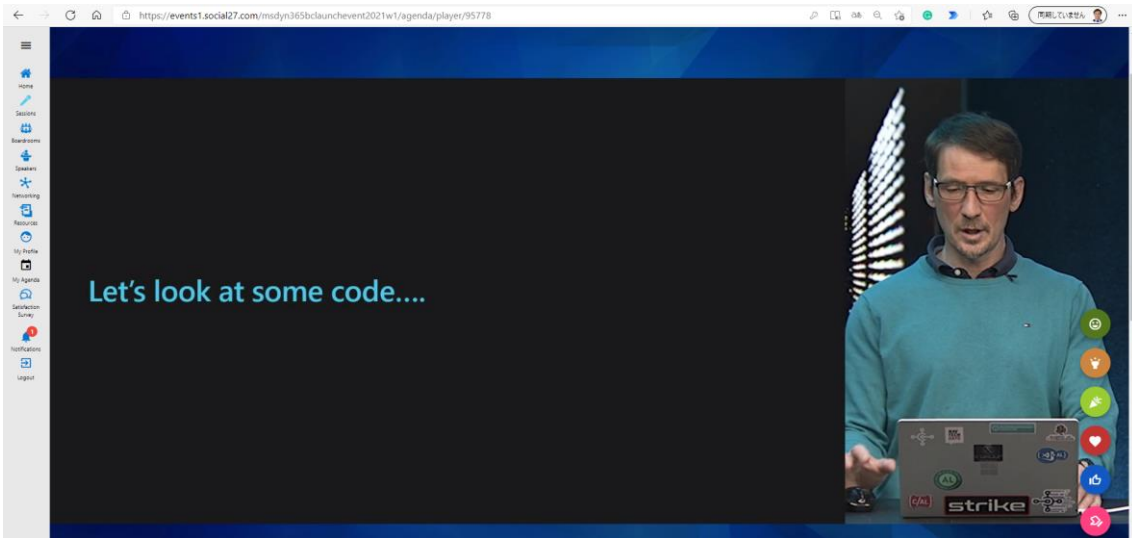
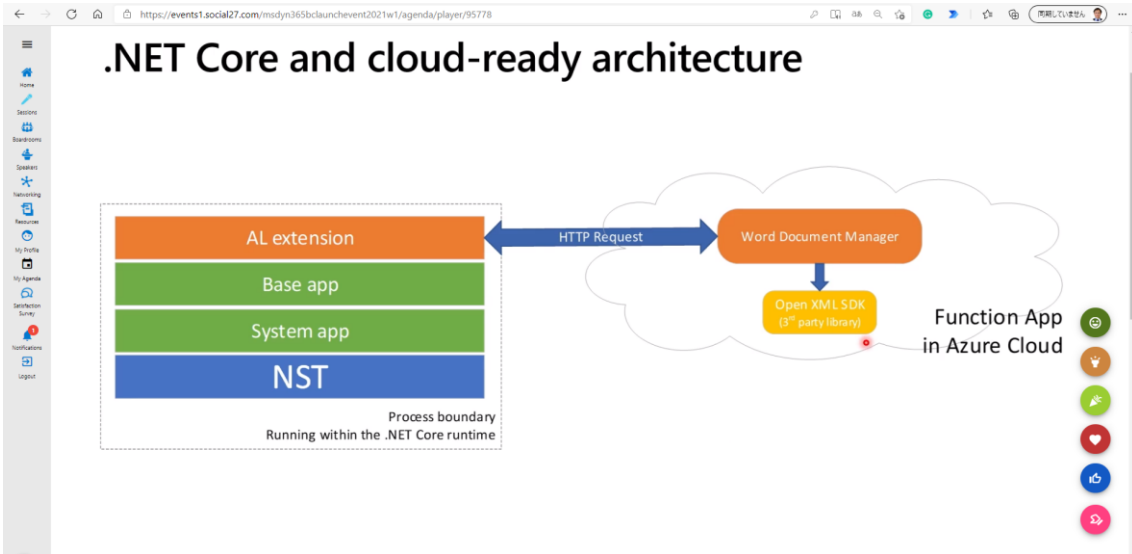
Candoxy Canada Inc.  
18 Cumberland Street  
CA-ON P7B 5E2  
Thunder Bay

Elkhorn Airport  
105 Buffalo Dr.  
CA-MB ROM 0N0  
Elkhorn

London Candoxy Storage Campus  
120 Wellington Rd.  
CA-ON N6B 1V7

CustomerList: 3,145 characters (an approximate value)

Vincent Nicolas



```

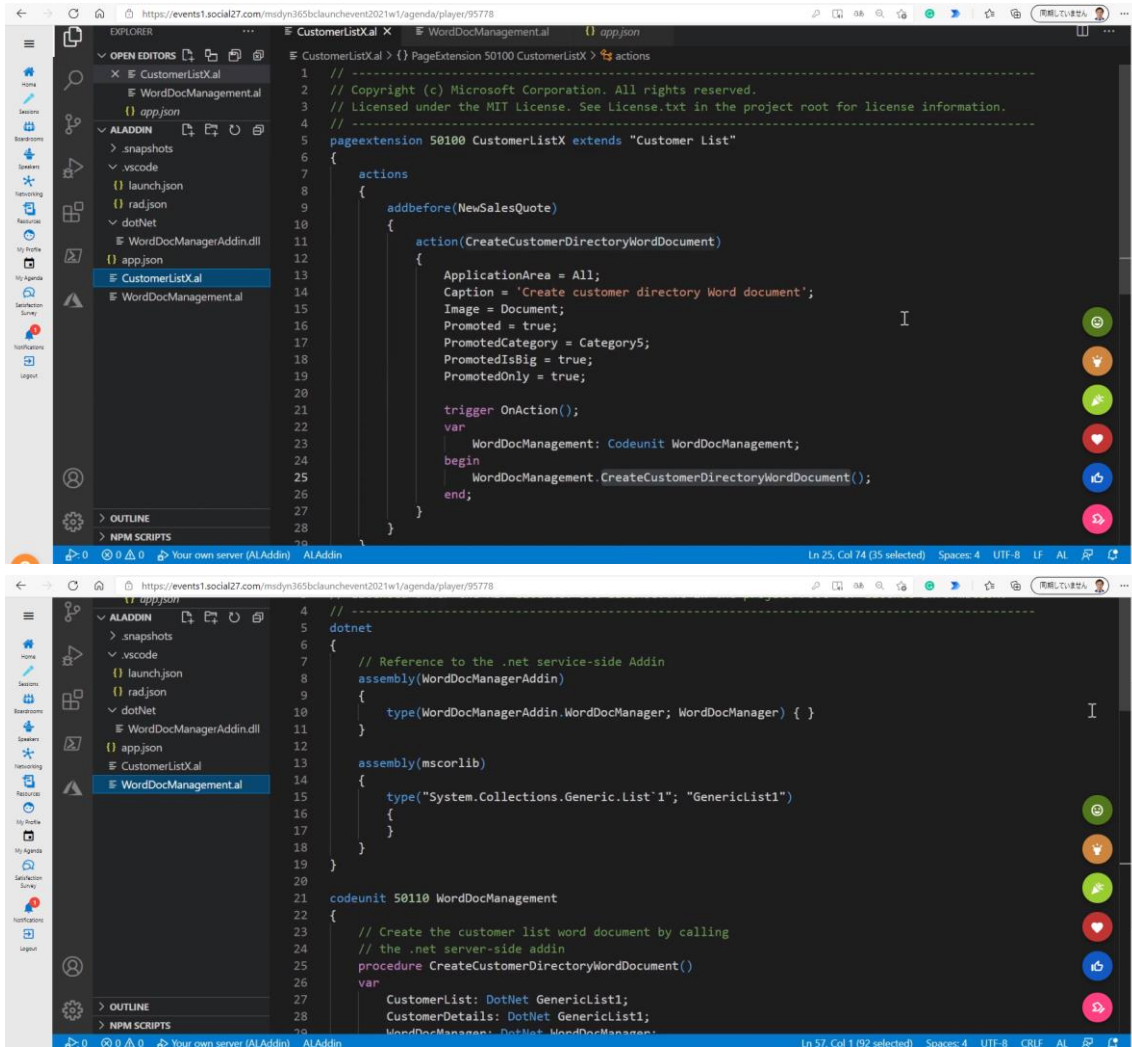
1 using DocumentFormat.OpenXml;
2 using DocumentFormat.OpenXml.Packaging;
3 using DocumentFormat.OpenXml.Wordprocessing;
4 using System;
5 using System.Collections.Generic;
6 using System.IO;
7 using System.Linq;
8
9 namespace WordDocManagerAddin
10 {
11     public class WordDocManager
12     {
13         public Stream CreateCustomerDirectoryDocument(List<Object> customers)
14         {
15             MemoryStream result = new MemoryStream();
16
17             using (WordprocessingDocument wordDocument = WordprocessingDocument.Create(result, WordprocessingDocumentType.Document))
18             {
19                 MainDocumentPart mainPart = wordDocument.AddMainDocumentPart();
20
21                 mainPart.Document = new Document();
22                 Body body = mainPart.Document.AppendChild(new Body());
23
24                 foreach (var customer in customers.Cast<IObject>())
25                 {
26                     Paragraph para = body.AppendChild(new Paragraph());
27                     Run run = para.AppendChild(new Run());
28
29                     foreach (var item in customer.Cast<string>())
30                     {
31                         run.AppendChild(new Text(item));
32                     }
33                 }
34             }
35         }
36     }
37 }

```

The code is shown in a Visual Studio IDE. The Solution Explorer on the right shows the project structure for 'WordDocManagerAddin' (1 of 1 project). The project contains a 'Properties' folder, 'References' (Analyzers, DocumentFormat.OpenXml, Microsoft.CSharp, System, System.Core, WindowsBase), and 'WordDocManager.cs'.



## AL Code



The image displays two screenshots of the Visual Studio Code editor interface, showing AL code for a page extension and a codeunit.

**Top Screenshot:** Shows the code for a page extension named `CustomerListX`. The code is as follows:

```
1 //
2 // Copyright (c) Microsoft Corporation. All rights reserved.
3 // Licensed under the MIT License. See License.txt in the project root for license information.
4 //
5 pageextension 50100 CustomerListX extends "Customer List"
6 {
7     actions
8     {
9         addbefore(NewSalesQuote)
10        {
11            action(CreateCustomerDirectoryWordDocument)
12            {
13                ApplicationArea = All;
14                Caption = 'Create customer directory Word document';
15                Image = Document;
16                Promoted = true;
17                PromotedCategory = Category5;
18                PromotedIsBig = true;
19                PromotedOnly = true;
20            }
21            trigger OnAction();
22            var
23                WordDocManagement: Codeunit WordDocManagement;
24            begin
25                WordDocManagement.CreateCustomerDirectoryWordDocument();
26            end;
27        }
28    }
29 }
```

**Bottom Screenshot:** Shows the code for a codeunit named `WordDocManagement`. The code is as follows:

```
4 //
5 //
6 //
7 //
8 //
9 //
10 //
11 //
12 //
13 //
14 //
15 //
16 //
17 //
18 //
19 //
20 //
21 codeunit 50110 WordDocManagement
22 {
23     // Create the customer list word document by calling
24     // the .net server-side addin
25     procedure CreateCustomerDirectoryWordDocument()
26     var
27         CustomerList: DotNet GenericList1;
28         CustomerDetails: DotNet GenericList1;
29         WordDocManagement: DotNet WordDocManagement;
```

The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying the file structure. The main editor area shows the `target` directory containing an `app.json` file. The code in `app.json` is as follows:

```
20   "idRanges": [
21     {
22       "from": 50100,
23       "to": 50149
24     }
25   ],
26   "screenshots": [],
27   "platform": "1.0.0.0",
28   "showMyCode": true,
29   "target": "OnPrem",
30   "runtime": "7.0"
31 }
```

The status bar at the bottom indicates the current file is `app.json` at line 29, column 3, with 19 characters selected. The encoding is UTF-8 and the language is JSON.

The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying the file structure. The main editor area shows the `WordDocManager.cs` file. The code in `WordDocManager.cs` is as follows:

```
7 namespace WordDocManagerAzureFunc
8 {
9     internal class WordDocManager
10    {
11        // Create a Word document using the OpenXML library
12        internal Stream CreateCustomerDirectoryDocument(List<Customer> customers)
13        {
14            MemoryStream result = new MemoryStream();
15
16            using (WordprocessingDocument wordDocument = WordprocessingDocument.Create(result, WordprocessingDocumentType.Document))
17            {
18                MainDocumentPart mainPart = wordDocument.AddMainDocumentPart();
19
20                mainPart.Document = new Document();
21                Body body = mainPart.Document.AppendChild(new Body());
22
23                foreach (var customer in customers)
24                {
25                    Paragraph para = body.AppendChild(new Paragraph());
26                    Run run = para.AppendChild(new Run());
27
28                    run.AppendChild(new Text(customer.name));
29                    run.AppendChild(new CarriageReturn());
30                    run.AppendChild(new Text(customer.street));
31                    run.AppendChild(new CarriageReturn());
32                    run.AppendChild(new Text(string.Format("{0} {1}", customer.zipCode, customer.city)));
33                    run.AppendChild(new CarriageReturn());
34                }
35            }
36            wordDocument.Close();
37        }
38    }
39 }
```

The status bar at the bottom indicates the current file is `WordDocManager.cs` at line 14, column 54, with 42 characters selected. The encoding is UTF-8 with BOM and the language is C#.

```
public static class CreateCustomerDirectoryWordDoc
{
    private static WordDocManager docManager = new WordDocManager();

    // Azure Function App
    [FunctionName("CreateCustomerDirectoryWordDoc")]
    public static async Task Run(
        [HttpTrigger(AuthorizationLevel.Anonymous, "get", "post", Route = null)] HttpRequest req, ILogger log)
    {
        string requestBody = await new StreamReader(req.Body).ReadToEndAsync();

        // Deserialize the customer list from JSON
        var customers = JsonConvert.DeserializeObject<List<Customer>>(requestBody);

        // Create the actual Word document in a stream
        Stream stream = docManager.CreateCustomerDirectoryDocument(customers);

        // Return the document as a binary stream
        stream.Position = 0;
        byte[] buffer = new byte[stream.Length];
        stream.Read(buffer, 0, (int)stream.Length);
        return new FileContentResult(buffer, "application/octet-stream");
    }
}
```

```
codeunit 50110 WordDocManagement
{
    // Create the customer list word document by calling
    // the Azure Function App
    procedure CreateCustomerDirectoryWordDocument()
    var
        FileName: Text;
        Json: Text;
        InStr: InStream;
    begin
        Json := SerializeCustomersToJson();
        InStr := GetWordDocumentFromService(Json);

        Message('Customer list created');
        FileName := 'CustomerList.docx';
        DownloadFromStream(InStr, 'Customer list document', '', 'All Files (*.*)', FileName);
    end;

    // Call the CreateCustomerDirectoryWordDoc Azure Function App
    // using a HTTP request and get the return Word document in a stream
    local procedure GetWordDocumentFromService(customers: Text) Result: InStream
    var
        Client: HttpClient;
        Response: HttpResponseMessage;
        Headers: HttpHeaders;
        Content: HttpContent;
        LocalURL: label 'http://localhost:7071/api/CreateCustomerDirectoryWordDoc', Locked = true;
        URL: label 'https://worddocmanagerazurefunc.azurewebsites.net/api/CreateCustomerDirectoryWordDoc', Locked = true;
    begin
        Content := Clean();
```

```
Response.Content().ReadAs(Result);
end;

// Serialize the customer list to JSON
local procedure SerializeCustomersToJson() Json: Text
var
    JsonCustomers: JsonArray;
    Customer: Record Customer;
begin
    Customer.FindFirst();
    repeat
        AddCustomer(JsonCustomers, customer);
    until customer.Next() = 0;

    JsonCustomers.WriteTo(Json);
end;

local procedure AddCustomer(JsonCustomers: JsonArray; customer: Record Customer)
var
    JsonCustomer: JsonObject;
begin
    JsonCustomer.Add('name', customer.Name);
    JsonCustomer.Add('street', customer.Address);
    JsonCustomer.Add('zipCode', customer."Post Code");
    JsonCustomer.Add('city', customer.City);
    JsonCustomers.Add(JsonCustomer);
end;
end;
```

```
https://events1.social27.com/msdyn365bclaunchevent2021w1/agenda/player/95778
EXPLORER WordDocManagement.lal X app.json CustomerListX.lal
WordDocManagement.lal
app.json
CustomerListX.lal
ALNOADDIN
snapshots
.vscode
dotNet
app.json
CustomerListX.lal
WordDocManagement.lal
OUTLINE
NPM SCRIPTS
ALNoAddin Ln 21, Col 47 (26 selected) Spaces: 4 UTF-8 CRLF AL
```

```
end;
// Call the CreateCustomerDirectoryWordDoc Azure Function App
// using a HTTP request and get the return Word document in a stream
local procedure GetWordDocumentFromService(customers: Text) Result: InStream
var
    Client: HttpClient;
    Response: HttpResponseMessage;
    Headers: HttpHeaders;
    Content: HttpContent;
    LocalURL: label 'http://localhost:7071/api/CreateCustomerDirectoryWordDoc', Locked = true;
    URL: label 'https://worddocmanagerazurefunc.azurewebsites.net/api/CreateCustomerDirectoryWordDoc', Locked = true;
begin
    Content.Clear();
    Content.WriteFrom(Format(customers));
    Content.GetHeaders(Headers);
    Headers.Remove('Content-Type');
    Headers.Add('Content-Type', 'application/json');

    Client.Post(URL, Content, Response);
    Response.Content().ReadAs(Result);
end;

// Serialize the customer list to JSON
local procedure SerializeCustomersToJson() Json: Text
var
    JsonCustomers: JsonArray;
    Customer: Record Customer;
begin
```


```
https://events1.social27.com/msdyn365bclaunchevent2021w1/agenda/player/95778
EXPLORER WordDocManagement.lal app.json X CustomerListX.lal
WordDocManagement.lal
app.json
CustomerListX.lal
ALNOADDIN
snapshots
.vscode
dotNet
app.json
CustomerListX.lal
WordDocManagement.lal
OUTLINE
NPM SCRIPTS
ALNoAddin Ln 23, Col 21 (19 selected) Spaces: 2 UTF-8 LF JSON
```

```
{
  "id": "63ca2fa4-4f03-4f2b-a480-172fef340d3f",
  "name": "System Application",
  "publisher": "Microsoft",
  "version": "17.0.0.0"
},
"screenshots": [],
"platform": "17.0.0.0",
"showMyCode": true,
"target": "Cloud",
"runtime": "7.0"
}
```

https://events1.social27.com/modyn365bclaunchevent2021w1/agenda/player/95778

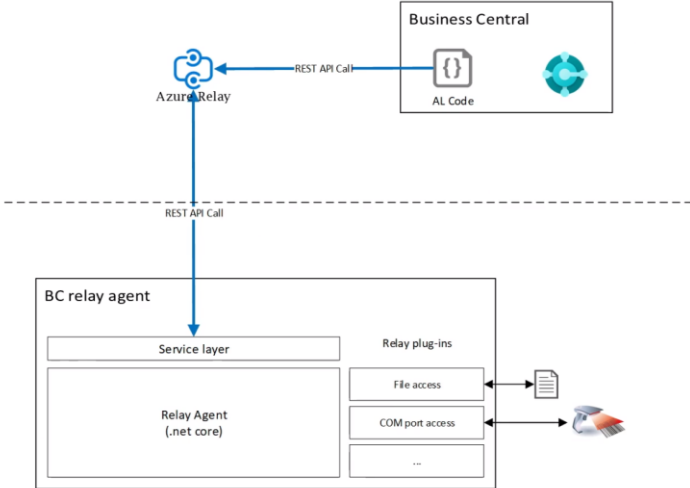
## Additional benefits of the cloud-ready architecture

- Better separation of concerns
- Cleaner and more explicit API
- Independent deployment life cycle
- No server downtime on updates



https://events1.social27.com/modyn365bclaunchevent2021w1/agenda/player/95778


## Accessing hardware or local files from the cloud



```
graph TD; BC[Business Central] -- REST API Call --> AR[Azure Relay]; AR -- REST API Call --> SLS[Service layer]; SLS --> RA[Relay Agent .net core]; RA --- RP[Relay plug-ins]; RP --> FA[File access]; RP --> CPA[COM port access]; RP --> Dots[...];
```

The diagram illustrates the flow of data from Business Central (AL Code) through Azure Relay to a BC relay agent. The BC relay agent consists of a Service layer and a Relay Agent (.net core). The Relay Agent (.net core) uses Relay plug-ins for File access and COM port access, which are connected to hardware or local files.

https://events1.social27.com/msdyn365bclaunchevent2021w1/agenda/player/95778



## Resources

**BCTech on Git Hub:**  
<https://github.com/microsoft/BCTech>

**Azure Functions :**  
<https://azure.microsoft.com/en-us/services/functions/>

**Azure Functions in VS Code:**  
<https://docs.microsoft.com/en-us/azure/azure-functions/functions-develop-vs-code>

**Postman:**  
<https://www.postman.com/>

**Web App Service:**  
<https://azure.microsoft.com/en-us/services/app-service/web/>

