THE MICROSOFT JOURNAL FOR DEVELOPERS

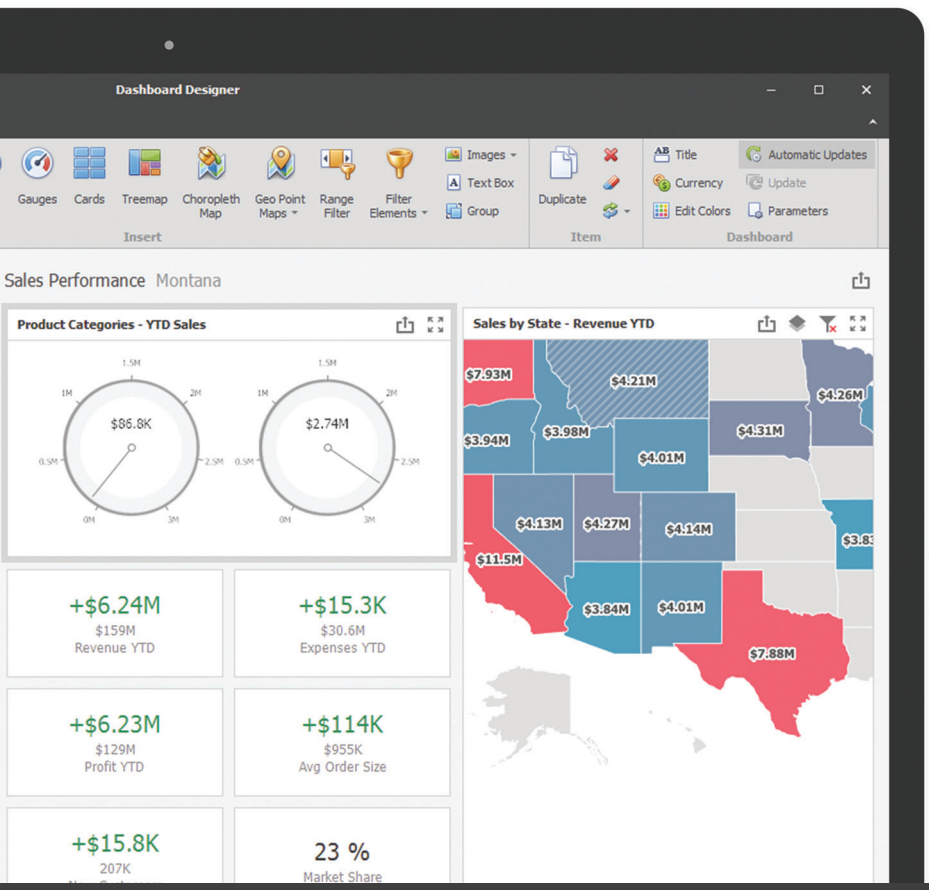
AUGUST 2019 VOL 34 NO 8

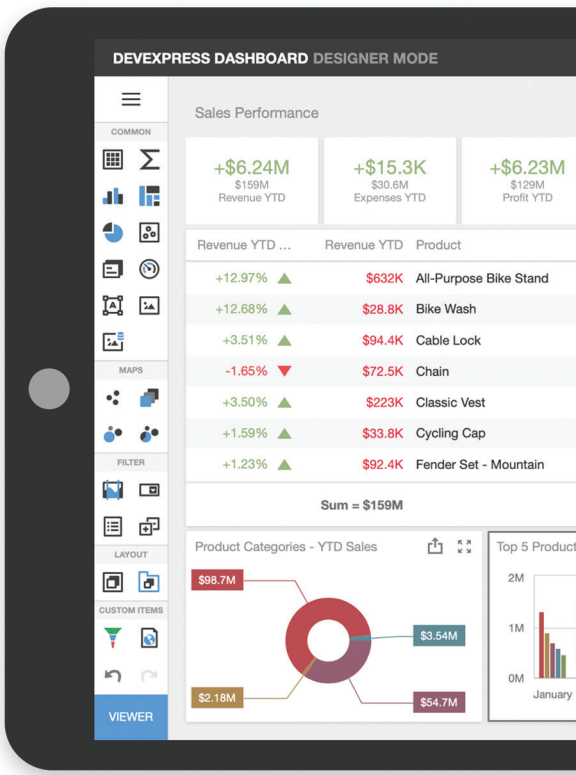
msdn magazine

DevExpress®

Enterprise-Ready Analytics

Go from Zero To Dashboard in Record Time





Get your free
30-day Trial today

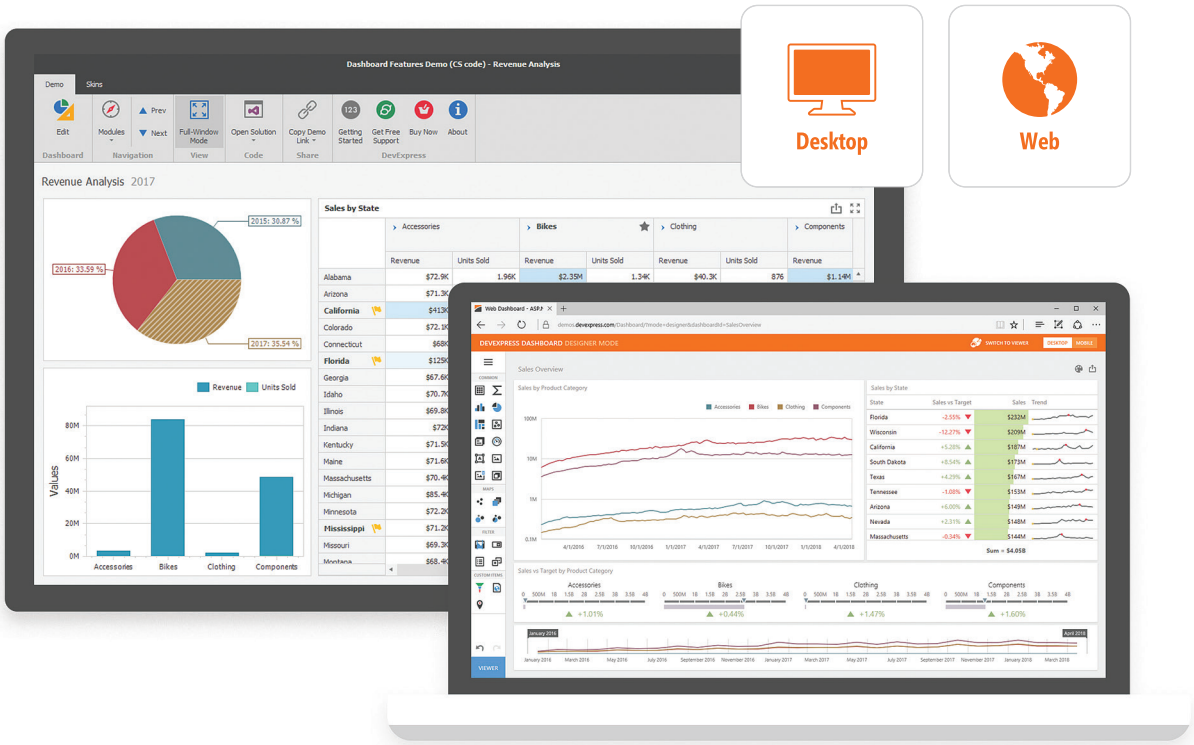
devexpress.com/dashboard

All trademarks or registered trademarks are property of their respective owners.

DevExpress®

DevExpress Dashboard for .NET

Create and distribute royalty-free decision support systems and effortlessly share business intelligence across your entire enterprise.



Data Grid · Pivot Grid · Card View · Charts · Gauges · Treemap · Map · Range Control · Filter Control · Rich Text Editor · Image

With DevExpress Dashboard for .NET, creating insightful and information-rich decision support systems for executives and business users across platforms and devices is a simple matter of selecting the appropriate UI element (Chart, Pivot Table, Data Card, Gauge, TreeMap, Map, Grid or simple Filter elements) and dropping data fields onto corresponding arguments, values, and series. And because DevExpress Dashboard automatically provides the best data visualization option for you, results are immediate, accurate and always relevant.

Learn More Today — Try DevExpress Dashboard Risk Free for 30 days

devexpress.com/dashboard

All trademarks or registered trademarks are property of their respective owners.

THE MICROSOFT JOURNAL FOR DEVELOPERS

AUGUST 2019 VOL 34 NO 8

msdn magazine

IoT and ASP.NET Core.....14

COLUMNS

Microsoft

INFRAGISTICS

Faster Paths to Amazing Experiences

Infragistics Ultimate includes 100+ beautifully styled, high performance grids, charts, & other UI controls, plus visual configuration tooling, rapid prototyping, and usability testing.

Angular | JavaScript / HTML5 | React | ASP.NET | Windows Forms | WPF | Xamarin

Get started today with a free trial:
Infragistics.com/Ultimate

To speak with sales or request a product demo with a solutions consultant call 1.800.231.8588

Delivering amazing experiences for 30 years.

Infragistics Ultimate

- Fastest **grids** & **charts** on the market – any device, any platform
- Build Spreadsheets with Charts in WPF, Windows Forms, Angular & JavaScript
- Get Angular code from Sketch designs with Indigo.Design
- 100% support for .NET Core 3

msdn magazine

AUGUST 2019 VOLUME 34 NUMBER 8

General Manager Jeff Sandquist
Director Dan Fernandez
Editorial Director Jennifer Mashkowski *mmeditor@microsoft.com*
Site Manager Kent Sharkey
Editorial Director, Enterprise Computing Group Scott Bekker
Editor in Chief Michael Desmond
Features Editor Sharon Terdeman
Group Managing Editor Wendy Hernandez
Senior Contributing Editor Dr. James McCaffrey
Contributing Editors Dino Esposito, Frank La Vigne, Julie Lerman, Mark Michaelis, Ted Neward, David S. Platt
Vice President, Art and Brand Design Scott Shultz
Art Director Joshua Gould

CONVERGE360
An 1105 MEDIA company

Chief Revenue Officer
Dan LaBianca

ART STAFF
Creative Director **Jeffrey Langkau**
Senior Graphic Designer **Alan Tao**

PRODUCTION STAFF
Print Production Coordinator **Teresa Antonio**

ADVERTISING AND SALES
Chief Revenue Officer **Dan LaBianca**
Regional Sales Manager **Christopher Kourtoglou**
Advertising Sales Associate **Tanya Egenolf**

ONLINE/DIGITAL MEDIA
Vice President, Digital Strategy **Becky Nagel**
Senior Site Producer, News **Kurt Mackie**
Senior Site Producer **Gladys Rama**
Site Producer, News **David Ramel**
Director, Site Administration **Shane Lee**
Front-End Developer **Anya Smolinski**
Junior Front-End Developer **Casey Rysavy**
Office Manager & Site Assoc. **James Bowling**

CLIENT SERVICES & DEMAND GENERATION
Senior Director **Eric Yoshizuru**
Director, IT (Systems, Networks) **Tracy Cook**
Director, Audience Development & Lead Generation Marketing **Irene Fincher**
Project Manager, Lead Generation Marketing **Mahal Ramos**
Coordinator, Client Services & Demand Generation **Racquel Kylander**

ENTERPRISE COMPUTING GROUP EVENTS
Vice President, Events **Brent Sutton**
Senior Director, Operations **Sara Ross**
Senior Director, Event Marketing **Mallory Bastionell**
Senior Manager, Events **Danielle Potts**

1105 MEDIA INC
YOUR GROWTH, OUR BUSINESS.

Chief Executive Officer
Rajeev Kapur

Chief Financial Officer
Sanjay Tanwani

Chief Technology Officer
Erik A. Lindgren

Executive Vice President
Michael J. Valenti

ID STATEMENT *MSDN Magazine* (ISSN 1528-4859) is published 13 times a year, monthly with a special issue in December by 1105 Media, Inc., 6300 Canoga Avenue, Suite 1150, Woodland Hills, CA 91367. Periodicals postage paid at Woodland Hills, CA 91367 and at additional mailing offices. Annual subscription rates payable in US funds are: U.S. $35.00, International $60.00. Annual digital subscription rates payable in U.S. funds are: U.S. $25.00, International $25.00. Single copies/back issues: U.S. $10, all others $12. Send orders with payment to: *MSDN Magazine*, File 2272, 1801 W.Olympic Blvd., Pasadena, CA 91199-2272, email MSDNmag@1105service.com or call 866-293-3194 or 847-513-6011 for U.S. & Canada; 00-1-847-513-6011 for International, fax 847-763-9564. POSTMASTER: Send address changes to MSDN Magazine, P.O. Box 2166, Skokie, IL 60076. Canada Publications Mail Agreement No: 40612608. Return Undeliverable Canadian Addresses to Circulation Dept. or XPO Returns: P.O. Box 201, Richmond Hill, ON L4B 4R5, Canada.

COPYRIGHT STATEMENT © Copyright 2019 by 1105 Media, Inc. All rights reserved. Printed in the U.S.A. Reproductions in whole or part prohibited except by written permission. Mail requests to "Permissions Editor," c/o MSDN Magazine, 2121 Alton Pkwy., Suite 240, Irvine, CA 92606.

LEGAL DISCLAIMER The information in this magazine has not undergone any formal testing by 1105 Media, Inc. and is distributed without any warranty expressed or implied. Implementation or use of any information contained herein is the reader's sole responsibility. While the information has been reviewed for accuracy, there is no guarantee that the same or similar results may be achieved in all environments. Technical inaccuracies may result from printing errors and/or new developments in the industry.

CORPORATE ADDRESS 1105 Media, Inc.
6300 Canoga Avenue, Suite 1150, Woodland Hills 91367
1105media.com

MEDIA KITS Direct your Media Kit requests to Chief Revenue Officer Dan LaBianca, 972-687-6702 (phone), 972-687-6799 (fax), dlabianca@Converge360.com

REPRINTS For single article reprints (in minimum quantities of 250-500), e-prints, plaques and posters contact: PARS International
Phone: 212-221-9595
E-mail: 1105reprints@parsintl.com
Web: 1105Reprints.com

LIST RENTAL This publication's subscriber list, as well as other lists from 1105 Media, Inc., is available for rental. For more information, please contact our list manager, Jane Long, Merit Direct.
Phone: (913) 685-1301
Email: jlong@meritdirect.com
Web: meritdirect.com/1105

Reaching the Staff

Staff may be reached via e-mail, telephone, fax, or mail.

E-mail: To e-mail any member of the staff, please use the following form: FirstinitialLastname@1105media.com

Irvine Office (weekdays, 9:00 a.m.-5:00 p.m. PT)
Telephone 949-265-1520; Fax 949-265-1528
2121 Alton Pkwy., Suite 240, Irvine, CA 92606

Corporate Office (weekdays, 8:30 a.m.-5:30 p.m. PT)
Telephone 818-814-5200; Fax 818-734-1522
6300 Canoga Ave., Suite 1150, Woodland Hills, CA 91367

The opinions expressed within the articles and other contentsherein do not necessarily express those of the publisher.

Microsoft

BPA WORLDWIDE

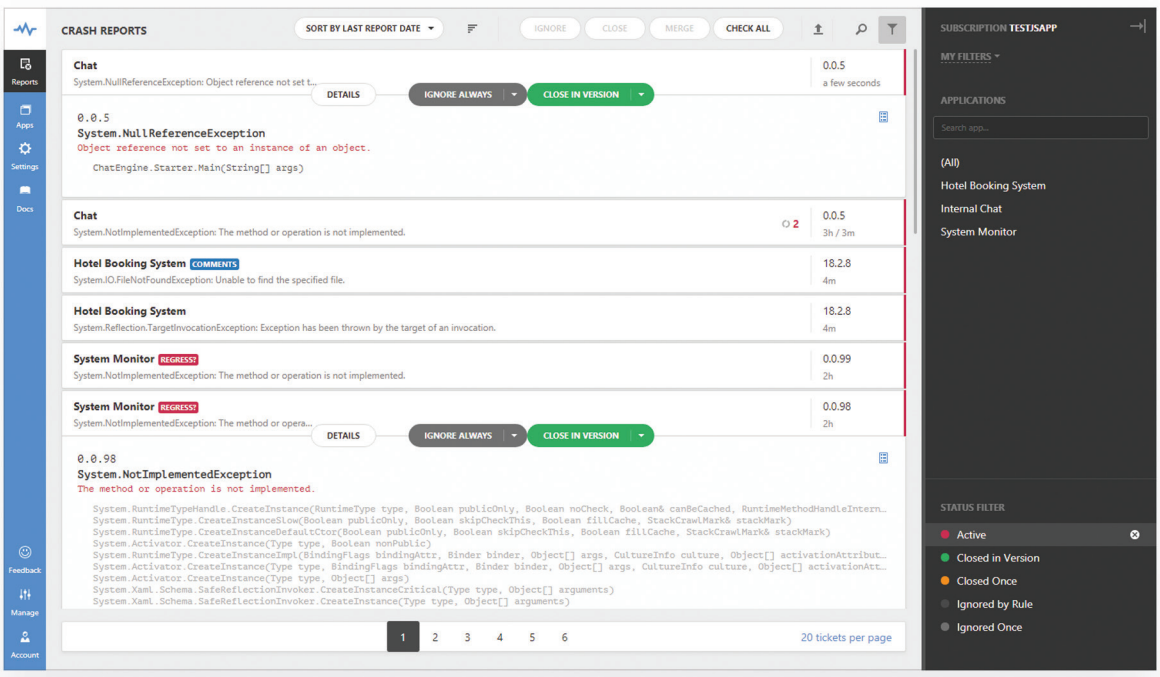
1906 100 2006 AMERICAN BUSINESS MEDIA

WPA Western Publications Association

logify™

Detect Errors and Exceptions
with 24/7 Application Monitoring

Logify's cloud-based application monitoring service allows you to automatically collect app crash events and runtime exceptions. With Logify, you will never have to deal with Visual Studio's® internal exception information. Logify presents all relevant exception data in an easy-to-understand, clutter-free manner. From loaded modules and cookies to browser info, OS build, user activity — Logify will organize results so you can address application issues in the shortest possible time.



Learn More Today — Try Logify Risk Free for 15 days

devexpress.com/logify

DevExpress®

All trademarks or registered trademarks are property of their respective owners.

Editor's Note

MICHAEL DESMOND

The Feynman Technique

Richard Feynman was a famed, Nobel-prize winning nuclear physicist, author and academic who helped introduce a generation to difficult concepts like quantum mechanics and particle physics. His contributions stretched from the Manhattan Project during World War II to his memorable role articulating how the O-rings failed on the ill-fated Space Shuttle Challenger in 1986. But he was notable first for the remarkable curiosity that informed his methods and his career.

I was thinking about Feynman after interviewing *MSDN Magazine* columnist Frank La Vigne about his transition to the field of artificial intelligence and machine learning, and how he grappled with the many foreign concepts in his new field. La Vigne shared something I've heard repeatedly from authors over the years—that writing technical articles is immensely valuable in mastering technical concepts.

"I found that writing about what I learned helped me focus and get a deeper understanding of what I was learning about," La Vigne told me. "This is an idea I later learned was referred to as the Feynman technique."

That technique boils down to four steps: Choose a concept, teach it to a child, identify and address gaps in your knowledge by going back to the source material, and finally, organize and simplify. It's that second step where the magic happens. Complicated language and jargon can act as a crutch. By forcing yourself to articulate concepts in the simplest possible terms, you force yourself to address gaps in knowledge.

I brought this up with another esteemed *MSDN Magazine* columnist, Dr. James McCaffrey. His response: "Writing articles for *MSDN Magazine* is one of the most important things I do with regard to learning any new technology."

For McCaffrey, the process starts with lots of reading ("blog posts, Wikipedia articles, Stack Overflow comments and so on," he says), followed by digging up a Hello World example that he can code and run. After playing around with the code, he'll often write a short blog post about his early discovery work. This, McCaffrey says, often uncovers areas that he didn't quite understand.

Then McCaffrey goes to Notepad ("Yes, Notepad," he says) to spend anywhere from a few hours to a few months trying to reverse engineer the technology. "In most cases, my Notepad version has greatly reduced functionality compared to the real thing, but if I can't implement a technology using Notepad, then I don't fully understand it," he says.

> McCaffrey says the public scrutiny of his work by tens of thousands of skilled developers is a keen motivator.

The final step? Writing an article for *MSDN Magazine*. McCaffrey says the public scrutiny of his work by tens of thousands of skilled developers is a keen motivator. He puts particular emphasis on crafting the opening sentence of each article, striving to concisely articulate the "what" of the topic. "That's hard to do," he says, "and it almost always requires a deep understanding of the topic. The 'whys' and 'hows' come later."

Ultimately, in his interpretation of the Feynman technique, McCaffrey's classroom of children ends up being himself.

"My guiding thought is to try to explain a technology to my former self," McCaffrey says. "That is, myself back in time when I first started the exploration."

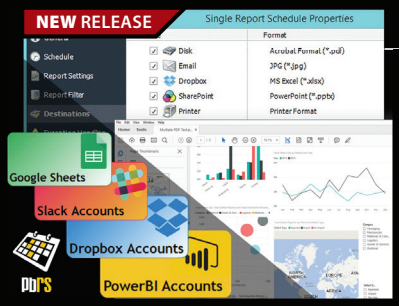
Visit us at msdn.microsoft.com/magazine. Questions, comments or suggestions for *MSDN Magazine*? Send them to the editor: mmeditor@microsoft.com.

© 2019 Microsoft Corporation. All rights reserved.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, you are not permitted to reproduce, store, or introduce into a retrieval system *MSDN Magazine* or any part of *MSDN Magazine*. If you have purchased or have otherwise properly acquired a copy of *MSDN Magazine* in paper format, you are permitted to physically transfer this paper copy in unmodified form. Otherwise, you are not permitted to transmit copies of *MSDN Magazine* (or any part of *MSDN Magazine*) in any form or by any means without the express written permission of Microsoft Corporation.

A listing of Microsoft Corporation trademarks can be found at microsoft.com/library/toolbar/3.0/trademarks/en-us.mspx. Other trademarks or trade names mentioned herein are the property of their respective owners.

MSDN Magazine is published by 1105 Media, Inc. 1105 Media, Inc. is an independent company not affiliated with Microsoft Corporation. Microsoft Corporation is solely responsible for the editorial contents of this magazine. The recommendations and technical guidelines in *MSDN Magazine* are based on specific environments and configurations. These recommendations or guidelines may not apply to dissimilar configurations. Microsoft Corporation does not make any representation or warranty, express or implied, with respect to any code or other information herein and disclaims any liability whatsoever for any use of such code or other information. *MSDN Magazine*, MSDN and Microsoft logos are used by 1105 Media, Inc. under license from owner.



PBRS (Power BI Reports Scheduler) | from $9,811.51

Christiansteven

Date & time Scheduling for Power BI reports with one Power BI License.

- Exports reports to PDF, Excel, Excel Data, Word, PowerPoint, CSV, JPG, HTML, PNG and ePub
- Send reports to email, printer, Slack, Google Sheets, folder, FTP, DropBox & SharePoint
- Uses database queries to automatically populate report filters, email addresses & body text
- Adds flexibility with custom calendars e.g. 4-4-5, holidays, "nth" day of the month, etc.
- Responds instantly by firing off reports when an event occurs e.g. database record is updated

LEADTOOLS Medical Imaging SDKs V20 | from $4,995.00 SRP

LEADTOOLS
THE WORLD LEADER IN IMAGING SDKs

Powerful DICOM, PACS, and HL7 functionality.

- Load, save, edit, annotate & display DICOM Data Sets with support for the latest specifications
- High-level PACS Client and Server components and frameworks
- OEM-ready HTML5 Zero-footprint Viewer with 3D rendering support and DICOM Storage Server
- Medical-specific image processing functions for enhancing 16-bit grayscale images
- Native libraries for .NET, C/C++, HTML5, JavaScript, WinRT, iOS, OS X, Android, Linux, & more

Intellifront BI | from $23,246.35

Christiansteven

Cost-Effective Data Analytics & Business Intelligence.

- Design and serve visually stunning real-time reports using Grids, Charts, Gauges, Pies & more
- Add automatically updating KPI cards with aggregation, rounding, units, goals, colors & glyphs
- Create & assign Reporting, KPIs and Dashboards as consolidated "Canvases" to user groups
- Give your users a simple and intuitive self-service portal for their BI reports and documents
- Secure User Access with Active Directory Integration, Single Sign On & 2-Factor Authentication

DevExpress DXperience 19.1 | from $1,439.99

DevExpress

A comprehensive suite of .NET controls and UI libraries for all major Microsoft dev platforms.

- WinForms – New Sunburst Chart, Office Navigation UX, SVG Office 2019 skins
- WPF – New Gantt control, improved Data Filtering UX and App Theme Designer
- ASP.NET & MVC – New Adaptive Layouts, improved Rich Text Editor and Spreadsheet
- Reporting – Vertical Band support, Free-hand drawing and improved Report wizards
- JavaScript – New HTML/Markdown WYSIWYG editor, Improved Grid and TreeList performance

© 1996-2019 ComponentSource. All Rights Reserved. All prices correct at the time of press. Online prices may vary from those shown due to daily fluctuations & online discounts.

We accept purchase orders.
Contact us to apply for a credit account.

USA
ComponentSource
650 Claremore Prof Way
Suite 100
Woodstock
GA 30188-5188
USA

Europe (UK)
ComponentSource
The White Building
33 King's Road
Reading, Berkshire
RG1 3AR
United Kingdom

Europe (Ireland)
ComponentSource
Unit 1E & 1F
Core B, Block 71
The Plaza, Park West
Dublin 12
Ireland

Asia Pacific
ComponentSource
7F Kojimachi Ichihara Bldg
1-1-8 Hirakawa-cho
Chiyoda-ku
Tokyo, 102-0093
Japan

Sales Hotline - US & Canada:
(888) 850-9911
www.componentsource.com

GSA Schedule Contract GS-35F-0188R

JULIE LERMAN

Cross-Platform EF6 with .NET Core 3.0!

Though Entity Framework Core has been available for a few years now, there are still myriad production apps out there using EF6. And even with all of the innovation going into EF Core, there's no reason to port working, stable production code to EF Core if you're not planning to change the EF6 logic and have no need to take advantage of the new capabilities or improved performance of EF Core. Thanks to its open source nature, EF6 is still being maintained and even tweaked by Microsoft and the community. Yet, many developers and teams who don't want to mess with their EF6 functionality would like to port their software from .NET to .NET Core to take advantage of the many .NET Core benefits, including its cross-platform capabilities and many innovations.

It has been possible to encapsulate EF6 logic into an ASP.NET Web API and access that from .NET Core applications. But that solution won't allow you to benefit from the features of ASP.NET Core APIs or the innovations to Windows desktop apps as they move to .NET Core 3.0. Luckily, with the upcoming releases of EF 6.3 and .NET Core 3.0, you'll be able to "have your cake and eat it, too." Not only will EF6.3 continue to run on .NET Framework, it will also work with .NET Core 3.0. This is possible because in addition to running on .NET 4.0 and .NET 4.5, EF6.3 is cross-compiled to target .NET Standard 2.1.

> I'll build a very simple project with the most simplistic model—enough to see EF6 working on my MacBook.

In this article, using the latest preview version of EF6.3 and ASP.NET Core 3.0, I'll go all in on trying out EF6.3 in a cross-platform scenario—creating a new ASP.NET Core 3.0 API using EF6.3. On my MacBook! In macOS! Using Visual Studio Code! And then deploying it to a Linux-based Docker container! Not one of these tasks was possible with EF6 until now. You can still use EF6.3 in .NET Framework apps in Visual Studio starting with version 2017 v15.3, although for .NET Core 3.0 apps you'll need preview versions of Visual Studio 2019 that support .NET Core 3.0.

The EF6.3 preview does have a few limitations with .NET Core 3.0 that are expected to be sorted out by the time it's released. Current limitations include:

- .NET Core 3.0 can't be used with the EF Designer in Visual Studio yet.
- Migration commands don't yet work with .NET Core 3.0 projects.
- The only working provider at the time of writing is SQL Server.

You may find the discussion tied to the announcement blog post useful (bit.ly/2F9xtDt), but keep in mind that as the release gets closer, many of these issues should be resolved.

Even with these limitations, my real interest is a proof of concept—witnessing EF6.3 on a non-Windows machine. So, I'll build a very simple project with the most simplistic model—enough to see it working on my MacBook. And because I'm working on macOS and the only database provider currently available is SQL Server, I get a great excuse to use SQL Server for Linux in a Docker container to persist the data.

Preparing the Base API Project

Installing .NET Core 3.0 and creating an ASP.NET Core API remains as it's always been. I'll relay these steps briefly for those of you who haven't done it yet.

Start by ensuring you have .NET Core 3.0 installed on your machine. As I'm writing this in early June 2019, the latest version is Preview 6, released on June 12, 2019. The installation page for .NET Core 3.0 is bit.ly/2KoSOxh. You want to install the SDK package, which includes not only the SDK but also the .NET Core and ASP.NET Core runtimes. Once it's installed, the command *dotnet --version* will list the latest version, while *dotnet --list-sdks* will show all of the versions on your system.

Next, create a new folder for your project (I called mine "coreapi") and be sure you're in that folder at the command line. Then type the CLI command: *dotnet new webapi*. This will create a new ASP.NET Core API project in the folder using C# as the default language and, by default, the latest version of .NET Core installed on your machine.

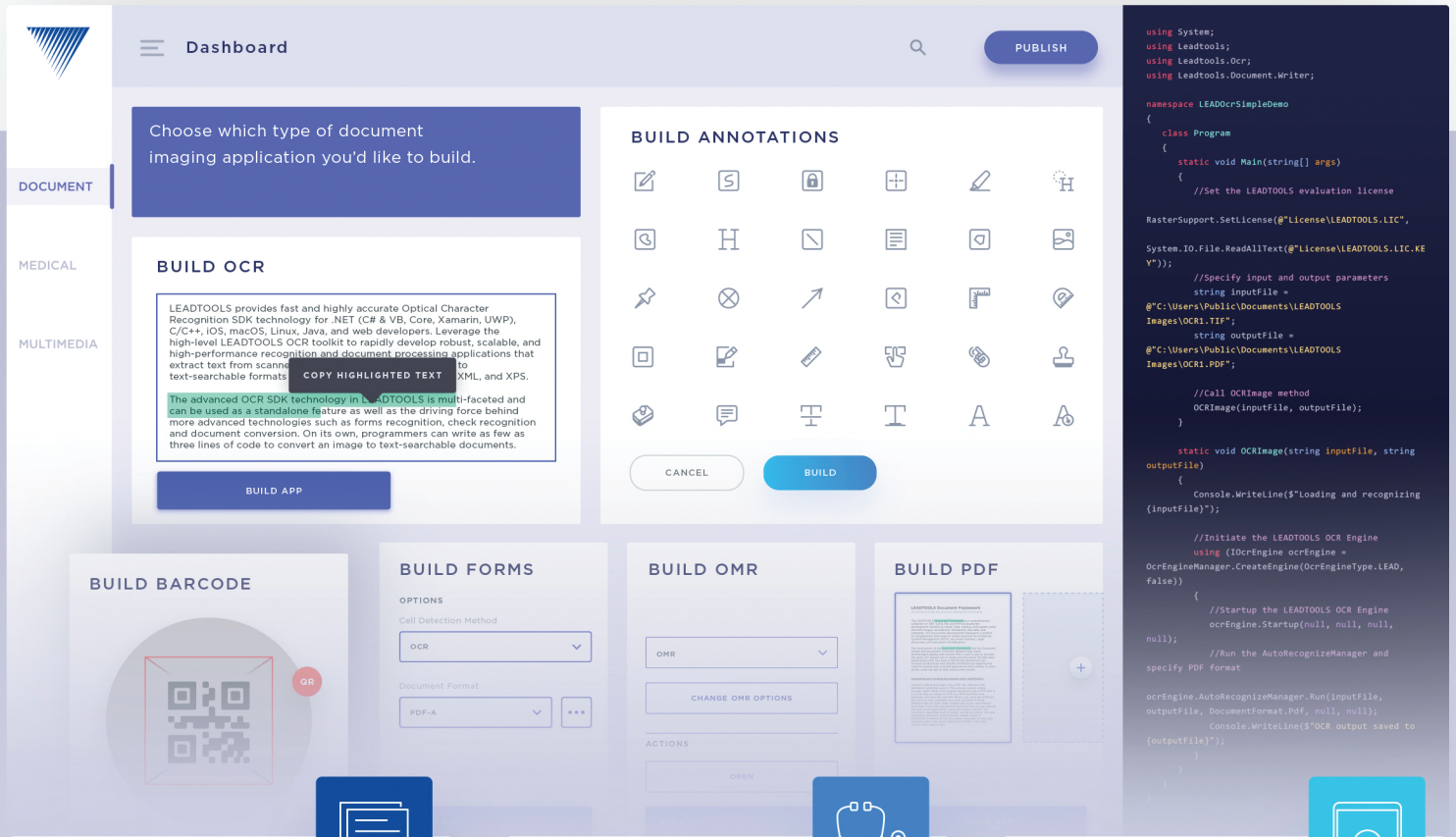
As I'm working on my MacBook, there's a way I can immediately experience the cross-platform support—by using Visual Studio Code for my development environment. I have a shortcut that allows me to type *code .* at the command line in my

Some of the technologies discussed in this article are in preview. All information is subject to change.

Code download available at msdn.com/magazine/0819magcode.

LEADTOOLS®

BUILD BETTER APPS WITH LEADTOOLS



DOCUMENT

- Document Viewer & Converter
- OCR, MICR, OMR & ICR
- 1D & 2D Barcode
- Forms Recognition
- PDF, DOCX, HTML, SVG, RTF, TXT
- Annotations & TWAIN

MEDICAL

- DICOM, CCOW & HL7
- PACS Client & Server Framework
- DICOMWeb (WADO)
- Web & Desktop Viewers
- Image Processing & Annotations
- Medical 3D (MPR, MIP, VRT)

MULTIMEDIA

- Play, Capture, Convert & DVR
- Media Streaming
- MPEG-2 TS, RTSP, HTML5 & more
- OGG, FLV, ISO, AVI, WebM & more
- Audio & Video Processing
- DirectShow & Media Foundation

Microsoft .NET · Xamarin · Windows · Java · Linux · iOS · macOS · Android · HTML5

Get Started Today

DOWNLOAD OUR FREE EVALUATION

LEADTOOLS.COM

project folder to start VS Code with that folder opened. You can also just start VS Code and open the project folder.

The essential assets of the project were created by the template, including a default ValuesController. As you can see in **Figure 1**, the project file targets .NET Core 3.0. There's no longer a need to explicitly reference any of the ASP.NET Core packages in csproj with this version of .NET Core, which is why ItemGroup is empty.

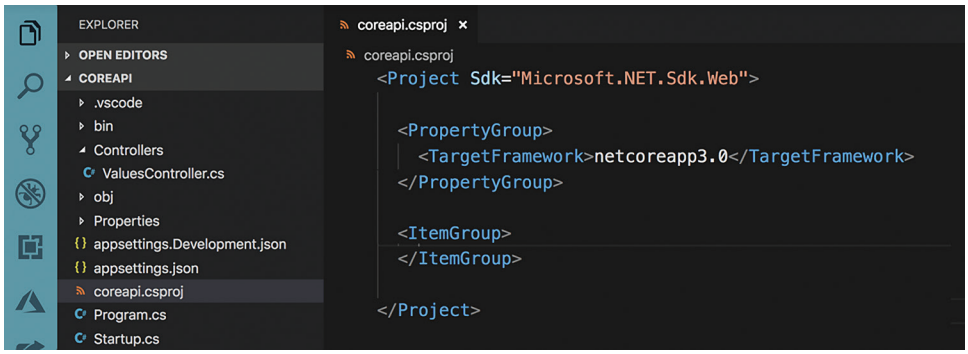


Figure 1 **The coreapi Project and Contents of Its csproj File**

Adding EF6.3 to the Project

You can add EF6.3 directly to the csproj file by including the following package reference within the ItemGroup section:

```
<PackageReference Include="EntityFramework" Version="6.3.0-preview6-19304-03" />
```

Note that this references the latest preview version available as I'm working on this article. Check the NuGet page for Entity Framework (bit.ly/2RgToOl) to determine the current version. By the time you read this article, EF6.3 may have already been fully released!

I'll need some data to store, so I created a tiny class called Human:

```
public class Human {
  public int HumanId { get; set; }
  public string Name { get; set; }
}
```

Note that namespaces and using statements are included in the download example, but not in the code listings here.

To persist the data, I created a DbContext class called HumanContext:

```
public class HumanContext : DbContext {
  public HumanContext (string connectionString) : base (connectionString)
  {
    Database.SetInitializer<HumanContext> (new HumanInitializer ());
  }
  public DbSet<Human> Humans { get; set; }
  protected override void OnModelCreating (DbModelBuilder modelBuilder
  {
    modelBuilder.Entity<Human> ().ToTable ("Humans");
  }
}
```

The constructor expects a connection string to be passed in.

To prevent EF from completely failing at pluralizing the word Human to Humen (!), I've used a mapping to force the table name to be Humans. Additionally, in the constructor, I'm setting the database initializer to a custom initializer, HumanInitializer, which will seed the test database with a few romantic humans if the model changes. In case you've forgotten your EF6 lessons, I'll remind you that this initializer will also create the database if it doesn't exist yet. Here's the HumanInitializer class:

```
public class HumanInitializer :
DropCreateDatabaseIfModelChanges<HumanContext>
{
  protected override void Seed (HumanContext context)
  {
    context.Humans.AddRange (new Human[] {
      new Human { Name = "Juliette" },
      new Human { Name = "Romeo" }
    });
  }
}
```

Wiring Up ASP.NET Core and EF6

When using EF6.3 in ASP.NET Core, there are a few advantages from EF Core that you'll miss out on. One is the integrated dependency injection. EF Core can tie itself into ASP.NET Core services using its AddDbContext extension method. This allows ASP.NET Core to inject object instances of a DbContext on demand into classes that need them, such as a controller class. But the AddDbContext method isn't available without EF Core. Moreover, the EF Core database providers, such as Microsoft.EntityFrameworkCore.SqlServer, have extension methods that allow you to add details about the provider to the AddDbContext method. For example, SqlServer provides a UseSqlServer method to which you can supply options such as the connection string.

While these methods won't be available when using EF6.3, you can still wire up the HumanContext to ASP.NET Core's dependency injection services and let it know about the provider and the connection string. You just have to use different code, which goes in the same place as the EF Core code—inside the ConfigureServices method of the API's startup class.

The IServiceCollection Add method (along with variations such as AddScoped) allow you to add any class to the services. Then you can pass in the instructions of what to do when, at runtime, that object is needed.

> As of Preview 6 of EF6.3, if no provider is specified, EF will use System.Data.SqlClient.

This code then says to keep an eye out for places where a HumanContext object is required and, in response, instantiate a new HumanContext, passing in a connection string found in appsettings.json:

```
services.AddScoped<HumanContext>
            (serviceProvider => new HumanContext (Configuration["Connection"]));
```

This takes care of the dependency injection for HumanContext.

What about making up for the loss of the UseSqlServer method to specify the database provider and connection string? As of Preview 6 of EF6.3, if no provider is specified, EF will use System.Data.SqlClient. So, because I'll be using SQL Server, there's no need to add any more code. SQL Server will be the default provider in this case.

File Format APIs

Open, Create, Convert, Print and Save files from your applications!

Try risk free - 30 day trial

ASPOSE

Download a Free Trial at

https://downloads.aspose.com

Aspose.Words

Create, edit, convert or print Word documents (DOC, DOCX, RTF etc.) in your .NET, Java and Android applications.

Aspose.Cells

Develop high performance .NET, Java and Android applications to Create, Edit or Convert Excel worksheets (XLS, XLSX, ODS etc).

Aspose.Pdf

Manipulate PDF file formats (PDF, PDF/A, XPS etc.) using our native APIs for .NET, Java and Android platforms.

Aspose.Slides

Create, edit or convert PowerPoint presentations (PPT, PPTX, ODP etc.) in your .NET, Java and Android applications.

Aspose.Email

Create, Edit or Convert Outlook Email file formats (MSG, PST, EML etc.) and popular network protocols.

Aspose.BarCode

Generate or recognize barcodes (Code128, PDF417, Postnet etc.) using our native APIs for .NET and Java.

Aspose.Imaging

Deliver efficient applications to Create, Draw, Manipulate or Convert image file formats.

Aspose.Tasks

Develop high performance apps to Create, Edit or Convert Microsoft Project® document formats.

- Aspose.Diagram
- Aspose.Note
- Aspose.3D
- Aspose.CAD
- Aspose.HTML
- Aspose.GIS

Americas: +1 903 306 1676 EMEA: +44 141 628 8900 Oceania: +61 2 8006 6987
sales@asposeptyltd.com

Figure 2 **The HumanController Class**

```
[Route ("api/[controller]")]
[ApiController]
public class HumanController : ControllerBase
{
  HumanContext _context;
  public HumanController (HumanContext context)
  {
    _context = context;
  }

  [HttpGet]
  public ActionResult<IEnumerable<Human>> Get ()
  {
   return _context.Humans.ToList ();
  }
}
```

The last piece of this wiring is to get the connection string into appsettings.json so the Configuration["Connection"] method can read it:

```
"Connection":"Server=localhost,1601;Database=Humans;User Id=sa;Password=P@ssword1"
```

Recall that earlier I said that because I'm on my MacBook, I'll be using SQL Server for Linux in a Docker container as the database. When I ran the container, I specified 1601 as the port on my machine from which to expose the database server. Here's the docker command I used to run the container:

```
docker run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=P@ssword1'
  -p 1601:1433 -d --name SQLforEF6
  mcr.microsoft.com/mssql/server:2017-latest
```

While it's possible to put Windows-dependent apps into Windows containers, Linux containers are a lot easier to work with.

Note that this command needs to be on a single line. If you want to learn more about using SQL Server in a container, check out my July 2017 article, "On-the-Fly SQL Servers with Docker," at msdn.com/magazine/mt784660. One difference from that article is that I'm now pointing to the new home of Microsoft's Docker containers in the Microsoft Container Registry, the "mcr" in the URL of the image that the docker run command is targeting.

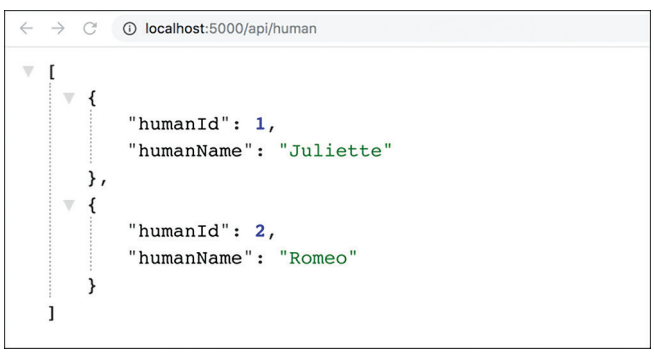


Figure 3 **The Results of the coreapi's Get Method**

Adding a Controller That Uses EF6

The controller that was created by the template is a good way to test that your API is working, but it only builds and returns strings. To test out the database interaction, add a new HumanController class as shown in **Figure 2**. For this proof of concept, all you really need is a single HttpGet method to get some data. The Seed method in HumanInitializer will take care of inserting data.

Note that the first time in an application instance that EF tries to perform some interaction with the database is when the initialization I defined in the HumanContext constructor will get triggered. Therefore, the database won't get created and seeded until I try to run this method. Because this is a simple demo, I'm not worrying about the possibility of having this application on multiple servers, which could cause a conflict if they were trying to run the initialization and seeding code concurrently. But it's always a side effect to keep in mind when allowing an application to be responsible for database creation and seeding.

Running the API

Finally, with everything in place, you can run or debug the API. I usually start by running it from the command line with *dotnet run*. Once it's running, I like to first browse to the values controller at *localhost:5000/api/values* to verify that the API itself is working. This should output "value1" and "value2" in the browser. Then I hit the database-dependent API at *localhost:5000/api/human*. If all goes well, the API will take a little time on its very first run to create the Humans database and create and seed the Humans table. Once that's done, it should output the Ids and names of the two Humans, as shown in **Figure 3**.

Checking Out the Containerized Database

This is proof enough that the database was created and seeded. However, I always feel better if I double-check the database, especially when it's in a containerized server.

Calling the *docker ps* command to list running containers proves that the container is indeed running:

```
   ~ docker ps
CONTAINER ID  IMAGE                                        COMMAND
4bfc01930095  mcr.microsoft.com/mssql/server:2017-latest   "/opt/mssql/bin/sqls…"

CREATED             STATUS              PORTS                    NAMES
23 hours ago        Up 13 hours         0.0.0.0:1601->1433/tcp   SQLforEF6
```

The VS Code Docker extension's Docker Explorer is another way to see that my SQLforEF6 container is running.

And by using Azure Data Studio, I can connect to that container's SQL Server instance to explore the database and its data (see **Figure 4**). For more information on this tool, see my earlier article on Azure Data Studio at msdn.com/magazine/mt832858.

Deploying the EF6.3-Dependent API to Docker for Linux

While it's possible to put Windows-dependent apps into Windows containers, Linux containers are a lot easier to work with. And because EF6.3 can now run cross-platform, I'll create a Linux-based image of the API.

In addition to creating the image, I'll create a docker-compose file to run a container based on this image and enable it to communicate with SQL Server in a container. My three-part series in

DocuVieware

Universal HTML5 and Document Management Kit

Easy integration

Full support for custom snap-in

Zero-footprint solution

Fully customizable UI

Mobile devices optimization

Fast & crystal-clear rendering

Check the New Features and the Online Demos
60-day Free Trial Support Included at **www.docuvieware.com**

Powered by **GdPicture**.NET

DocuVieware is an ORPALIS imaging technologies product

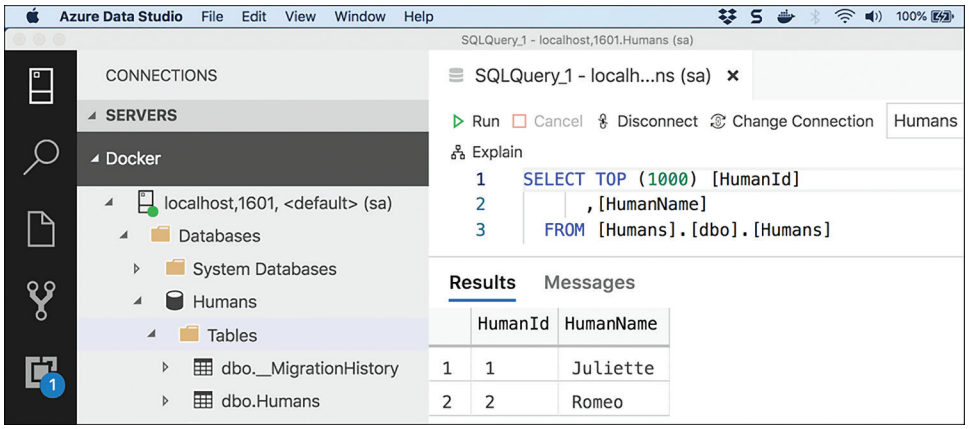


Figure 4 **Exploring the New Database in Azure Data Studio**

April, May and June 2019 issues of this magazine dig into how all of this works, so here I'll just relay the highlights. You can also see all of the code in the sample download.

Again, the VS Code Docker extension helps with most of this effort. By pressing F1 and then typing Docker, you'll find a number of commands that the extension provides. Choose "Docker: Add Docker Files to Workspace." Then when prompted, choose ASP.NET Core as the application platform. Follow the rest of the prompts by choosing Linux as the OS and the default port 80.

The template-generated Dockerfile needs an update, though, because I'm targeting Preview 6 of .NET Core 3.0. I had to change the FROM image targets to pull Preview 6 versions directly from the Microsoft container registry (MCR). Change the first four lines of the Dockerfile as follows:

```
FROM mcr.microsoft.com/dotnet/core/aspnet:3.0.0-preview6 AS base
WORKDIR /app
EXPOSE 80
FROM mcr.microsoft.com/dotnet/core/sdk:3.0.100-preview6 AS build
```

Next, I added a docker-compose file to orchestrate building this new image, running it in a container and also running another SQL Server container to which the API can talk. If you haven't read my earlier articles yet, this is important so that the API container knows how to find the database container.

Figure 5 shows the docker-compose.yml file I added to my project.

The last change to make is the connection string in the API's appsettings.json file. Currently it's pointing to localhost, 1601. But the ASP.NET API inside the new container doesn't have the server on its own localhost. However, if you change the server name to match the service name (db) in the docker-compose file, Docker will ensure that the API container can find the database container.

Here's the new connection listing in appsettings.json:

```
"Connection":"Server=db;Database=Humans;User Id=sa;Password=P@ssword1"
```

That's it. Now you can run the containers using docker-compose. Just right-click on the docker-compose.yml file in the document explorer and, thanks to the Docker extension, one of the options is Compose Up. Choose that and wait for it to complete. It may take another 30 seconds or so before SQL Server finishes setting up its system databases. Then you can browse to the API at localhost:80/api/human. Give it a few extra seconds to create and seed the Humans database. SQL Server is a little slow at doing that, but when it's done, you can browse to the API and see the same output as in **Figure 4**! You can then right-click that yml file again and choose Compose Down to remove the new containers.

> If I change the server name to match the service name (db) in the docker-compose file, Docker will ensure that the API can find the container.

Figure 5 **The docker-compose.yml File to Run Both the API and Database Containers**

```
version: '3.4'

services:
  coreapi:
    image: ${DOCKER_REGISTRY-}api
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - 80:80
  db:
    image: mcr.microsoft.com/mssql/server
    environment:
      SA_PASSWORD: "P@ssword1"
      ACCEPT_EULA: "Y"
    ports:
      - "1601:1433"
```

EF6.3 Without Windows!

While I haven't taken EF6.3 for a real ride, I am amazed and impressed that I was able to do this entire proof of concept in a non-Windows environment. I wouldn't begin a new project using EF6.3. All new work should use EF Core, which has so many advantages, especially with its cross-platform reach. But if you have production code happily using EF6 and want dependent apps to benefit from .NET Core features—whether for cross-platform capabilities or other great functionality—you'll now be able to have your EF6 cake and eat it, too. ■

Julie Lerman *is a Microsoft Regional Director, Microsoft MVP, Docker Captain and software team coach who lives in the hills of Vermont. You can find her presenting on data access and other topics at user groups and conferences around the world. She blogs at thedatafarm.com/blog and is the author of "Programming Entity Framework," as well as a Code First and a DbContext edition, all from O'Reilly Media. Follow her on Twitter: @julielerman and see her Pluralsight courses at bit.ly/PS-Julie.*

Thanks to the following Microsoft technical experts for reviewing this article: *Brice Lambson, Diego Vega*

msdn magazine VP

MSDN MAGAZINE VENDOR PROFILE

Xceed
pari passu

WPF, .NET, Open Source, Java Script: Time for a Reality Check?

By Regent "Gabriel" Gravel,
CEO, Xceed Software

Q First thing first: for those who wouldn't know, who is Xceed Software?

A We like to think of ourselves as the biggest small guy in the component space. This has huge benefits for our partners/clients, because we are accessible and extremely agile. If this is the first you hear from us, well, we have been developing .NET and WPF components since 1994, and our clients reside all over the world. We focus on code; marketing spend is a by-product. We are experts and not generalists. You will see this by our product line that is concentrated on 25 key products as opposed to the branched-out offerings of other vendors. What else can I say? We have been recognized for one of the most powerful DataGrid on the market for WPF, and we are leveraging from this expertise to build our next top products. So, stay tuned! It's getting exciting.

Q Many of your clients describe Xceed as "the" experts in WPF: what do you think of the potential for WPF in the next few years?

A Several years ago, many had predicted the imminent death of WPF. There was even a time when JavaScript often seemed the only way to go for many development projects: does this reflect the reality in the market, now? Not if I refer to our experience at Xceed. Two years ago, we thought we had reached the top in terms of WPF sales (DataGrid, ToolKit, etc.), but we were wrong: new users have kept ordering our WPF products week after week, including some who had gone the JavaScript way with unsatisfactory results. Up to a certain extent, this is understandable. JavaScript today is not where it was 3 or 4 years ago. WPF will be alive and kicking for a couple more years.

Q How can you conciliate the fact that your company is recognized for its leadership in WPF with the JavaScript world?

A At Xceed, we got into the JavaScript "boat" only recently: to be transparent, we were not sure if we were too late to the party. But we're not! On the contrary. You see, JavaScript is a lot more mature and structured now, more "disciplined"(think of TrueScript, for instance). So, we're coming out with the brand new, "fresh" technology for our clients! We're not going to be stuck with "old" (yeah, already!) inefficient code that we would have been if we had developed our products a few years ago. Naturally, the 12-year+ experience we gained in developing our WPF DataGrid has allowed us to avoid many pitfalls and conceive a much better DataGrid in JavaScript! All this to say that we will soon offer both our praised DataGrid for WPF and our new DataGrid JavaScript version. Definitely a winning combination!

Q What about Open-source software?

A There are no doubts that there are plenty of good programs available in the Open-source platform. But it does have its limitation in a business context. Businesspeople I meet often express serious concerns such as the risk for potential viruses or malware, weak or inexistent technical support as well as legal issues concerning actual property rights, licenses and restrictions. Accountants, compliance officers, governments are just a few examples of people who may be alarmed by those red flags: we believe extreme caution should be exercised. Which is why many organizations opt for established products that have gone through rigorous and documented testing, and that are backed by a tech support team. Think of it as an added insurance policy for these big companies.

To access your 45-day full version trial, visit → www.xceed.com

Cross-Platform IoT Programming with .NET Core 3.0

Dawid Borycki

Microsoft Build 2019 brought exciting news for .NET developers: .NET Core 3.0 now supports C# 8.0, Windows Desktop and IoT, so you can use your existing .NET skills to develop cross-platform apps for smart devices. In this article I'll show you how to create a .NET Core app for Raspberry Pi 2/3 with the Sense HAT add-on board. The app will obtain various sensor readings, and the most recent readings will be accessible through the ASP.NET Core Web API service. I'll create the simple UI for this service with Swagger (**Figure 1**), which will let you easily interact with the IoT device. Besides getting data from the device, you'll also be able to remotely change the color of the Sense HAT's LED array (**Figure 2**). The companion code is available through my GitHub page: bit.ly/2WCj0G2.

Some of the technologies discussed here are still in preview. This article was tested with version 0.1.0-prerelease.19352.2 of the Iot.Device.Bindings. Some of the used APIs might have been changed. Please refer to github.com/dotnet/iot for the latest info.

This article discusses:

- Setting up a Raspberry Pi 2/3
- Accessing Sense HAT components using IoT.Device.Bindings
- Creating a console IoT app
- Deploying the app to your device
- Exposing sensor readings and device control through Web API

Technologies discussed:

Windows 10 IoT, C#, .NET Core 3.0, ASP.NET Core Web API, Visual Studio 2019

Code download available at:

bit.ly/2WCj0G2

My Device

I started by setting up my IoT device, comprising Raspberry Pi 2 (or RPi2 for short) with the Sense HAT add-on board (see the right side of **Figure 2**). RPi2 is a popular single-board computer that can run the Linux or Windows 10 IoT Core operating systems. You can get this device from, for example, adafruit.com. The Sense HAT add-on board is equipped with several sensors, including thermometer, barometer, magnetometer, gyroscope, and accelerometer. Additionally, Sense HAT has 64 RGB LEDs you can use as an indicator or a low-resolution screen. Sense HAT is easily attachable to the RPi2, so you can quickly access sensor readings without any soldering. To power up the RPi2 I used a USB cable, then connected to the local Wi-Fi network using a USB dongle (because RPi2, unlike RPi3, doesn't have the onboard Wi-Fi module).

After setting up the hardware, I installed the Windows 10 IoT Core. The full instructions are available at bit.ly/2Ic1Ew1. Briefly, you flash the OS onto the microSD card. This can be done easily with Windows 10 Core Dashboard (bit.ly/2WXm76). Once the dashboard is installed, go to the Set up a new device tab and choose the device type, OS Build, Device name, Administrator password and Wi-Fi

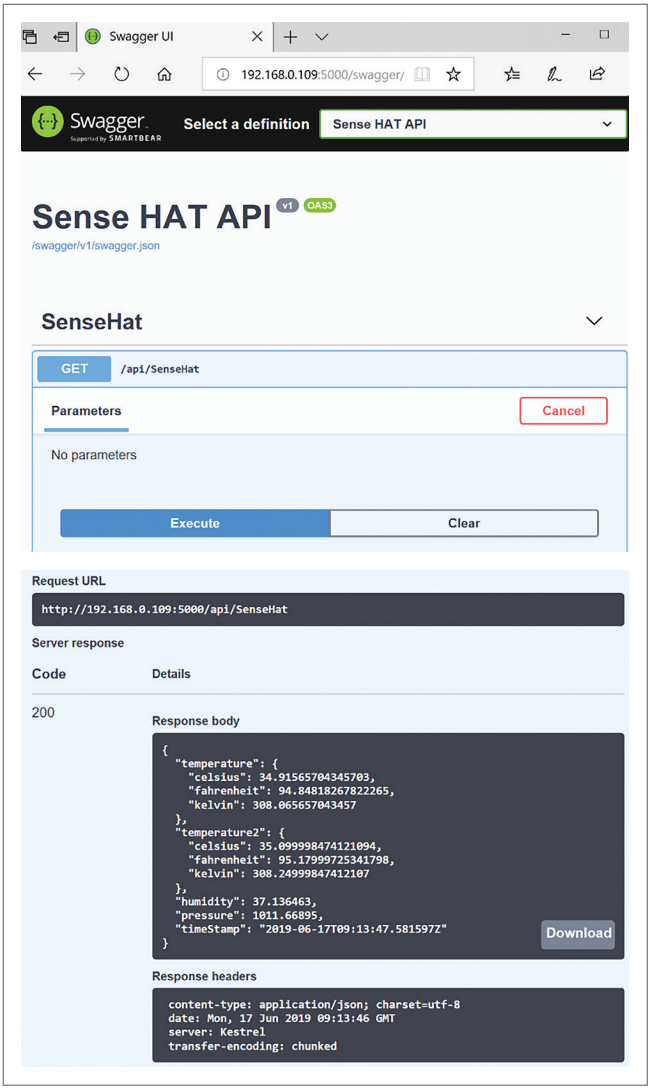


Figure 1 **Obtaining Sensor Readings from the IoT Device Running .NET Core 3.0 App Through the Web API**

connection available to your PC. Accept the software license terms and click Download and install. When this process is finished, insert the microSD card into the IoT device and power it up. You'll soon see your device as an entry under the dashboard's My devices tab.

Common Library

Before starting the actual implementation, I installed .NET Core 3.0 Preview 5. I then opened Visual Studio 2019 and created the new project using the Class Library (.NET Core) template. I set the project and solution names to SenseHat.DotNetCore.Common and SenseHat.DotNetCore, respectively. Then I installed the Iot.Device.Bindings NuGet package (github.com/dotnet/iot) by invoking the following command in the Package Manager Console (bit.ly/2KRvCHj):

```
Install-Package IoT.Device.Bindings -PreRelease
  -Source https://dotnetfeed.blob.core.windows.net/dotnet-iot/index.json
```

IoT.Device.Bindings is an open source .NET Core implementation for popular IoT hardware components that lets you quickly implement .NET Core apps for IoT devices. The Sense HAT binding, which is most relevant here, is available under the src/devices/SenseHat subfolder. A quick glance at this code shows that IoT.Device.Bindings uses the I²C bus to access Sense HAT components. Before showing you how to use IoT.Device.Bindings for the Sense HAT, I first implemented the simple POCO class, SensorReadings (SenseHat.DotNetCore.Common/Sensors/SensorReadings.cs):

```
public class SensorReadings
{
  public Temperature Temperature { get; set; }
  public Temperature Temperature2 { get; set; }
  public float Humidity { get; set; }
  public float Pressure { get; set; }
  public DateTime TimeStamp { get; } = DateTime.UtcNow;
}
```

> RPi2 is a popular single-board computer that can run the Linux or Windows 10 IoT Core operating systems.

This class has five public members. Four members store the actual sensor readings, two temperature readings, plus humidity and pressure. There are two temperatures because Sense HAT has

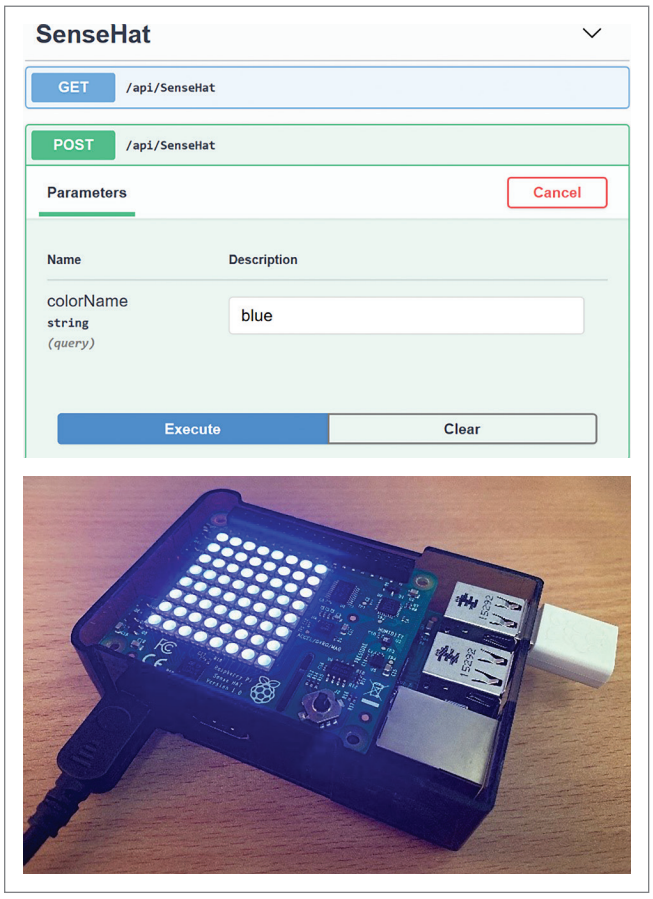


Figure 2 **Remote Control of the IoT Device (Raspberry Pi 2 with the Sense HAT Add-On Board)**

two temperature sensors, one embedded into the LPS25H pressure sensor (bit.ly/2MiYZEl) and the other in the HTS221 humidity sensor (bit.ly/2HMP9GO). Temperatures are represented by the Iot.Units.Temperature struct from Iot.Device.Bindings. This struct doesn't have any public constructors but can be instantiated using one of the static methods: FromCelsius, FromFahrenheit or FromKelvin. The struct, given the temperature in one of these scales, will convert a value to other units. You can then obtain the temperature in selected units by reading the appropriate properties: Celsius, Fahrenheit or Kelvin.

The fifth member of the SensorReadings class, TimeStamp, contains the time point when the sensor readings were recorded. This can be useful when you stream data to the cloud and then perform a time series analysis with dedicated services like Azure Stream Analytics or Time Series Insights.

In addition, the SensorReadings class overrides the ToString method to properly display sensor values in the console:

```
public override string ToString()
{
  return $"Temperature: {Temperature.Celsius,5:F2} °C"
    + $" Temperature2: {Temperature2.Celsius,5:F2} °C"
    + $" Humidity: {Humidity,4:F1} %"
    + $" Pressure: {Pressure,6:F1} hPa";
}
```

The next step is to implement the class SenseHatService to access selected Sense HAT components. For testing purposes, I also decided to implement another class, SenseHatEmulationService, which serves as the emulator. I wanted to quickly test the Web API and other elements of the code without the need to connect the hardware. To easily switch between two concrete implementations, I used the dependency injection software design pattern. To this end, I first declared an interface:

```
public interface ISenseHatService
{
  public SensorReadings SensorReadings { get; }

  public void Fill(Color color);

  public bool EmulationMode { get; }
}
```

This interface defines the common members for the two concrete implementations:

- SensorReadings: This property enables the caller to get values obtained from the sensors—real values in the case of SenseHatService—and randomly generated values with SenseHatEmulationService.
- Fill: This method will be used to uniformly set the color of all LEDs.
- EmulationMode: This property indicates whether the Sense HAT is emulated (true) or not (false).

Next, I implemented the SenseHatService class. Let's look at the class constructor first, as shown in **Figure 3**. This constructor instantiates three private read-only fields: ledMatrix, pressureAndTemperatureSensor and temperatureAndHumiditySensor. To do this, I used appropriate classes from the IoT.Device.Bindings package: SenseHatLedMatrixI2c, SenseHatPressureAndTemperature and SenseHatTemperatureAndHumidity, respectively. Each class has a public constructor, which accepts one parameter, i2cDevice of an abstract type System.Device.I2c.I2cDevice. The default value of this parameter is null. In such a case, IoT.Device.Bindings will internally use an instance of the System.Device.I2c.Drivers.UnixI2cDevice or System.Device.I2c.Drivers.Windows10I2cDevice, depending on the OS.

Afterward, you can obtain sensor values (in the companion code, refer to SenseHat.DotNetCore.Common/Services/SenseHatService.cs):

```
public SensorReadings SensorReadings => GetReadings();

private SensorReadings GetReadings()
{
  return new SensorReadings
  {
    Temperature = temperatureAndHumiditySensor.Temperature,
    Humidity = temperatureAndHumiditySensor.Humidity,
    Temperature2 = pressureAndTemperatureSensor.Temperature,
    Pressure = pressureAndTemperatureSensor.Pressure
  };
}
```

and set the color of LED array:

```
public void Fill(Color color) => ledMatrix.Fill(color);
```

Before trying that out, though, let's first implement the emulator, SenseHatEmulationService. The full code for this class is available in the companion code (SenseHat.DotNetCore.Common/Services/SenseHatEmulationService.cs). The class derives from the ISenseHatService interface, so it has to implement the three public members described earlier: SensorReadings, Fill and EmulationMode.

I started by synthesizing sensor readings. For this purpose, I implemented the following helper method:

```
private double GetRandomValue(SensorReadingRange sensorReadingRange)
{
  var randomValueRescaled = randomNumberGenerator.NextDouble()
    * sensorReadingRange.ValueRange();

  return sensorReadingRange.Min + randomValueRescaled;
}

private readonly Random randomNumberGenerator = new Random();
```

GetRandomValue uses the System.Random class to generate the double, whose value falls in the specified range. This range is represented by an instance of the SensorReadingRange (SenseHat.DotNetCore.Common/Sensors/SensorReadingRange.cs), which has two public properties, Min and Max, that specify the minimum and maximum values for the emulated sensor reading. Additionally, the SensorReadingRange class implements one instance method, ValueRange, that returns the difference between Max and Min.

GetRandomValue is employed in the GetSensorReadings private method to synthesize sensor readings as shown in **Figure 4**.

Figure 3 **Selected Members of the SenseHatService Class**

```
public class SenseHatService : ISenseHatService
{
  // Other members of the SenseHatService (refer to the companion code)

  private readonly SenseHatLedMatrix ledMatrix;
  private readonly SenseHatPressureAndTemperature pressureAndTemperatureSensor;
  private readonly SenseHatTemperatureAndHumidity temperatureAndHumiditySensor;

  public SenseHatService()
  {
    ledMatrix = new SenseHatLedMatrixI2c();

    pressureAndTemperatureSensor = new SenseHatPressureAndTemperature();

    temperatureAndHumiditySensor = new SenseHatTemperatureAndHumidity();
  }

}
```

Manipulating Documents?

APIs to view, convert, annotate, compare, sign, assemble and search documents in your applications.

Try GroupDocs APIs for FREE

Download a Free Trial at

https://downloads.groupdocs.com

GROUPDOCS

GroupDocs.Viewer

View over 50 documents and image formats in any application using document viewer APIs.

GroupDocs.Annotation

Add annotations to specific words, phrases and any region of the document.

GroupDocs.Conversion

Fast batch document conversion APIs for any .NET, Java or Cloud app.

GroupDocs.Comparison

Compare two documents and get a difference summary report.

GroupDocs.Signature

Digitally sign Microsoft Word, Excel, PowerPoint and PDF documents.

GroupDocs.Assembly

Document automation APIs to create reports from templates and various data sources.

GroupDocs.Metadata

Organize documents with metadata within any cross platform application.

GroupDocs.Search

Transform your document search process for advance full text search capability.

- GroupDocs.Text
- GroupDocs.Editor
- GroupDocs.Parser
- GroupDocs.Watermark

Americas: +1 903 306 1676 EMEA: +44 141 628 8900 Oceania: +61 2 8006 6987
sales@asposeptyltd.com

Finally, I implemented the public members:

```
public SensorReadings SensorReadings => GetSensorReadings();
public void Fill(Color color) {/* Intentionally do nothing*/}
public bool EmulationMode => true;
```

I also created the SenseHatServiceHelper class, which I'll use later (SenseHat.DotNetCore.Common/Helpers/SenseHatServiceHelper.cs). SenseHatServiceHelper class has one public method, which returns an instance of either the SenseHatService or SenseHatEmulationService, depending on the Boolean parameter, emulationMode:

```
public static ISenseHatService GetService(bool emulationMode = false)
{
  if (emulationMode)
  {
    return new SenseHatEmulationService();
  }
  else
  {
    return new SenseHatService();
  }
}
```

The Console App

I can now test the code using my console app. This app can be executed on your development PC or the IoT device. When running on the PC, the app can use the emulator. To switch between emulation and non-emulation modes, I'll use a command-line argument, which will be a string with either a Y or N letter. The console app will parse this argument and then use either SenseHatEmulationService (Y) or SenseHatService (N). In emulation mode, the app will only display synthesized sensor readings. In non-emulation mode, the app will display values obtained from real sensors, and will also sequentially change the LED array color.

In non-emulation mode, the app will display values obtained from real sensors, and will also sequentially change the LED array color.

To create the console app, I supplemented the SenseHat.DotNetCore solution with a new project, SenseHat.DotNetCore.ConsoleApp, which I created using the Console App (.NET Core) project template. Then I referenced SenseHat.DotNetCore.Common and started implementing the Program class. I defined three private members:

```
private static readonly Color[] ledColors =
  { Color.Red, Color.Blue, Color.Green };

private static readonly int msDelayTime = 1000;

private static int ledColorIndex = 0;
```

The first member, ledColors, is a collection of colors that will be used sequentially to uniformly change the LED array. The second member, msDelayTime, specifies a time duration between accessing consecutive sensor readings and changing the LED array. The last member, ledColorIndex, stores the value of the currently displayed color from ledColors collection.

Figure 4 **SenseHatEmulationService**

```
private SensorReadings GetSensorReadings()
{
  return new SensorReadings
  {
    Temperature =
      Temperature.FromCelsius(GetRandomValue(temperatureRange)),
    Humidity = (float)GetRandomValue(humidityRange),
    Temperature2 =
      Temperature.FromCelsius(GetRandomValue(temperatureRange)),
    Pressure = (float)GetRandomValue(pressureRange)
  };
}

private readonly SensorReadingRange temperatureRange =
  new SensorReadingRange { Min = 20, Max = 40 };
private readonly SensorReadingRange humidityRange =
  new SensorReadingRange { Min = 0, Max = 100 };
private readonly SensorReadingRange pressureRange =
  new SensorReadingRange { Min = 1000, Max = 1050 };
```

Next, I wrote two helper methods:

```
private static bool ParseInputArgumentsToSetEmulationMode(string[] args)
{
  return args.Length == 1 && string.Equals(
    args[0], "Y", StringComparison.OrdinalIgnoreCase);
}
```

and:

```
private static void ChangeFillColor(ISenseHatService senseHatService)
{
  if (!senseHatService.EmulationMode)
  {
    senseHatService.Fill(ledColors[ledColorIndex]);

    ledColorIndex = ++ledColorIndex % ledColors.Length;
  }
}
```

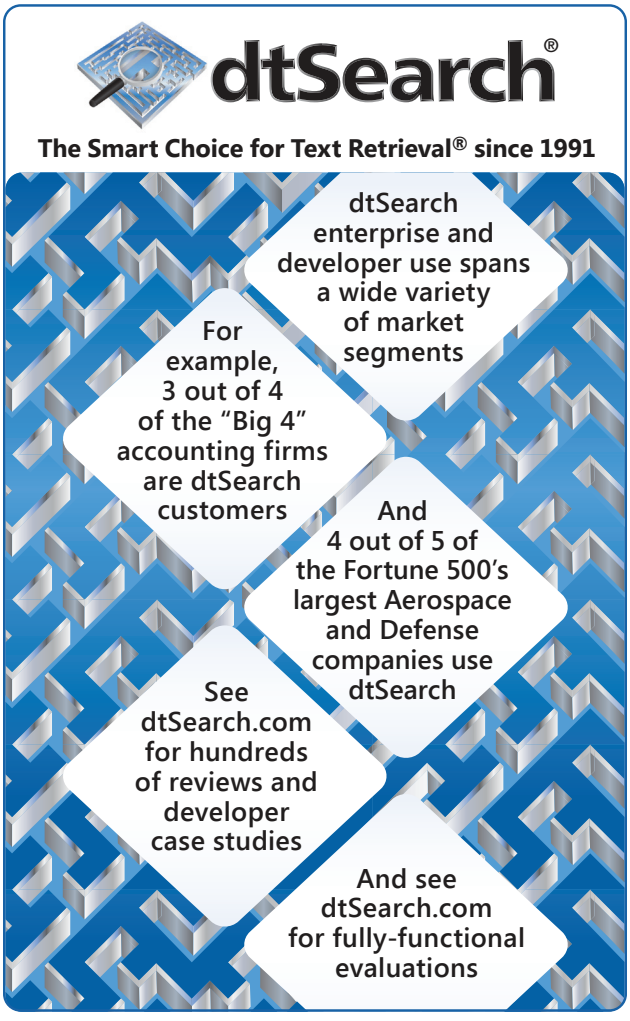
The first method, ParseInputArgumentsToSetEmulationMode, analyzes the collection of input arguments (passed to the Main method) to determine whether emulation mode should be used. The method will return true only if the collection has one element equal to y or Y.

The second method, ChangeFillColor, is effective only when emulation mode is off. If it is, the method will take a color from the ledColors collection and pass it to the Fill method of the concrete implementation of the SenseHatService; ledColorIndex is then incremented. In this specific implementation, the if statement in ChangeFillColor can be omitted because the Fill method of the SenseHatEmulationService does nothing. However, in general, the if statement should be included.

Finally, I implemented the Main method, shown in **Figure 5**, which wires everything together. First, the input arguments are parsed and, based on the result, I invoke the GetService static method of the SenseHatServiceHelper. Second, I display the string to inform the user whether or not the app works in emulation mode. Third, I start the infinite loop, in which sensor readings are obtained and, eventually, the LED array color is changed. The loop uses the msDelayTime to pause the app execution.

Deploying an App to a Device

To run the app in the RPi, you can download the .NET Core 3.0 SDK to your device, copy your code there, build the app, and finally execute it using the *dotnet run* .NET Core CLI command. Alternatively, you can publish the app using your development PC and then copy the binaries to the device. Here, I'll go with the second option.



Instantly Search Terabytes of Text

Executive Summary

- The dtSearch enterprise and developer product line instantly searches terabytes of text, with no limit on the number of concurrent search threads.
- dtSearch's own document filters support a wide variety of data formats, including "Office" files, PDFs (through PDF 2.0), emails and attachments, online data and other databases.
- The products offer over 25 hit-highlighted search options, with special forensics search features and international language support.
- For developers, the dtSearch Engine offers faceted searching and other advanced data classification options.
- SDKs cover Windows, Linux and macOS, and can also run on cloud platforms like Azure and AWS.
- Has cross-platform APIs for C++, Java and .NET with .NET Standard / .NET Core.

Key Benefits

Terabyte Indexer. dtSearch enterprise and developer products can index over a terabyte of text in a single index, spanning multiple directories, emails and attachments, online data and other databases. dtSearch products can create and search any number of indexes, and can search indexes during updates.

Concurrent, Multithreaded Searching. dtSearch developer products support efficient multithreaded searching, with no limit on the number of concurrent search threads.

Document Filters and Other Supported Data Types. dtSearch's own document filters cover "Office" documents, PDFs (through PDF 2.0), compression formats, emails and multilevel nested attachments, databases and online data. A DataSource API covers databases like SharePoint, NoSQL and SQL, along with BLOB data.

Over 25 Search Options. The dtSearch product line has over 25 search features, including federated searching with integrated relevancy-ranking options and multicolor highlighted hits across online and offline data.

Forensics; International Languages. dtSearch products offer special forensics search options like the ability to identify credit card numbers and email addresses, and the ability to generate and search for hash values. For international languages, dtSearch products support Unicode, including right-to-left languages, and Chinese/Japanese/Korean character handling options.

Faceted Search and Other Data Classification Options. The dtSearch Engine supports user-interface-driven faceted or "drill down" category searching, as well as numerous other full-text and metadata classification options.

SDKs. Cross-platform APIs for C++, Java and .NET with .NET Standard / .NET Core span Linux, macOS and Windows. SDKs also work on on cloud platforms like Azure and AWS. Document filters are built into the product line and also available for separate licensing.

For hundreds of developer case studies and press review, and fully-functional evaluations (including the search engine and the document filters), visit → **dtSearch.com**

Figure 5 **Entry Point of the Console App**

```
static void Main(string[] args)
{
  // Parse input arguments, and set emulation mode accordingly
  var emulationMode = ParseInputArgumentsToSetEmulationMode(args);

  // Instantiate service
  var senseHatService = SenseHatServiceHelper.GetService(emulationMode);

  // Display the mode
  Console.WriteLine($"Emulation mode: {senseHatService.EmulationMode}");

  // Infinite loop
  while (true)
  {
    // Display sensor readings
    Console.WriteLine(senseHatService.SensorReadings);

    // Change the LED array color
    ChangeFillColor(senseHatService);

    // Delay
    Task.Delay(msDelayTime).Wait();
  }
}
```

You first build the SenseHat.DotNetCore solution and then, in the solution folder, invoke the following command:

```
dotnet publish -r win-arm
```

The parameter -r win-arm can be omitted if project file contains the following property: <RuntimeIdentifier>win-arm</RuntimeIdentifier>. You can use the command-line interface of your choice or the Package Manager Console from Visual Studio. If you're using Visual Studio 2019, you can also publish the app using the UI tools. To do so, go to the Solution Explorer, right-click the ConsoleApp project, and choose Publish from the context menu. Visual Studio will display a dialog in which you choose Folder as a publish target. Then, under the publish profile settings, set Deployment Mode to Self-contained and Target Runtime to win-arm.

No matter which method you choose, the .NET Core SDK will prepare the binaries for deployment. By default, you'll find them in the bin\$(Configuration)\netcoreapp3.0\win-arm\publish output folder. Copy this folder to your device. The most straightforward way to copy these files is to use Windows File Explorer (bit.ly/2WYtnrT). Open File Explorer and type into the address bar the IP address of your device preceded by a double backslash and followed by c$. My RPi has an IP of 192.168.0.109, so I typed \\192.168.0.109\c$. After a while, File Explorer will display a prompt, asking you for the Administrator's password. Be patient, this can take several seconds. Finally, copy the binaries to your device.

To actually run the app, you can use PowerShell. The easiest way is to use the IoT Dashboard, shown in **Figure 6**. Simply right-click your device under the My devices tab and then choose PowerShell. You'll need to type the Administrator's password again when prompted. Then go to the folder with your binaries and execute the app by invoking the following:

```
.\SenseHat.DotNetCore.ConsoleApp N
```

> To actually run the app, you can use PowerShell. The easiest way is to use the IoT Dashboard.

You'll see the actual sensor readings as shown in **Figure 7**, and the SenseHat LED array will change color. You can see that the temperature reported by the two sensors is almost the same. The humidity and pressure are also within the expected range. To further confirm that everything works correctly, let's induce some changes. To do so, cover the device with your hand. As you'll see, the humidity goes up. In my case, the humidity increased from about 37% to 51%.

Web API

With .NET Core I can go even further and expose the sensor readings through the Web API service. I'll create a simple UI using Swagger UI (bit.ly/2IEnXXV). This UI will let the end user send HTTP requests to the IoT device as he would send them to a regular Web app! Let's see how this works.

I started by extending the SenseHat.DotNetCore solution with another ASP.NET Core Web Application project, SenseHat.DotNetCore.WebApp, creating the project with the API template. Then I referenced the SenseHat.DotNetCore.Common project and installed the Swashbuckle NuGet package (*Install-Package Swashbuckle.AspNetCore*). Detailed instructions for setting up Swagger

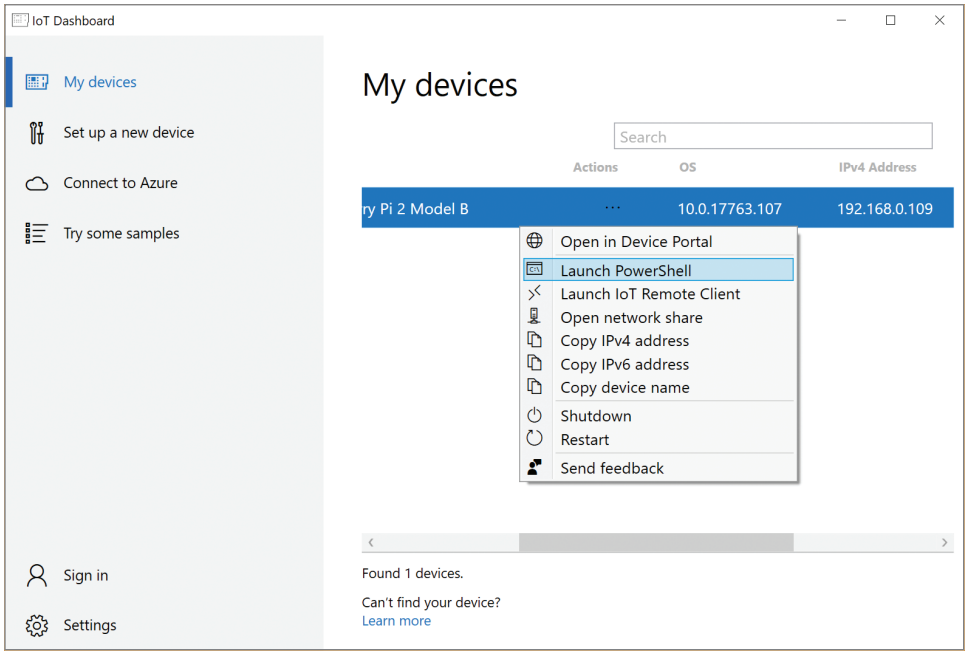


Figure 6 **Launching PowerShell Using the Windows 10 IoT Dashboard**

msdn magazine VP

MSDN MAGAZINE VENDOR PROFILE

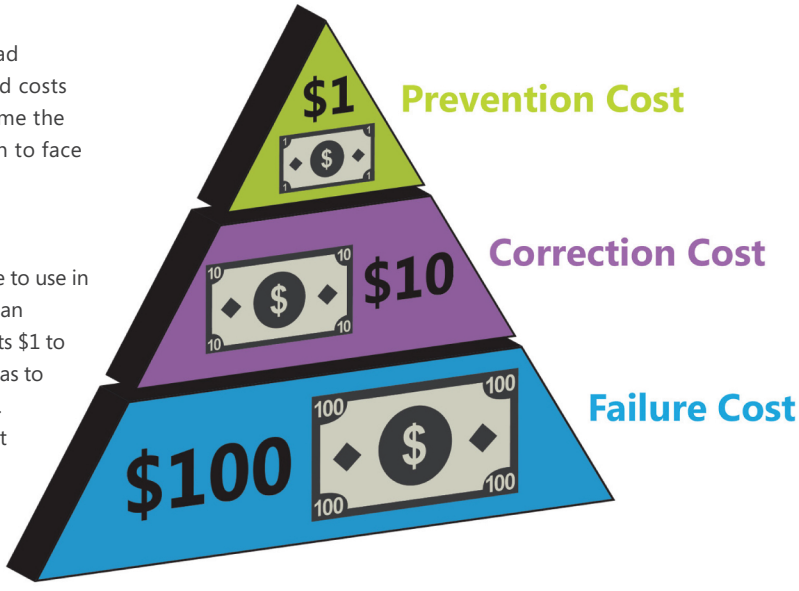
melissa

There's a Big Elephant in Your Business—Name it and Tame it!

Q&A with Bud Walker, Vice President of Enterprise Sales & Strategy

Q What's the elephant companies avoid talking about?
A Bad addresses. Everyone's experienced the trouble of bad addresses—poor business decisions, wasted time, increased costs and dissatisfied customers. That's why it's important to name the big beast in your business, so you can come up with a plan to face your address issues head on.

Q How do bad addresses affect the bottom line?
A This brings to mind, the 1-10-100 rule as it's a great example to use in this case. The 1-10-100 rule states that failing to fix an error at an earlier point will escalate the costs over time. So, let's say it costs $1 to verify a customer record at point-of-entry. Later, if the record has to be corrected after it enters the data warehouse, it will cost $10. And, if the record is never verified, that's $100 per record in lost opportunity costs. So the longer bad data lingers within the system, the more expensive it becomes. And this leads to increases in undeliverable mail, fraud, late deliveries/shipments—all of which ultimately affects the bottom line.



Q How can companies address their elephant?
A Tame it—by adopting a proactive rather than reactive approach to data quality. Why? Because mistakes happen. Customers or sales reps might enter contact info incorrectly while filling out forms, resulting in typos, and so on. Fixing addresses at the point-of-entry ensures that only clean data enters a system. With less time spent filling out forms, this improves the customer experience and call center/personnel efficiency.

Q What tools does Melissa offer that can help businesses embrace a proactive approach?
A Our Global Address Verification solution can verify and standardize addresses at point-of-entry and in batch for more than 240 countries and territories. Addresses are verified up to the delivery point, meaning addresses are verified to ensure that mail can be sent to that address. And if addresses are outdated, our tools can help update the customer's address to maintain and improve communication. Plus, it'll reduce waste associated with undeliverable mail, because the address is updated before mailing.

Q What else can businesses do once the addresses are good?
A Once addresses are clean, demographics, firmographics, geographic and property data can be added to gain customer insights. We can even fill missing pieces of contact data, like phone numbers and emails—completing the picture of who the customer is. The result is trusty data for improved analytics, efficient operations and stronger customer relationships.

Q How can Melissa help?
A Melissa has more than three decades of experience helping companies by providing a full-spectrum of data quality solutions across the entire data lifecycle. We offer super flexible options to fit your needs—from on-premise solutions to Cloud APIs, to components for SSIS, Excel and Dynamics CRM. Try our APIs in our Developer Portal. It's a great way to test for RAD, and get 1,000 credits a month with easy, on-demand refills.

For more information, please visit → **i.melissa.com/bigelephant**

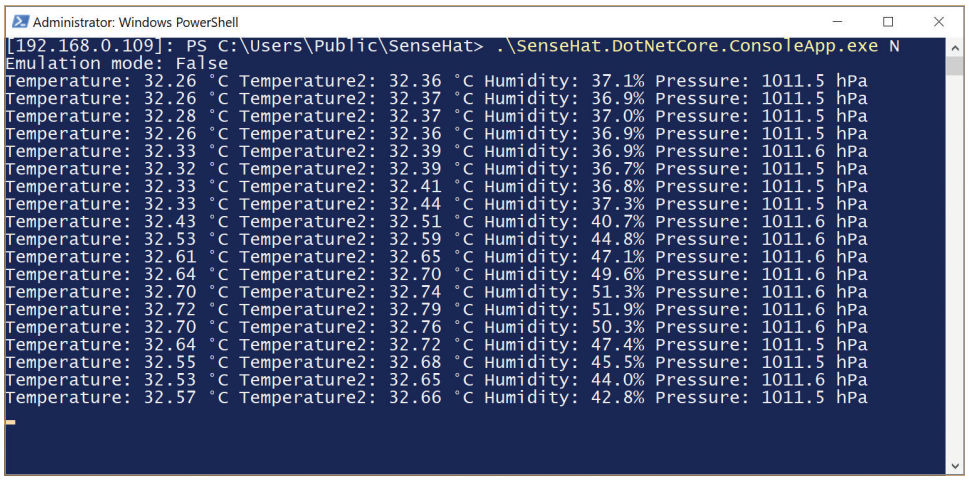


Figure 7 **Obtaining Sensor Readings Using the Console App Executed on the Raspberry Pi 2**

in ASP.NET Core Web Application are given at bit.ly/2BpFzWC, so I'll omit all the details and only show the instructions necessary to set up the Swagger UI in my app. All these changes will be made in the Startup.cs file of the SenseHat.DotNetCore.WebApp project.

First, I implemented the read-only field openApiInfo:

```
private readonly OpenApiInfo openApiInfo = new OpenApiInfo
{
  Title = "Sense HAT API",
  Version = "v1"
};
```

Then I modified the ConfigureServices method by adding the following statements:

```
services.AddSwaggerGen(o =>
{
  o.SwaggerDoc(openApiInfo.Version, openApiInfo);
});
```

These statements are responsible for setting up the Swagger generator. Next, I registered the concrete implementation of the ISenseHatService interface for dependency injection:

```
var emulationMode = string.Equals(Configuration["Emulation"], "Y",
  StringComparison.OrdinalIgnoreCase);

var senseHatService = SenseHatServiceHelper.GetService(emulationMode);
services.AddSingleton(senseHatService);
```

Here, the concrete implementation of the SenseHat service is set depending on the Emulation setting from the configuration file. The Emulation key is set to N in appsettings.Development.json, and to Y in appsettings.json. Consequently, the Web app will use the emulator in the development environment and the true Sense HAT hardware in the production environment. As in any other ASP.NET Core Web app, the production environment is enabled by default for the Release build configuration.

The last modification is to extend the Configure method with the instructions for setting up the Swagger endpoint:

```
app.UseSwagger();
app.UseSwaggerUI(o =>
{
  o.SwaggerEndpoint($"/swagger/
    {openApiInfo.Version}/swagger.
    json",
      openApiInfo.Title);
});
```

Then, I implemented the actual class for the Web API controller. In the Controllers folder I created the new file, SenseHatController.cs, which I modified as shown in **Figure 8**. SenseHatController has a public constructor, which is used for dependency injection to obtain an instance of the ISenseHatService. A reference to this instance is stored within the senseHatService field.

SenseHatController has two public methods, Get and SetColor. The first handles HTTP GET requests, and returns sensor readings from the Sense HAT add-on board. The second method, SetColor, handles HTTP POST requests. SetColor has one string argument, colorName. The client apps use this argument to choose the color, which is then utilized to uniformly change the LED array color.

I can now test the final version of the app. Again, I can do it using the emulator or true hardware. Let's start with the development PC and run the app with the Debug build configuration. After executing the app, type localhost:5000/swagger in the browser address bar. Note that if you use the companion code and execute the app using the Kestrel Web server, the browser will open automatically, and you'll be redirected to the swagger endpoint. I configured this using launchUrl of the launchSettings.json.

Figure 8 **A Full Definition of the SenseHatController Web API Service**

```
[Route("api/[controller]")]
[ApiController]
public class SenseHatController : ControllerBase
{
  private readonly ISenseHatService senseHatService;

  public SenseHatController(ISenseHatService senseHatService)
  {
    this.senseHatService = senseHatService;
  }

  [HttpGet]
  [ProducesResponseType(typeof(SensorReadings), (int)HttpStatusCode.OK)]
  public ActionResult<SensorReadings> Get()
  {
    return senseHatService.SensorReadings;
  }

  [HttpPost]
  [ProducesResponseType((int)HttpStatusCode.Accepted)]
  [ProducesResponseType((int)HttpStatusCode.BadRequest)]
  [ProducesResponseType((int)HttpStatusCode.InternalServerError)]
  public ActionResult SetColor(string colorName)
  {
    var color = Color.FromName(colorName);

    if (color.IsKnownColor)
    {
      senseHatService.Fill(color);

      return Accepted();
    }
    else
    {
      return BadRequest();
    }
  }
}
```

In the Swagger UI, you'll see a page with the Sense HAT API header. Right below this header are two rows with GET and POST labels. If you click one of them, a more detailed view appears. This view allows you to send requests to the SenseHatController, see responses and investigate the API documentation (shown previously in the left side of **Figure 1**). To show the sensor readings, expand the GET row and then click the Try It Out and Execute buttons. The request is sent to the Web API controller and handled, and sensor readings will appear under the Response body (shown previously in the right side of **Figure 1**). You send POST requests in a similar way—you just need to set the colorName parameter before clicking the Execute button.

To test the app on my device, I published the app using the Release configuration and then deployed the resulting binaries to Raspberry Pi (as with the Console App). Then I had to open port 5000, which I did by invoking the following command from PowerShell on the RPi2:

```
netsh advfirewall firewall add rule name="ASP.NET Core Web Server port"
  dir=in action=allow protocol=TCP localport=5000
```

Finally, I executed the app from PowerShell using the command:

```
.\SenseHat.DotNetCore.WebApp.exe --urls http://*:5000
```

The additional command-line argument (urls) is used to change the default Web server endpoint from localhost to a local IP address (bit.ly/2VEUlji) so the Web server can be accessed from other devices within the network. Once this was done, I opened the browser on my development PC, typed 192.168.0.109:5000/swagger, and the Swagger UI appeared (of course, you'll need to use the IP of your device). I then started sending HTTP requests to get sensor readings and change the LED array color, as shown previously in **Figure 1** and **Figure 2**.

Wrapping Up

In this article I showed how to implement a cross-platform IoT app with .NET Core 3.0. The app runs on the Raspberry Pi 2/3 and interacts with components of the Sense HAT add-on board. Using the app, I obtained various sensor readings (temperature, humidity and atmospheric pressure) through the Iot.Device.Bindings library. Then I implemented the ASP.NET Core Web API service and created a simple UI with Swagger. Now anyone can access those sensor readings and control the device remotely with just a few mouse clicks. The code can run, without any changes on the other system, including Raspbian. This example shows how you—a .NET developer—can utilize your existing skills and codebase to program various IoT devices. ■

Dawid Borycki *is a software engineer and biomedical researcher with extensive experience in Microsoft technologies. He has completed a broad range of challenging projects involving the development of software for device prototypes (mostly medical equipment), embedded device interfacing, and desktop and mobile programming. Borycki is an author of two Microsoft Press books: "Programming for Mixed Reality (2018)" and "Programming for the Internet of Things (2017)."*

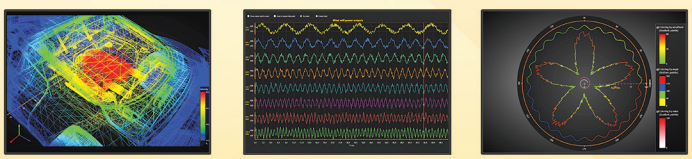
Thanks to the following Microsoft technical expert for reviewing this article: *Krzysztof Wicher*

LightningChart®

GPU-accelerated Data Visualization controls for the most demanding developers

LightningChart® .NET

The fastest 2D & 3D charts add-on for Visual Studio – **Windows Forms and WPF**



LightningChart® JS

The highest-performance **JavaScript charting library** focusing on real-time data visualization

Arction

Learn more about LightningChart at www.arction.com/LC

Visual Studio LIVE!
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

CHICAGO

Navigate Today's Tech in the Windy City

OCTOBER 6-10, 2019
SWISSOTEL, CHICAGO, IL

#VSLive

INTENSE DEVELOPER TRAINING CONFERENCE

In-depth Technical Content On:

- AI, Data and Machine Learning
- Cloud, Containers and Microservices
- Delivery and Deployment
- Developing New Experiences
- DevOps in the Spotlight
- Full Stack Web Development
- .NET Core and More

New This Year!

On-Demand Session Recordings Now Available

Get on-demand access for one full year to all keynotes and sessions from Visual Studio Live! Chicago, including everything Tuesday – Thursday at the conference.

SAVE $400! When You Register by August 2

Your Adventure Starts Here!

EVENT PARTNER Microsoft

GOLD SPONSOR CSC corptax.com

SUPPORTED BY Visual Studio | msdn magazine | Visual Studio MAGAZINE

PRODUCED BY CONVERGE 360 An 1105 MEDIA company

vslive.com/chicago

AGENDA AT-A-GLANCE

DevOps in the Spotlight | Cloud, Containers and Microservices | AI, Data and Machine Learning | Developing New Experiences | Delivery and Deployment | .NET Core and More | Full Stack Web Development

START TIME	END TIME				
START TIME	**END TIME**	**Pre-Conference Full Day Hands-On Labs: Sunday, October 6, 2019** *(Separate entry fee required)*			
7:30 AM	9:00 AM	Pre-Conference Workshop Registration - Coffee and Morning Pastries			
9:00 AM	6:00 PM	**HOL01** Full Day Hands-On Lab: Develop an ASP.NET Core 2 App with EF Core 2 in a Day - *Philip Japikse*		**HOL02** Full Day Hands-On Lab: Azure and Xamarin: Build the Mobile Apps of Tomorrow with Serverless, AI and Cross-platform clients - *Laurent Bugnion & Brandon Minnick*	
START TIME	**END TIME**	**Post-Conference Workshops: Monday, October 7, 2019** *(Separate entry fee required)*			
7:30 AM	8:00 AM	Post-Conference Workshop Registration - Coffee and Morning Pastries			
8:00 AM	5:00 PM	**M01** Workshop: Go Serverless in the Azure Cloud with Azure DevOps - *Brian Randell*	**M02** Workshop: SQL Server for Developers: The Grand Expedition - *Andrew Brust and Leonard Lobel*	**M03** Workshop: Cross-Platform C# Using .NET Core, Kubernetes, and WebAssembly - *Rockford Lhotka & Jason Bock*	
6:45 PM	9:00 PM	Dine-A-Round			
START TIME	**END TIME**	**Day 1: Tuesday, October 8, 2019**			
7:00 AM	8:00 AM	Registration - Coffee and Morning Pastries			
8:00 AM	9:15 AM	**T01** Guiding Principles for ASP.NET Core - *K. Scott Allen*	**T02** Cross-Platform Development with Xamarin, C#, and CSLA .NET - *Rockford Lhotka*	**T03** AI and Analytics with Apache Spark on Azure Databricks - *Andrew Brust*	**T04** Signing Your Code the Easy Way - *Oren Novotny*
9:30 AM	10:45 AM	**T05** Angular 101 - *Tracy Lee*	**T06** Azure 101 - *Laurent Bugnion*	**T07** DevOps Practices Can Make You A Better Developer - *Robert Green*	**T08** A Tour of Visual Studio 2019 - *Jason Bock*
11:00 AM	12:00 PM	Keynote: To Be Announced			
12:00 PM	1:00 PM	Lunch			
1:00 PM	1:30 PM	Dessert Break - Visit Exhibitors			
1:30 PM	2:45 PM	**T09** Reactive Programming Using RXJS - RXJS Operators - Real World Use Cases, Anti Patterns & Debugging - *Tracy Lee*	**T10** To Be Announced	**T11** Power BI: What Have You Done For Me Lately? - *Andrew Brust*	**T12** What's New in .NET Core 3.0 - *Jason Bock*
3:00 PM	4:15 PM	**T13** Advanced Azure App Services - *K. Scott Allen*	**T14** (WPF + WinForms) * .NET Core = Modern Desktop - *Oren Novotny*	**T15** Get Started with Git - *Robert Green*	**T16** Modernize Your App to be Delivered as a SaaS Service - *Nick Pinheiro*
4:15 PM	5:30 PM	Welcome Reception			
START TIME	**END TIME**	**Day 2: Wednesday, October 9, 2019**			
7:30 AM	8:00 AM	Registration - Coffee and Morning Pastries			
8:00 AM	9:15 AM	**W01** Designing and Building Advanced Angular Components - *Anthony Monsees*	**W02** Azure Cosmos DB Part I - Introduction to Cosmos DB - *Leonard Lobel*	**W03** Reach Any User on Any Platform with Azure AD B2C - *Nick Pinheiro*	**W04** Azure Pipelines - *Brian Randell*
9:30 AM	10:45 AM	**W05** Advanced Typescript - Dive into Inheritance, Union Types, Generics, and More - *Anthony Monsees*	**W06** Azure Cosmos DB Part II - Building Cosmos DB Applications - *Leonard Lobel*	**W07** 9 Azure Services Every Developer Should Know About - *Eric D. Boyd*	**W08** Non-Useless Unit Testing Entity Framework & ASP.NET MVC - *Benjamin Day*
11:00 AM	12:00 PM	General Session: To Be Announced			
12:00 PM	1:00 PM	Birds-of-a-Feather Lunch			
1:00 PM	1:30 PM	Dessert Break - Visit Exhibitors - Exhibitor Raffle @ 1:15pm (Must be present to win)			
1:30 PM	1:50 PM	**W09** Fast Focus: Getting Started with ASP.NET Core 2.0 Razor Pages - *Walt Ritscher*	**W10** Fast Focus: What's New in EF Core 2.x - *Jim Wooley*	**W11** Fast Focus: Serverless of Azure 101 - *Eric D. Boyd*	
2:00 PM	2:20 PM	**W12** Fast Focus: What is WSL and Why Do I Care? - *Brian Randell*	**W13** Fast Focus: A Real-time SignalR Core Chat - *Jim Wooley*	**W14** Fast Focus: Scrum in 20 Minutes - *Benjamin Day*	
2:30 PM	3:45 PM	**W15** Diving Deep Into ASP.NET Core 2.x - *Philip Japikse*	**W16** Azure Data Explorer— An In-depth Look at the New Microsoft PaaS offering - *Raj Krishnan*	**W17** Keep Secrets with Azure Key Vault - *Eric D. Boyd*	**W18** Getting to SAFe in the Enterprise - *Jim Szubryt*
4:00 PM	5:15 PM	**W19** JavaScript for the C# (and Java) Developer - *Philip Japikse*	**W20** Building End-to-End ML Solutions Using Azure Machine Learning Service - *Raj Krishnan*	**W21** Achieving DevSecOps with Azure DevOps and Containers - *Jim Szubryt*	**W22** Scrum Under a Waterfall - *Benjamin Day*
7:00 PM	9:00 PM	VSLive!'s Trolley Tour of the Windy City			
START TIME	**END TIME**	**Day 3: Thursday, October 10, 2019**			
7:30 AM	8:00 AM	Registration - Coffee and Morning Pastries			
8:00 AM	9:15 AM	**TH01** WebAssembly: the Browser is Your OS - *Jeremy Likness*	**TH02** UX Beyond the Keyboard: Designing Conversational Interfaces - *John Alexander*	**TH03** Building a Stronger Team, One Strength at a Time - *Angela Dugan*	**TH04** C# Language Enhancements, Roslyn and You - *Jim Wooley*
9:30 AM	10:45 AM	**TH05** What's New in Bootstrap 4 - *Paul Sheriff*	**TH06** How to Have Better Business Intelligence Through Visualizations - *Walt Ritscher*	**TH07** How Do You Measure Up? Collect the Right Metrics for the Right Reasons - *Angela Dugan*	**TH08** Get Your BASH on with WSL - *Brian Randell*
11:00 AM	12:15 PM	**TH09** Create Your Own SPA Using jQuery - *Paul Sheriff*	**TH10** Microsoft Power Platform, RAD Done Right: Building Dynamic Mobile Apps with PowerApps - *Walt Ritscher*	**TH11** Writing Testable Code and Resolving Dependencies - DI Kills Two Birds with One Stone - *Miguel Castro*	**TH12** Entity Framework Performance Monitoring and Tuning - *Jim Wooley*
12:15 PM	1:30 PM	Lunch			
1:30 PM	2:45 PM	**TH13** Real-time Web Applications with ASP.NET Core SignalR - *Roland Guijt*	**TH14** Building UWP Apps for Multiple Devices - *Tony Champion*	**TH15** Demystifying Microservice Architecture - *Miguel Castro*	**TH16** .NET Standard: Reuse All the Code - *Jonathan "J." Tower*
3:00 PM	4:15 PM	**TH17** Top 10 Browser Threats Mitigated - *Roland Guijt*	**TH18** Building Cross Device Experiences with Project Rome - *Tony Champion*	**TH19** Hacking Time: Using Micro-habits to Customize Your Environment - *John Alexander*	**TH20** Dotnet CLI or: How I Learned to Stop Worrying and Love the Command Line - *Jonathan "J." Tower*

Speakers and sessions subject to change

Secure Your Supply Chain with the Azure IoT and Blockchain Cloud

Stefano Tempesta

The adoption of blockchain and IoT technologies has the potential to help the manufacturing industry in making supply chains more secure and the associated processes more transparent. Chances are, you've heard of blockchain only in the context of cryptocurrency, so how is blockchain relevant to supply chains?

A Distribution Supply Chain Example

The traditional supply chain management structure has limitations in the reliability and accuracy of measurements, and in the transparency of processes among all parties involved. IoT and blockchain technologies represent a potential solution to address these challenges by introducing devices to automate the collection of metrics at any stage of the supply chain, as well as a distributed digital ledger for storing transaction logs in an immutable way. In addition, blockchain smart contracts can use rule-based intelligence to perform validation on this data and update the state of each supply chain stage for all parties involved, in a completely trustworthy and transparent manner. For example, terms, conditions and any other logic can be coded into a smart contract for verifying the correct execution of a transaction of goods between two stages of the supply chain.

This article discusses:

- A sample distribution supply chain
- Using IoT Central to connect, monitor and manage devices
- Managing supply chain status with Microsoft Flow
- Creating and deploying smart contracts with Azure Blockchain Workbench

Technologies discussed:

Azure IoT Central, Microsoft Flow, Azure Blockchain Development Kit, Azure Blockchain Workbench, Ethereum, Blockchain Smart Contracts

Code download available at:

bit.ly/2LnRASp

Figure 1 simplifies a distribution supply chain in which a food producer transfers goods to a food processing company for packaging and shipping. The produce is sealed at a set temperature and humidity level, and these or similar conditions must be maintained during transportation to the warehouse and retail store.

These conditions are coded in a blockchain smart contract, as illustrated later in this article. IoT devices with sensors for measuring temperature and humidity are installed in the containers and carriers storing and shipping goods. Telemetry data is captured at regular intervals during the storage and distribution phases, and the actual metrics validated against the set levels in the smart contract. If the values for temperature or humidity exceed such levels, the state of the transport of the goods changes to "out of compliance" and this state is reflected in the transaction log stored inside the blockchain digital ledger. All participants in the supply chain can view the state and details of the contract at any point in time. The

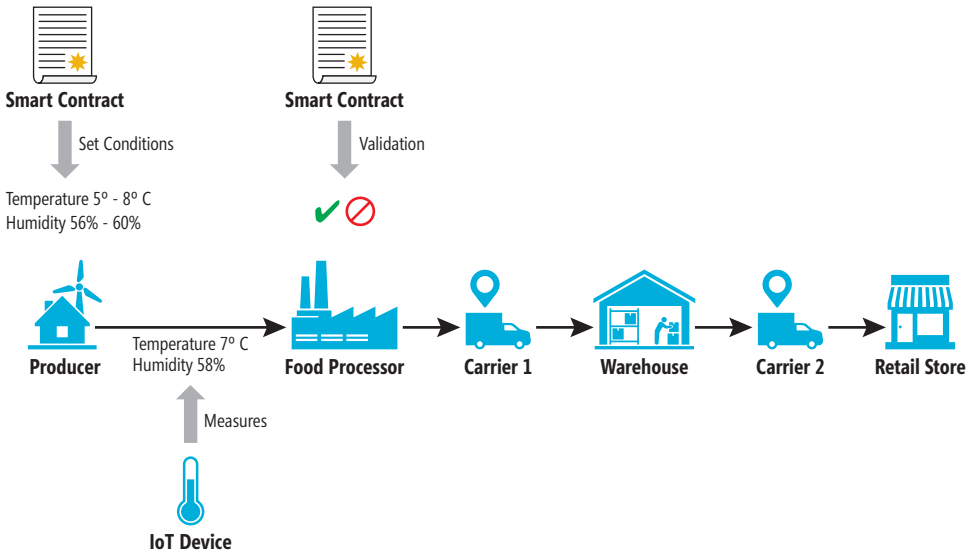


Figure 1 **A Distribution Supply Chain Using a Smart Contract**

counterparty responsible for storage or transportation of goods will specify the next counterparty handling the parcels as the next stage, and the registered IoT devices will forward telemetry data to a central IoT hub. This allows the initiator of the supply chain and all its participants to identify which counterparty didn't fulfill the compliance regulations if at any point in the process either the temperature or humidity requirements weren't met.

A supply chain requires a state transition diagram to articulate the possible flows, and the various transition functions at each state. Each party in the supply chain is only allowed to take certain actions depending on their role and the state of the products being transported. States, workflows and involved parties are described in a smart contract and deployed in Azure Blockchain Workbench, as described later in the article. The example provided here implements the four states shown in **Figure 2**.

Azure IoT Central

The example in this article uses Azure IoT Central (bit.ly/2X6JIwe) to connect the IoT devices. IoT Central is a fully managed Software-as-a-Service (SaaS) solution that makes it easy to connect, monitor and manage devices at scale. The initial setup of devices is simplified by adopting device templates for a variety of platforms, and the project implementation of the IoT component of the supply chain solution can benefit from device simulation for testing purposes.

Figure 2 **The Four States of the Example Supply Chain**

State	Description
Created	Indicates that the contract has initiated, and tracking is in progress.
InTransit	Indicates that a counterparty currently is in possession and responsible for goods being transported.
Completed	Indicates the product has reached its intended destination.
OutOfCompliance	Indicates that the agreed upon terms for temperature and humidity conditions weren't met.

The IoT solution implemented for the distribution supply chain in the current example consists of a cloud-based gateway that receives telemetry data from device sensors and the devices themselves that connect to IoT Central. Before being allowed to stream any data, IoT devices are registered with the cloud platform and are assigned an identity. When connecting to IoT Central, a device sends its identity information, along with sensor data. If the device fails to send this identity information, the connectivity will fail. Once a successful connection is established between the devices and the cloud gateway, data is ingested at set intervals. Azure IoT Central facilitates this flow by providing device registration and data ingestion capabilities, as well as visualization of telemetry points on a dashboard, as shown in **Figure 3**.

Figure 3 shows measurements produced by a simulated device. In Azure IoT Central you can define a device template for the devices that connect to your application, a sort of blueprint that defines the characteristics and behavior of a type of device. A template can be used for connecting a physical device or for simulating the streaming of data within certain boundaries. Simulated data includes:

- Device properties that are set by a device and are read-only in the application.
- The status, which determines the behavior of the device.
- Measurements specific to the installed sensors.

A supply chain requires a state transition diagram to articulate the possible flows, and the various transition functions at each state.

The setup of an application in IoT Central starts by defining at least one device template and then adding a device, either real or simulated. You can access the IoT Central dashboard at bit.ly/2RBtWD6, and then navigate to the Azure IoT Central Application Manager page to start creating a new Azure IoT Central application. Once you create an application, access the Device Explorer section to add a real or simulated device associated with each device template in the application. When you create a device template, Azure IoT Central generates a simulated device from the template. The

Figure 3 **The Azure IoT Central Dashboard**

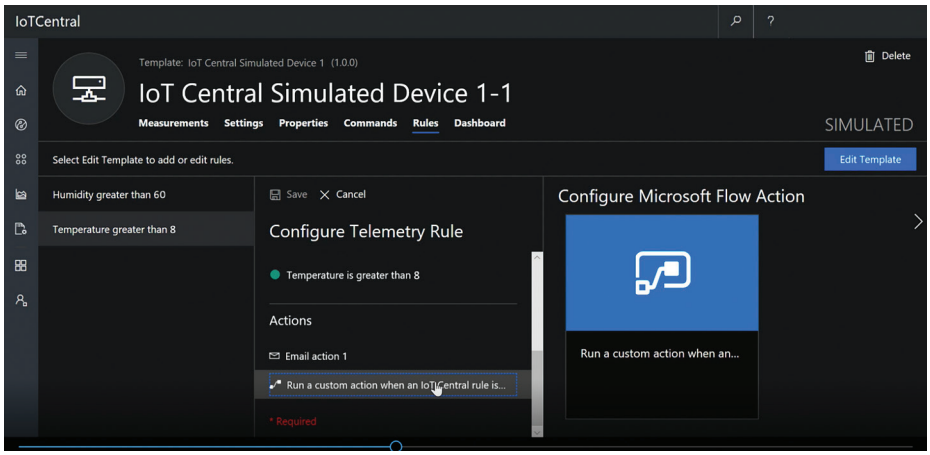


Figure 4 **Configuring a Microsoft Flow Action**

simulated device generates telemetry that enables you to test the behavior of your application before you connect a real device. From the Measurements tab, you can specify the measurements, such as telemetry, event, and state, sent by the device, and define the rules associated with the device.

After setting up a device and its measurements, you can use events to define point-in-time data the device should send when an event such as an error or a component failure occurs. Azure IoT Central can simulate device events to enable you to test the behavior of your application before you connect a real device. After a short while, the Measurements tab will show a chart of the events randomly generated from your simulated connected device.

Finally, you create a rule that runs a custom action when the measured temperature or humidity exceeds the specified conditions. In this case, I configured a Microsoft Flow action, as shown in **Figure 4**, to run when the humidity level is greater than 60 and the temperature greater than 8.

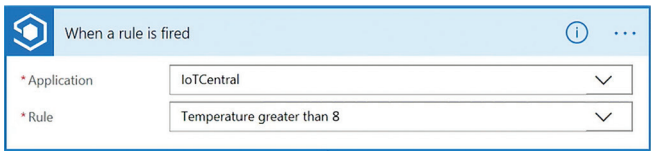


Figure 5 **How a Flow Is Triggered in IoT Central**

Microsoft Flow

Why Flow? Blockchain networks are isolated from the outside world, meaning that no data produced "off-chain" can be brought in. Smart contracts operate on data stored within the blockchain itself. The Azure Blockchain Development Kit (bit.ly/2TkG23Z) is an extension to the capabilities of Azure Blockchain Workbench (more on that in a moment). The Blockchain Dev Kit incorporates Azure services for key management, off-chain identity and data, monitoring, and messaging APIs into a reference architecture that can be used to rapidly build blockchain-based applications that integrate with any external system. With Flow, it's possible to send a message to the Azure Service Bus deployed as part of Azure Blockchain Workbench and transfer data, securely, to a blockchain digital ledger. **Figure 5** shows the trigger of the flow when a rule is fired in IoT Central.

A message is then prepared and eventually placed onto the Azure Blockchain Workbench Service Bus, in accordance to the REST API-based integration pattern described at bit.ly/31WqmIs. In preparing the message to send to Service Bus, the following parameters are created as part of the flow itself:

- requestId: This is a unique value to identify a request; in Flow, this can be implemented by a variable with the expression guid.
- timestamp: This is the date and time of the request; in Flow, this can be obtained by extracting the current time instant's ticks. However, as there are 10 million ticks in a second, this would result in a really big number, so I typically calculate the current Unix epoch time, which is the number of seconds that have elapsed since Jan. 1, 1970 (midnight UTC).

> Blockchain networks are isolated from the outside world, meaning that no data produced "off-chain" can be brought in.

In Flow, a timestamp can be easily calculated with the subtraction expression:

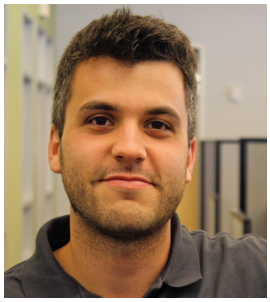
```
sub(variables('TicksNow'), variables('TicksUnixEpoch'))
```

Here, TicksNow is an integer variable defined as ticks(utcNow()), and TicksUnixEpoch is another variable defined as ticks('1970-01-01').

msdn magazine VP

MSDN MAGAZINE VENDOR PROFILE

LEADTOOLS®



LEADTOOLS Simplifies Xamarin Development

A Q&A with Hadi Chami, Developer Support Manager

Describe LEADTOOLS and what you have to offer Xamarin developers?

LEADTOOLS is a family of comprehensive toolkits designed to help programmers integrate document, medical, multimedia and raster imaging technologies into their desktop, server, tablet, and mobile applications.

Our Xamarin SDK includes .NET Standard development libraries to create Android, iOS, macOS, and UWP applications that leverage the Xamarin app model. The Xamarin interface makes it easy for developers to add state-of-the art technology including OCR, Barcode, PDF, DICOM, Annotations, 150+ File Formats, Compression, Image Processing, and more.

I saw Xamarin highlighted in the most recent LEADTOOLS release, what's the latest update?

We are always looking for new ways to streamline the development process for programmers. In our latest Version 20 update, we introduced two new components: the Xamarin Image Viewer and the Xamarin Camera Control.

Just like our .NET Image Viewer, the new LEADTOOLS Xamarin Image Viewer supports more than 150 different file formats as well as multi-page functionality. This image viewer also has touch-screen optimized interactive modes, navigation tools, and full support of LEADTOOLS Annotations.

The LEADTOOLS Xamarin Camera Control is for developers looking to build applications that utilize the device's camera without having to go through the headache of using both native Android and iOS code. Using the control requires no knowledge of the iOS and Android native camera APIs because LEADTOOLS takes care of that for you.

From my experience, I have only used native code when implementing a camera control into my applications. How easy is it to add the Xamarin Camera Control to my application?

The control can be added to any Xamarin.Forms application with just one line of XAML code that gives developers full access to the device's cameras. This means that the user can toggle the focus control, camera torch, and camera flash. Developers can also tap into the live camera feed to add custom image processing or other LEADTOOOLS features, including OCR, barcode recognition, and image processing filters.

What future additions can we expect for the LEADTOOLS Xamarin SDK?

Later this year, we will be adding a Document Viewer to our Xamarin SDK. Programmers will be able to add document viewing, composing, and converting of documents to Xamarin applications. It will be perfect for developers wanting to add document related functionality to their cross-platform applications.

For more information, visit → www.leadtools.com

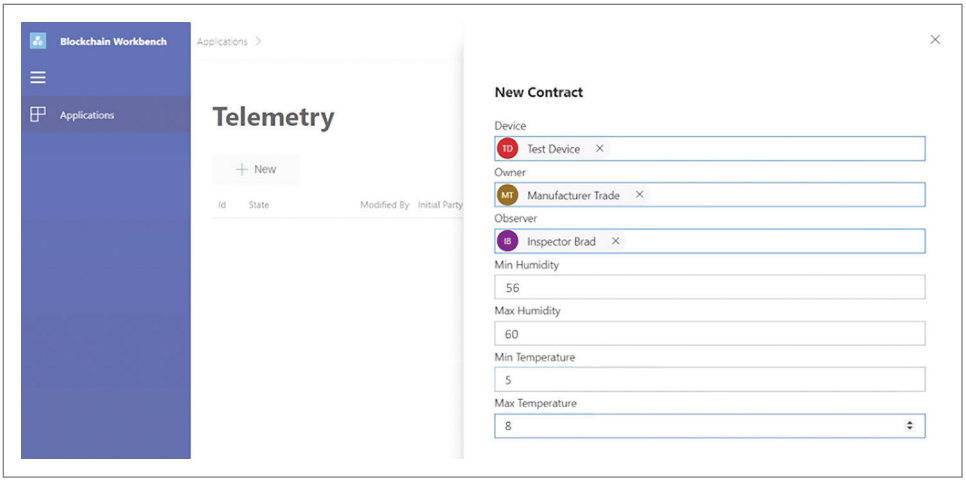


Figure 6 **The First Stage of Execution of the Telemetry Smart Contract**

The Service Bus message also requires:

- userChainIdentifier: The blockchain address of the user or device that was created on the blockchain network.
- contractLedgerIdentifier: The blockchain address of the contract on the ledger.
- workflowFunctionName: The name of the smart contract's function to invoke.
- Parameters: An array of objects identifying telemetry data.

These transactions are trackable and irreversible.

The JSON format of this message is as follows:

```
{
  "requestId": "",
  "userChainIdentifier": "",
  "contractLedgerIdentifier": "",
  "workflowFunctionName": "ReadTelemetry",
  "Parameters": [
    { "name": "humidity", "value": "" },
    { "name": "temperature", "value": "" }
    { "name": "timestamp", "value": "" }
  ]
}
```

Azure Blockchain Workbench

The initial objective of blockchain technology was to facilitate trusted fiscal transactions between two parties without the need for a third party, such as a bank, to be involved. Because blockchain relies on a decentralized, networked system, it's harder to corrupt. If the information on one of the blocks is edited, all others viewing that block can see the edits that have been made. The introduction of blockchain technology into supply chains can help prevent issues such as fraud and counterfeiting. Transactions on the blockchain are stored and distributed across multiple nodes within the network. These transactions, or records, are very secure due to how the blockchain works (see my article at bit.ly/2ZQcm16 for more information about cybersecurity with blockchain), which makes this technology the perfect method for transparently and securely recording transactions in an independent and immutable log. In the context of a supply chain, in addition to a transaction log, blockchain smart contracts verify and enforce an agreement between two or more parties involved in a transaction. Smart contracts allow for credible transactions to happen without authorization and verification by a central authority. Also, these transactions are trackable and irreversible.

The blockchain ledger in this solution is hosted in Azure; specifically, it's an Ethereum installation deployed as part of Azure Blockchain Workbench. With Azure Blockchain Workbench (bit.ly/2XBhWqw), you can configure and deploy a consortium network in just a few minutes, thanks to automatic ledger deployment, network construction and pre-built blockchain commands that greatly reduce infrastructure deployment time. Once the necessary infrastructure is provisioned, you can code your smart contracts and then deploy them in Blockchain Workbench for execution. Blockchain Workbench builds a Web UI automatically, with no coding, after your smart contract's workflow, and allows you to keep track of any status change. For example, the message sent by IoT Central using Flow sends the measured values for temperature and humidity. A smart contract validates these values, and if they don't meet the conditions coded in the smart contract itself, it will set the status of this transaction to Out of Compliance. This stage of the supply chain is then invalid, and the process won't (or shouldn't) progress to the next stage. **Figure 6** shows the first stage of execution of the Telemetry smart contract used in this supply chain solution: A new contract is created between two parties involved; a device is registered for capturing telemetry data; and the initial conditions for humidity and temperature are set.

Figure 7 **The Telemetry Smart Contract**

```
contract Telemetry
{
  // States
  enum StateType { Created, InTransit, Completed, OutOfCompliance }
  enum SensorType { None, Humidity, Temperature }

  // Parties
  address public Owner;
  address public InitiatingCounterparty;
  address public Counterparty;
  address public PreviousCounterparty;
  address public Device;
  address public SupplyChainOwner;
  address public SupplyChainObserver;

  // Properties
  StateType public State;
  SensorType public ComplianceSensorType;
  int public MinHumidity;
  int public MaxHumidity;
  int public MinTemperature;
  int public MaxTemperature;
  int public ComplianceSensorReading;
  bool public ComplianceStatus;
  string public ComplianceDetail;
  int public LastSensorUpdateTimestamp;
```

msdn magazine VP

MSDN MAGAZINE VENDOR PROFILE

DevExpress®

Build Your Best Without Limits or Compromise

Q&A with Julian Bucknall, Chief Technical Officer, DevExpress

Q DevExpress just released a major update to its product line. Can you tell us about your newest UI components and Visual Studio development tools?

A Thanks to great feedback from the DevExpress developer community, we've extended the scope and breadth of all DevExpress subscriptions. The list of new products and features is quite long, but it includes new UI components for Blazor, .NET Core 3 support, and a myriad of updates to existing control libraries (from our Office-inspired desktop controls to our adaptive web components).

Q DevExpress was among the first third-party tool vendors to announce support for Blazor. Can you detail some of the products you are shipping or expect to ship for Microsoft's new web development platform?

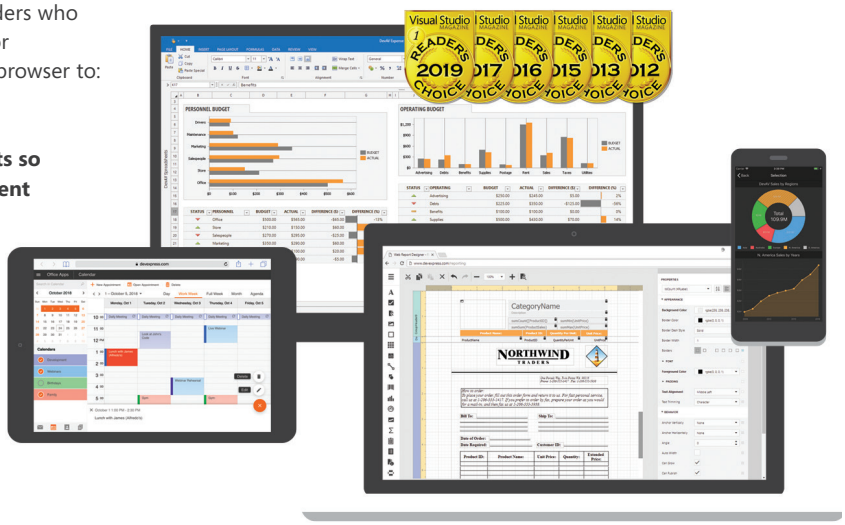
A As of July, 2019, we offer ten Blazor UI components, including a Data Grid, Pivot Grid, TreeView and a series of multi-purpose components (such as a ComboBox and Date Editor). We are fully committed to the Blazor platform and expect to release many more Blazor UI components by year's end. Readers who want to follow our progress or to download our Blazor community tech preview can do so by pointing their browser to: devexpress.com/blazor.

Q Last year, we asked you how DevExpress supports so many different products across multiple development platforms. What would you say are the platforms of highest importance to you and your customers in 2019?

A DevExpress ships and supports hundreds of UI components for both desktop and web development. Guided by internal poll results and usage statistics, we remain fully committed to the needs of those targeting WinForms, WPF, ASP.NET, .NET Core, Blazor, as well as the pure client-side web. Our long-term goal has not changed – we want to deliver best-of-breed development tools and help our customers create compelling user-experiences that amaze. For a complete list of new products and features shipping in our award-winning Universal Subscription, I'd ask MSDN readers to visit the following webpage: devexpress.com/2019.

Q DevExpress received 14 first place Visual Studio Readers' Choice Awards in 2019. What's the secret to your success?

A Truth is, the secret to our success is not much of a secret. What sets us apart from our competition is our incredibly loyal user base. Our customers help shape the direction of the product line and make certain that DevExpress remains focused on products/features that matter. We rarely chase rainbows because our customers keep us to the straight and narrow: they want us to dedicate resources to tools that add legitimate business value. As a result, we strive to help our customers build their best, and for 21 years, we've worked hard to deliver on this promise. I want to personally thank each and every DevExpress customer for his or her continued support. We would be nothing were it not for our users.



Download—Compare—Decide

Download your 30 day risk-free DevExpress trial today → devexpress.com/try

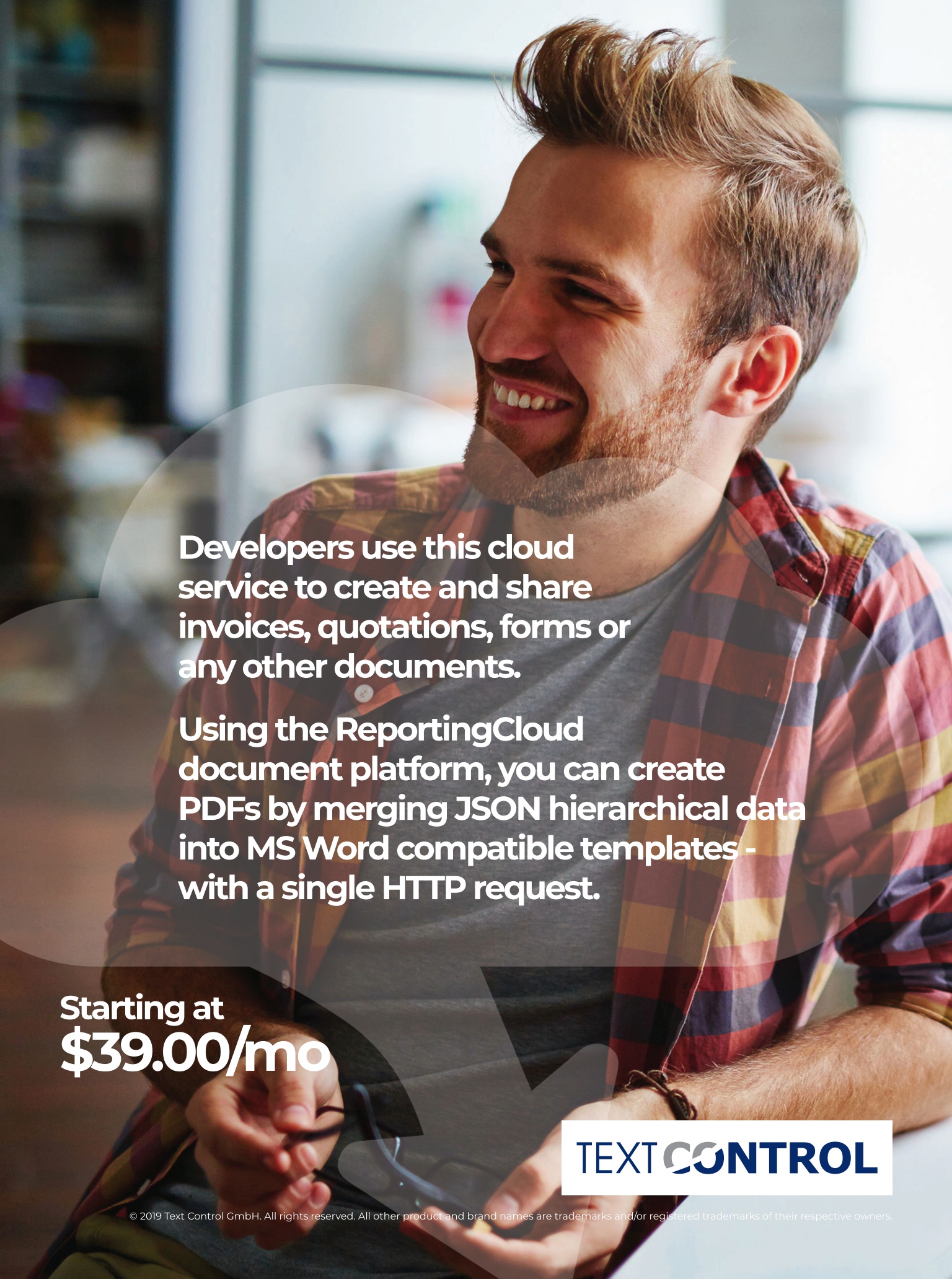
reportingcloud

Tired of “programming” PDFs?

Use this REST API to create pixel-perfect documents from MS Word templates in any application.

Free trial at www.reporting.cloud

A product from TEXT CONTROL. We love documents.



© 2019 Text Control GmbH. All rights reserved. All other product and brand names are trademarks and/or registered trademarks of their respective owners.

Figure 8 **The Telemetry Smart Contract Constructor**

```
constructor(address device, address supplyChainOwner, address supplyChainObserver,
  int minHumidity, int maxHumidity, int minTemperature, int maxTemperature) public
{
  ComplianceStatus = true;
  ComplianceSensorReading = -1;
  InitiatingCounterparty = msg.sender;
  Owner = InitiatingCounterparty;
  Counterparty = InitiatingCounterparty;
  Device = device;
  SupplyChainOwner = supplyChainOwner;
  SupplyChainObserver = supplyChainObserver;
  MinHumidity = minHumidity;
  MaxHumidity = maxHumidity;
  MinTemperature = minTemperature;
  MaxTemperature = maxTemperature;
  State = StateType.Created;
  ComplianceDetail = "N/A";
}
```

Figure 9 **The ReadTelemetry Function**

```
function ReadTelemetry(int humidity, int temperature, int timestamp) public
{
  if (Device != msg.sender)
  {
    revert();
  }

  if (State == StateType.Completed || State == StateType.OutOfCompliance)
  {
    revert();
  }

  LastSensorUpdateTimestamp = timestamp;

  if (humidity < MinHumidity || humidity > MaxHumidity)
  {
    ComplianceSensorType = SensorType.Humidity;
    ComplianceSensorReading = humidity;
    ComplianceDetail = "Humidity value out of range.";
    ComplianceStatus = false;
  }
  else if (temperature < MinTemperature || temperature > MaxTemperature)
  {
    ComplianceSensorType = SensorType.Temperature;
    ComplianceSensorReading = temperature;
    ComplianceDetail = "Temperature value out of range.";
    ComplianceStatus = false;
  }

  if (ComplianceStatus == false)
  {
    State = StateType.OutOfCompliance;
  }
}
```

The Telemetry smart contract in **Figure 7** implemented in Solidity programming language (solidity.readthedocs.io) for the Ethereum platform, defines parties, properties and states, which all together are a representation of the status of the contract at any given time. Supply chain parties are identified by a blockchain address; in Azure Blockchain Workbench, this is the user or device address as registered on the platform. To access Blockchain Workbench, parties have to be registered in Azure Active Directory. States represents an enumeration of stages in the supply chain process (contract created, goods in transit, transaction completed or out of compliance recorded), as well as the two different types of sensors in use in this example (humidity and temperature). Enumeration allows for more entries to be added easily, should the supply chain process require additional stages or introduce different sensor types.

The UI for entering a new contract in Blockchain Workbench is generated automatically based on the constructor of the Telemetry smart contract in **Figure 8**. This constructor sets the initial conditions for the transportation process that have to be met by all parties involved in the supply chain. It also sets the initial state to StateType.Created.

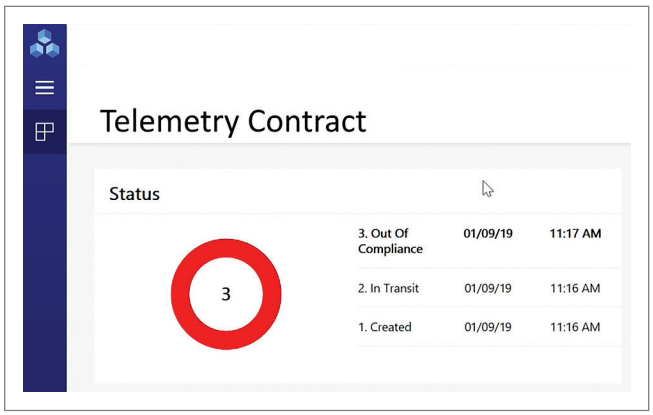


Figure 10 **The Updated Telemetry Contract**

Finally, when telemetry data is streamed and a rule exception occurs in IoT Central, the flow described earlier is executed, which in turn sends a message to Blockchain Workbench to invoke the ReadTelemetry function in **Figure 9**. This function validates the input values for humidity or temperature recorded by the IoT device, and raises an out-of-compliance state if the conditions set in the first stage aren't met.

From a UI perspective, Blockchain Workbench automatically records each state change and updates the overall status of the Telemetry contract accordingly, as shown in **Figure 10**.

Wrapping Up

The Azure platform offers a collection of services for building fully integrated solutions for the needs of Industry 4.0, including the IoT platform and managed blockchain services. In this article, I described a common blockchain pattern for IoT-enabled monitoring of goods as they move along a multi-party supply chain. Specific compliance rules must be met throughout the transportation process. In this scenario, an initiating party specifies contractual conditions, such as a required humidity and temperature range, to which a carrier on the supply chain must adhere. At any point, if the device takes a temperature or humidity measurement that's out of range, the smart contract state will be updated to indicate that it's out of compliance, recording a transaction on the blockchain and triggering remediating events downstream.

For those interested in learning more, a great starting point is the Azure reference architecture for supply chain, which you'll find at bit.ly/2X7NIr7. ■

Stefano Tempesta *is a Microsoft Regional Director, MVP on Azure, AI and Business Applications, and a member of Blockchain Council. A regular speaker at international IT conferences, including Microsoft Ignite and Tech Summit, Tempesta's interests extend to blockchain and AI-related technologies. He created Blogchain Space (blogchain.space), a blog about blockchain technologies, writes for* MSDN Magazine *and* MS Dynamics World*, and publishes machine learning experiments on the Azure AI Gallery (gallery.azure.ai).*

Thanks to the following Microsoft technical expert for reviewing this article: *Danilo Diaz*

msdn magazine VP

MSDN MAGAZINE VENDOR PROFILE

SQL for API Integration

Q&A with Eric Madariaga, Chief Marketing Officer, CData Software

Q What are some of the biggest challenges that you think developers are facing today, and how do you address them?

A Integration continues to be a pervasive challenge. With the continuing expansion of cloud and SaaS applications, data has grown increasingly fragmented. With such disparate systems, each tied to a unique API, integrating, managing, and maintaining integrations for all of a company's data creates a whole new set of challenges.

Our goal is to help developers solve their integration challenges. We offer standards-based drivers that enable straightforward SQL-92 connectivity to more than 150+ SaaS, NoSQL, and Big Data sources through established standards like ODBC, JDBC, ADO.NET, ODATA, and JSON.

Q What benefits do drivers offer software developers for API integration?

A Developers are often far more productive when they use SQL for API access. At the core, developers need to query, aggregate, join, and summarize data. Drivers provide a single uniform interface (SQL) that developers can use to retrieve and manipulate tabular data. Because tables are self-describing, developers spend more time connecting data and less time hunting through documentation.

Furthermore, drivers are widely supported across popular developer tools and components. In Visual Studio, for example, developers can explore data in real-time through the Server Explorer and leverage standard .NET data binding in the same way that they would with integrate with SQL Server. By standardizing how developers interact with APIs, our drivers reduce development costs, slash architectural complexity, and dramatically reduce ongoing maintenance.

Q There are many ways that developers can integrate with APIs—what makes the CData Drivers unique?

A Our Drivers offer a lot more than your typical application SDK or connector. For starters, a full SQL-92-ready query engine is built into each of our drivers. This allows developers to read data using well-known primitives like JOINs, GROUP BY, ORDER BY, and all filters (including ones with formulas). Often APIs don't offer all of those algorithms and developers have to post-process the data from the API. With SQL, all of that is clearly expressed in a query.

Our drivers support high-performance features like query folding/push-down, which translates SQL into optimized calls to the underlying application APIs, embedded caching, transparent paging of large datasets, and bulk operations. This translates into better performance and fewer number of API calls for any given task which is increasingly important as platform vendors now enforce limits.

Ultimately developers should be able to leverage their SQL knowledge to work with data, regardless of how or where it's stored.

For more information, visit → https://www.cdata.com/

Visual Studio LIVE! EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS | SQL Server LIVE! TRAINING FOR DBAs AND IT PROS | TECHMENTOR IN-DEPTH TRAINING FOR IT PROS | Office & SharePoint LIVE! ON-PREMISE, CLOUD, & CROSS-PLATFORM TRAINING | Artificial Intelligence LIVE! AI FOR DEVELOPERS AND DATA SCIENTISTS | Cloud & Containers LIVE! CLOUD-NATIVE, PAAS & SERVERLESS COMPUTING

The Ultimate Education Destination

LIVE! 360

TECH EVENTS WITH PERSPECTIVE

6 Great Events, 1 Low Price!

November 17-22, 2019 | ORLANDO
Royal Pacific Resort at Universal

AGENDA JUST ANNOUNCED!
live360events.com/orlando

Live! 360 brings the IT and developer community together for a unique conference, featuring 6 co-located conferences for (almost) every IT title. Customize your learning by choosing from hundreds of sessions, dozens of track topics, workshops and hands-on labs from hundreds of expert speakers.

EVENT PARTNER | PLATINUM SPONSOR | SILVER SPONSOR | SUPPORTED BY

Microsoft | accusoft | aws | msdn magazine | Redmond Channel Partner | Redmond Magazine | VIRTUALIZATION & Cloud Review | Visual Studio MAGAZINE

Visual Studio LIVE!

EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

Visual Studio Live! features unbiased and practical developer training on the Microsoft Platform. Explore hot topics such as ASP.NET Core, JavaScript, Xamarin, Database Analytics and more!

SQL Server LIVE!

TRAINING FOR DBAs AND IT PROS

Whether you are a DBA, developer, IT Pro or Analyst, SQL Server Live! will provide you with the skills you need to drive your data to succeed.

Office & SharePoint LIVE!

ON-PREMISE, CLOUD, & CROSS-PLATFORM TRAINING

Office & SharePoint Live! provides leading-edge training to administrators and developers who must customize, deploy and maintain SharePoint Server on-premises and in Office 365.

TECHMENTOR

IN-DEPTH TRAINING FOR IT PROS

TechMentor gives a strong emphasis on doing more with the tech you already own plus solid coverage of what is next - striking the perfect balance of training essentials for the everyday IT Pro.

Artificial Intelligence LIVE!

AI FOR DEVELOPERS AND DATA SCIENTISTS

Artificial Intelligence Live! offers real-world training on the languages, libraries, APIs, tools and cloud services you need to implement real AI and Machine Learning solutions today and into the future.

NEW! IN 2019 Cloud & Containers LIVE!

CLOUD-NATIVE, PAAS & SERVERLESS COMPUTING

Cloud & Containers Live! covers both IT infrastructure and software development aspects of cloud-native software design, development, release, deployment, operations, instrumentation/monitoring and maintenance.

Who Should Attend:

- Developers
- IT Professionals
- Database Administrators
- Business Intelligence Professionals
- SharePoint Professionals
- and more!

Save $500 when you register by August 16!

Promo Code MSDN

live360events.com/orlando

PRODUCED BY CONVERGE 360 by 1105 MEDIA

CONNECT WITH LIVE! 360

twitter.com/live360
@live360

facebook.com
Search "Live 360"

instagram.com
@live360_events

linkedin.com
Join the "Live! 360" group!

Introducing Azure Deployment Manager

David Tepper

How complex are your services? Do you have tens of resources split over a bunch of resource groups and regions? Hundreds? Thousands? Does your engineering system make it easy to declaratively define the topology of those resources, how they work together, and how they can be deployed and updated in a safe, scalable fashion?

Currently in preview, Azure Deployment Manager (ADM) provides a new set of features for Azure Resource Manager (bit.ly/2Ib6WHM) and greatly expands your deployment capabilities. If you need to deploy a complex service to several regions, or gain more control over when resources are deployed in relation to one another, or limit your customer's exposure to bad updates by catching them while in progress, then ADM is for you.

ADM allows you to perform staged rollouts of resources, meaning they're deployed region by region in an ordered fashion. As part of this rollout, the health of your services is monitored, and if unacceptable performance degradation is detected, deployment automatically stops, allowing you to troubleshoot and reduce the scale of the impact. These tools are used internally by hundreds of Microsoft services to ensure safe, reliable deployments, ensuring high availability and preventing or dramatically reducing service unavailability caused by regressions in updates.

This article discusses:

- Define your infrastructure as code to include your entire service hierarchy
- Deploy services in stages using safe deployment practices
- Integrate health checks into service deployment to ensure healthy updates

Technologies discussed:

Azure Resource Manager, Azure PowerShell, Azure Storage Explorer

Code download available at:

bit.ly/2F6CInD

Azure Deployment Manager Concepts

ADM introduces two new concepts to the Azure Resource Manager (ARM) deployment story: service topologies and rollouts.

As depicted in **Figure 1**, a service topology describes the hierarchy of the resources you deploy to Azure by organizing them into groups. Template and parameter files form the basis of any ARM deployment. These represent the Service Units, or individually deployable chunks of a given service, such as the front end of a service. A group of Service Units comprises a Service, and a Service Topology is nothing more than a group of Services.

ADM makes it easy to define these kinds of complex topologies, even when the Services span multiple regions, subscriptions, and resource groups. Once defined, they can be deployed and then updated any number of times through the use of Rollouts, which

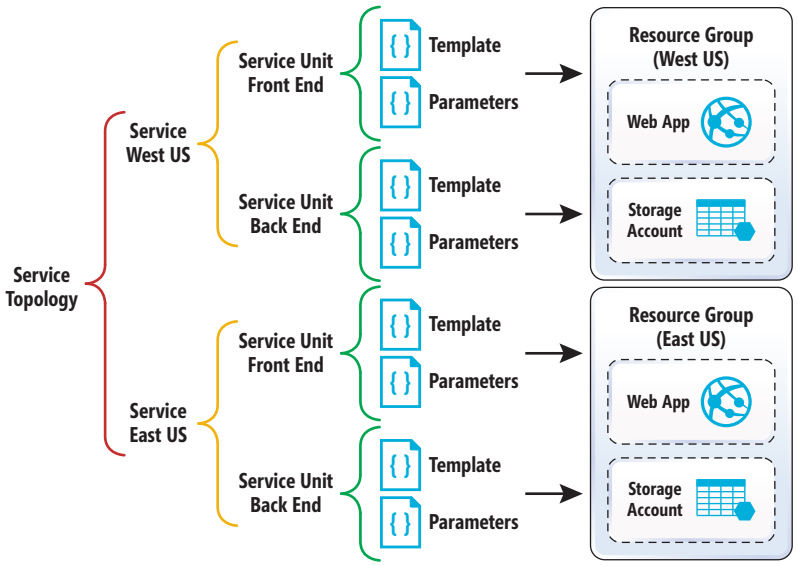


Figure 1 **Service Topology Structure**

defines the structure and timing for how to deploy the Service Units in a staged, safe fashion.

As depicted in **Figure 2**, a Rollout is composed of Step Groups. Step Groups define the order of deployment and support both sequential and parallel operations. For example, I can have Step Group 2 execute after Step Group 1 (as shown), or have Step Groups 1 and 2 deploy in parallel before deploying Step Group 3.

As the name implies, Step Groups are further broken down into Steps, which address steps to take before a deployment and steps to take after a deployment. Examples of such steps include:

- **Wait:** Pauses deployment for a specified amount of time before continuing, allowing for manual tests, as well as time for resources to bake.
- **Health checks:** ADM integrates with existing service monitoring providers to check the health of services as deployments are underway. If the health of a service degrades, deployment is halted.

More steps, including ones that allow for execution of custom code, are coming soon. To better understand these concepts, we'll work through an example.

Deploy Your First Rollout

Before we get started, it's a good idea to familiarize yourself with some of the resources and tools you'll be working with. Here are a few links to help get your environment set up and give you the background you need to work with ARM templates and follow through with the example:

- Become familiar with Azure Resource Manager templates (bit.ly/2Ib6WHM)
- Visual Studio Code and the Resource Manager extension (bit.ly/2MH29IF)
- Start up a PowerShell terminal in Azure Cloud Shell (bit.ly/2yESmTP)
- Download and install Microsoft Azure Storage Explorer (bit.ly/2IaaxWA)
- Download and unzip the example files (bit.ly/2F6ClnD)

Once you've downloaded and installed the example files, you can find the ArtifactStore folder, which contains a templates folder and a binaries folder.

The templates folder contains the ARM template artifacts. 1.0.0.0 and 1.0.0.1 represent the two versions of the binary artifacts. Within each version, there's a folder for each service (Service East US and Service West US). Each service has a pair of template and parameter files for creating a storage account, and another pair for creating a Web application. The Web application template calls a compressed package, which contains the Web application files. The compressed file is a binary artifact stored in the binaries folder.

The binaries folder contains the binary artifacts with 1.0.0.0 and 1.0.0.1, again representing the two versions of the binary artifacts. Within each version there's one .zip file for creating the Web application in the West US location, and another .zip file to create the Web application in the East US location.

Even though both the template artifacts and the binary artifacts have two versions, only the binary artifacts are different between the two versions. In practice, binary artifacts are updated more frequently than template artifacts. For a full description of what the contents of these artifact files are, check out the ADM tutorial at aka.ms/admtutorial.

ADM introduces two new concepts to the Azure Resource Manager (ARM) deployment story: service topologies and rollouts.

Template artifacts are used by the service topology template, and binary artifacts are used by the rollout template. Both the topology template and the rollout template define an artifact source Azure resource, which is a resource used to point Resource Manager to

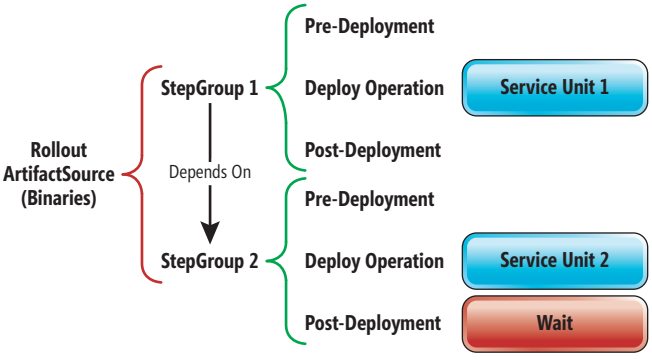


Figure 2 **Rollout Structure**

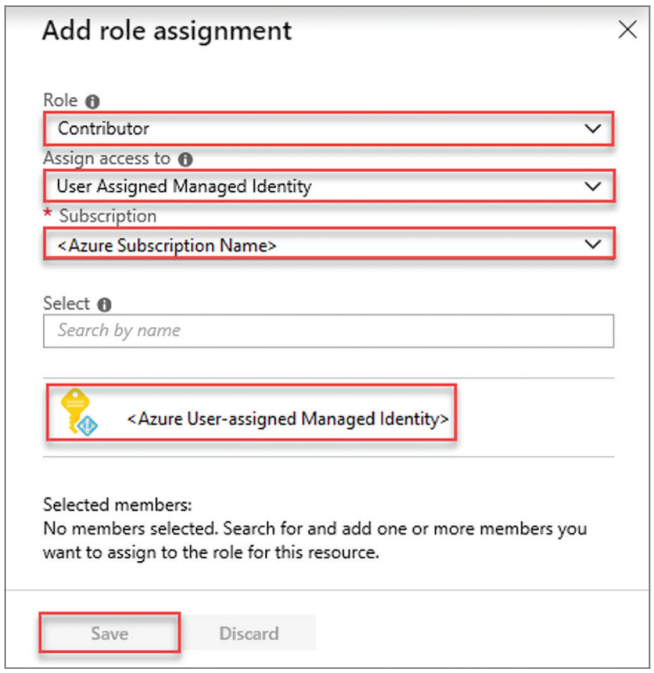


Figure 3 **Creating a User Assigned Managed Identity**

the template and binary artifacts that are used in the deployment. To simplify this example, one storage account is used to store both the template artifacts and the binary artifacts. Both artifact sources point to the same storage account, so let's get that set up.

First, create an Azure storage account. For this you can follow the instructions published in the "Quickstart: Upload, Download, and List Blobs Using the Azure Portal" article at bit.ly/2F6NMRt.

Next, create a blob container in the storage account, then copy the two folders (binaries and templates) and the content of the two folders to the blob container. Then get the SAS location of the container using the following steps. From Azure Storage Explorer, navigate to the blob container and right-click the blob container from the left pane, and then select Get Shared Access Signature. Configure Start time and Expiry time and then select Create.

Make a copy of the URL. This URL is the SAS URI needed to populate a field in the two parameter files—the topology parameters file and rollout parameters file—during a later step.

Later in this example, you'll deploy a rollout. A user-assigned managed identity is needed to perform the deployment actions. This identity must be granted access to the Azure subscription to which you're deploying the service, and must have sufficient permission to complete the artifact deployment. Let's create a user-assigned managed identity and configure the access control for your subscription.

Sign in to the Azure portal (portal.azure.com) and create a user-assigned managed identity. For more on this, see bit.ly/2F7lqqx. From the portal, select Subscriptions from the left menu, and then select your subscription. Select Access control (IAM), and then select Add role assignment. You should see the dialog box in **Figure 3**.

Let's enter values into the displayed fields. The Role dropdown provides permission to complete the artifact deployment (the Web applications and the storage accounts). For this example, select Contributor from the list. In practice, you want to restrict the permissions to the minimum.

Next, click the Assigned access dropdown and select User Assigned Managed Identity. Then select the user-assigned managed identity you created earlier in the example. Finally, select Save and be sure to write down the Name of your managed identity, as you'll use this in a later step.

Prepare the Templates

Now that we've set up the necessary Azure account resources, it's time to make some small edits to the provided files so they can correctly target your subscription and storage location. Start by opening \ADMTemplates\CreateADMServiceTopology.Parameters in Visual Studio Code or any text editor. **Figure 4** shows the parameters in the Service Topology. Let's walk through each parameter now.

namePrefix: This parameter is used to create the names for the Deployment Manager resources. For example, using the "jdoe" prefix, the service topology name is jdoeServiceTopology. The resource names are defined in the variables section of this template. For this one, use a string with four to five characters. This prefix is used to create unique Azure resource names.

azureResourceLocation: To simplify the example, all resources will share this location unless otherwise specified. Don't touch this parameter for now.

artifactSourceSASLocation: This field should be populated with the saved SAS URI to the blob container where your service unit template and parameters files are stored for deployment.

templateArtifactRoot: This parameter provides the offset path from the Blob container where the templates and parameters are stored. The default value is templates/1.0.0.0. Don't change this value unless you want to change the folder structure used when uploading the files to Azure Storage. Relative paths are used in this example, and the full path is constructed automatically by ADM. If, in practice, you want to use absolute paths, ADM supports that, too.

targetSubscriptionID: This is simply the subscription ID to which the Deployment Manager resources are going to be deployed and billed. Use your subscription ID in this example.

Now, open \ADMTemplates\CreateADMRollout.Parameters in Visual Studio Code or any text editor. Enter the parameter values as just described. It's important that these fields have the same values in both files. There's a new field in this file that must also be filled in called managedIdentityID. Enter the user-assigned

```
"parameters": {
  "namePrefix": {…
  },
  "azureResourceLocation": {…
  },
  "artifactSourceSASLocation": {…
  },
  "templateArtifactRoot": {…
  },
  "targetSubscriptionID": {…
  }
},
```

Figure 4 **Service Topology Parameters**

msdn magazine VP

MSDN MAGAZINE VENDOR PROFILE

Migrating from Desktop to Web with GrapeCity

Q&A with Chris Bannon, Global Product Manager of GrapeCity

Q Could you tell us a bit about GrapeCity?
A GrapeCity's family of products provides developers, designers, and architects with the ultimate collection of easy-to-use tools for building fast and feature-complete applications. Our diverse product line delivers data grids, spreadsheets, UI components, reporting and document solutions for JavaScript, mobile, and desktop including ComponentOne Studio, ActiveReports, Wijmo, Spread, and GrapeCity Documents API.

Q What separates GrapeCity from other developer tools in the market?
A Our products are built to empower developers. Our motto is: "get the toolkit and get the team." The GrapeCity Customer Engagement team is really attentive and is one of our strongest assets. We pride ourselves on the relationships we build with our customers. We solve problems so that developers can focus on creativity, innovation, and optimization.

Q What are GrapeCity's predictions for the developer market?
A We follow the development landscape closely and support all major platforms, including ActiveX, .NET (WinForms, WebForms, ASP.NET, WPF, UWP, and Xamarin) and JavaScript (Angular, React, and Vue). Currently, mobile and web are surpassing desktop use; this influences development as a whole. We are helping developers migrate their enterprise desktop apps into cutting edge web apps. Many companies today are moving from older desktop software to more modern web applications. Desktop developers need to learn how to build enterprise applications in the modern JS ecosystem and it can be overwhelming. We want to help guide them and give them a structured way of bringing their desktop apps to the web quickly and effectively.

We are also excited with the latest developments in Blazor technology from Microsoft which extends the reach of .NET code to the browser using web assembly.

Q How are you helping developers migrate their desktop apps to web?
A We are taking a multifaceted approach to address developer-specific migration needs. Our .NET offerings (ComponentOne Studio Enterprise and ComponentOne Ultimate) continue to grow and address the needs of .NET, mobile, and web developers.

In the last ComponentOne release, we included Wijmo (our suite of 60+ JavaScript UI controls) with all ComponentOne subscriptions. We offer these tools as a package, along with a mixture of resources written by developers to help lay out the best framework and tools to simplify the migration process. We are also producing a guide for development teams and software leads to find the best approach to migrating. We explain consideration issues when migrating from desktop to web, dev tools, and then walk you through our recommended path. Furthermore, we offer developer-authored technical papers to help any type of developer with their individual process.

Feel free to reach out to me personally at **cbannon@grapecity.com.** With GrapeCity, get the toolkit, get the team; we're here to help.

For more information, please visit → **www.grapecity.com/migration**

managed identity, replacing the <ManagedIdentity> portion with the saved Name of your managed identity.

First Deployment

It's time to deploy the two Web sites! Use the Azure CloudShell PowerShell terminal to execute the scripts that follow. Let's start by running a script to deploy the service topology, like so:

```
$resourceGroupName = "<Enter a Resource Group Name>"
$location = "Central US"
$filePath = "<Enter the File Path to the Downloaded Example Files>"
# Create a resource group
New-AzResourceGroup -Name $resourceGroupName -Location "$location"
# Create the service topology
New-AzResourceGroupDeployment `
  -ResourceGroupName $resourceGroupName `
  -TemplateFile "$filePath\ADMTemplates\CreateADMServiceTopology.json" `
  -TemplateParameterFile "$filePath\ADMTemplates\CreateADMServiceTopology.
Parameters.json"
```

Note that New-AzResourceGroupDeployment is an asynchronous call. The success message only means the deployment has successfully begun. In a moment I'll go over how to check the deployment status. For now, verify that the service topology and the underlying resources have been created successfully using the Azure portal, as depicted in **Figure 5**.

Now, let's deploy the rollout template, as follows:

```
# Create the rollout
New-AzResourceGroupDeployment `
  -ResourceGroupName $resourceGroupName `
  -TemplateFile "$filePath\ADMTemplates\CreateADMRollout.json" `
  -TemplateParameterFile "$filePath\ADMTemplates\CreateADMRollout.Parameters.json"
```

We can check the rollout progress using the following script:

```
# Get the rollout status
$rolloutname = "<Enter the Rollout Name>"
# "adm0925Rollout" is the rollout name used in this example
Get-AzDeploymentManagerRollout `
  -ResourceGroupName $resourceGroupName `
  -Name $rolloutName `
  -Verbose
```

After the rollout is deployed successfully, you'll see two more resource groups created, one for each service. Open the Azure portal and browse to the newly created Web applications under the new resource groups created by the rollout deployment. Open the Web application in a Web browser and the site's HTML will verify the location and the version.

To try out deploying a new version (1.0.0.1) of the Web application, open CreateADMRollout.Parameters.json, then update binaryArtifactRoot to binaries/1.0.0.1. Redeploy the rollout as instructed before. You don't need to redeploy the Service Topology. Verify the update by navigating to the Web site and noting the version number displayed by the page.

Congratulations, you just deployed two services into multiple regions using a staged rollout and updated them to a new version! Next, let's walk through integrating health checks into these steps so that if a service update is unhealthy, the deployment will automatically stop before impacting your most important regions.

Deploy a Health-Integrated Rollout

Health monitoring providers offer several mechanisms to monitor and alert you to service health issues. Azure Monitor (azure.microsoft.com/services/monitor) is an example of one such service, which fires alerts when certain thresholds are exceeded. Some thresholds are indicative of a problem with service health when exceeded. For example, if you're deploying a new update to your service and your memory and CPU utilization spike beyond expected levels, Azure Monitor and health monitoring providers like it will notify you of the issues so you can take corrective action.

> Health monitoring providers offer several mechanisms to monitor and alert you to service health issues.

These health providers typically offer REST APIs so the status of your service's monitors can be examined programmatically. The REST APIs will either come back with a simple healthy/unhealthy signal (usually determined by the HTTP response code) or with detailed information about the signals it receives.

The new healthCheck step in Azure Deployment Manager allows you to declare HTTP codes that indicate a healthy service, or, for more complex REST results, you can even specify regular expressions that indicate a healthy response if they match. To make this even easier to use, the Azure Deployment Manager team is working closely with the top health monitoring providers to pre-author these HTTP codes and regular expressions, so you can simply copy and paste this part of the healthCheck step if you're using one of these providers.

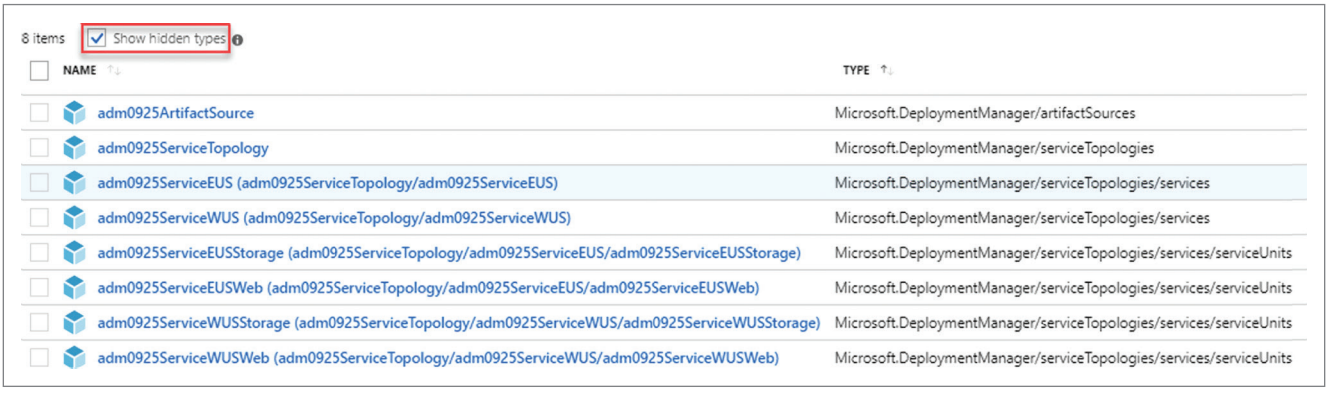


Figure 5 **Viewing Hidden Deployment Resources**

msdn magazine VP

MSDN MAGAZINE VENDOR PROFILE

ASPOSE

File APIs for Developers: DOC, PDF, XLS, EML, PPT and more

Q&A with Boris Pazin, Paid Support & Marketing Director at Aspose

Q Who is Aspose, and what do you do?
A Aspose makes APIs for working with business-centric files like DOC, PDF, XLS, EML, PPT and more. Developers trust Aspose APIs to Create, Edit, Export and Convert over 100 file formats in their applications.

Q What types of operations could be done with files?
A Developers use files in many ways, like importing and exporting data, reporting, user collaboration and data visualization, just to name a few. Aspose APIs allow the developer to create new files, edit or format existing files, combine and compare files, print, convert to other formats, etc.

Q What makes Aspose unique?
A Quality and comprehensiveness; no one else supports files like Aspose. Whether you look at the number of supported file formats, file features or platforms, Aspose is the leader. We offer native APIs for almost every file type and major platform.

Q Does Aspose use Microsoft automation, or have third-party dependencies?
A No. Aspose APIs are 100% standalone. They are a pure .NET alternative to the Microsoft Office Object Model.

Q Who uses Aspose?
A Our customers range from small shops to the largest enterprise companies. Aspose have over 300,000 registered developers in 114 countries. More than 75% of Fortune 100 companies trust and use our products.

Q Is it hard or costly to get started with Aspose?
A Not at all; Free evaluation versions are available for all products. Aspose also provides free support, even before a purchase is made. To test how full version works and build a showcase with all features, developers can request Free evaluation licenses.

Q What about the cloud; does Aspose have a cloud solution?
A Aspose products work just fine in the cloud too. In addition to on-premise solutions, Aspose also offers a completely hosted solution, which is pay as you go. Developers can setup a free account and begin testing within minutes.

Start your free trial at → www.aspose.com

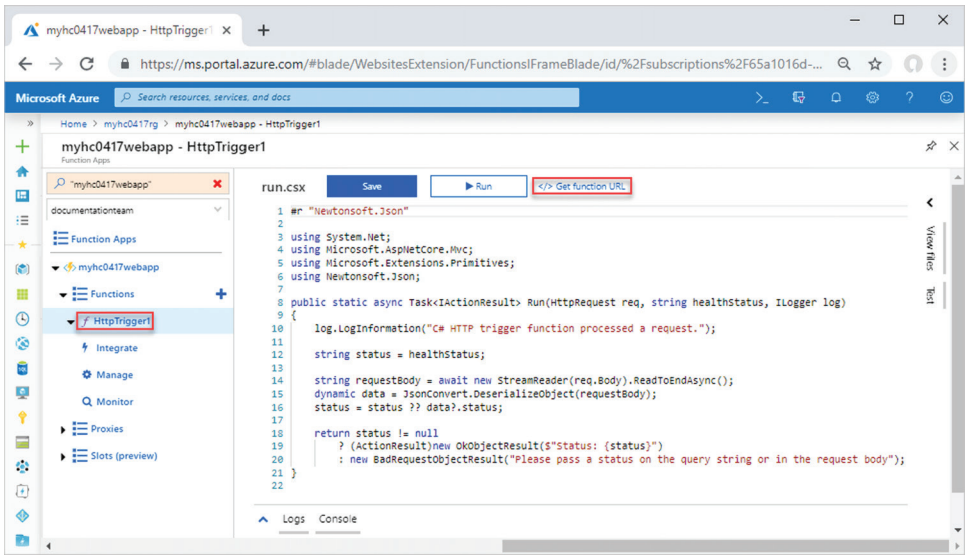


Figure 6 **Working with Azure Functions**

The process of getting set up with Azure Deployment Manager health checks is straightforward. First, you create your health monitors through a health service provider of your choice. Then you create one or more healthCheck steps as part of your Azure Deployment Manager rollout and fill out those steps with the following:

- The URI to the REST API for your health monitors, as defined by your health service provider.
- Authentication information (only API-key style auth is currently supported).
- HTTP status codes or regular expressions that define a healthy response.

To ease health integration, Microsoft has been working with top service health monitoring companies to provide a simple copy/paste solution to integrate health checks into deployments.

Finally, you invoke the healthCheck steps at the appropriate time in your Azure Deployment Manager rollout, and ADM takes care of the rest.

Phases of a Health Check

At this point, ADM knows how to query for the health of your service and at what phases in your rollout to do so. However, it also allows for deep configuration of the timing of these checks. A healthCheck step is executed in a simple three-phase process with configurable durations, rich enough to support the health monitoring needs of the largest Microsoft services composing Azure itself.

After a deployment operation is completed, it may not make sense to check for service health right away, as the update needs to reach a steady state. It takes time for services to start emitting health signals to be aggregated by the health monitoring provider into something useful. Likewise, VMs may be starting for the first time, rebooting, or re-configuring based on new data. The result: Service health may oscillate between healthy and unhealthy states. During the Wait phase service health isn't monitored, so deployed resources have time to bake before beginning the health check process.

It's often impossible to know how long after startup resources take to become stable, so the Elastic phase provides a flexible period for them to settle into a healthy, steady state. When the Elastic phase begins, Azure Deployment Manager polls the provided REST endpoint for service health periodically. The interval is set to three minutes in the Azure Deployment Manager preview, but will be configurable in the future. If the health monitor comes back with signals indicating that a service is unhealthy, those signals are ignored, the Elastic phase is maintained, and polling continues. Once the health monitor comes back with signals indicating that a service is healthy, the Elastic phase ends and the HealthyState phase begins. Thus, the duration specified for the Elastic phase is the maximum time that can be spent polling for service health before a healthy response is considered mandatory.

During the HealthyState phase, service health is continually polled at the same interval as the Elastic phase. The service is expected to maintain healthy signals from the health-monitoring provider for the entire specified duration. If at any point an unhealthy response is detected, ADM stops the entire rollout and returns the REST response carrying the unhealthy service signals. Once the HealthyState phase duration has ended, the healthCheck is complete, and deployment continues to the next step.

Getting Started

In production, you typically use one or more monitoring providers. To ease health integration, Microsoft has been working with top service health monitoring companies to provide a simple copy/paste solution to integrate health checks into deployments. You can view a list of these health monitoring providers at bit.ly/2XOpROG.

For this example, you'll create an Azure Function (bit.ly/2wPgq5d) to simulate a health monitoring service. This function simply takes a status code and returns the same code.

To deploy the Azure Function, use the same Azure CloudShell PowerShell terminal that you used in the previous steps, and then execute the following script. Note that projectName should be less than 12 characters, with only lowercase letters and numbers.

Does your company create enterprise computing solutions for customers?

If so, does your ability to successfully win clients depend on their understanding of the vastly complex Microsoft stack?

CHALLENGE

The Microsoft ecosystem is moving at a blistering pace

Your customers and prospects need clear guidance on some of the key architecture and directional questions for how to proceed.

SOLUTION

Redmond Intelligence Research Reports

Actionable research and intelligence reports from our network of Microsoft experts, including Most Valuable Professionals (MVPs).

Sponsor a Redmond Intelligence Research Report, a Best Practices guide or a Solution Spotlight report. You'll come away with an authoritative, independent report, professionally edited and designed by the team behind Converge360, which you exclusively distribute. Contact us today for more details.

Redmond INTELLIGENCE
redmondintelligence.com

CONTACT

Dan LaBianca | General Manager | **Voice** 818.674.3416 / **E-mail** dlabianca@Converge360.com

CONVERGE 360
An 1105 MEDIA company

It will be reused throughout this example, so be sure to save it. Here's the script:

```
$projectName = <Enter a project name that is used to generate Azure resource names>
$location = <Enter the location (i.e. centralus)>
$resourceGroupName = "${projectName}rg"
New-AzResourceGroup -Name $resourceGroupName -Location $location
New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName
-TemplateUri "https://armtutorials.blob.core.windows.net/admtutorial/
deploy_hc_azure_function.json" -projectName $projectName
```

To verify and test the Azure function, open the Azure portal and open the resource group. The default name is the project name with "rg" appended. Select the app service from the resource group. The default name of the app service is the project name with "webapp" appended. Expand Functions, and then select HttpTrigger1.

As shown in **Figure 6**, select </> Get function URL and then click Copy to copy the URL to the clipboard. The URL should look similar to:

```
https://myhc0417webapp.azurewebsites.net/api/healthStatus/
{healthStatus}?code=hc4Y1wY4AqsskAkVw6WLAN1A4E6aBOh3MbQ3YJRF3XtXgHvooaGOaw==
```

Now replace {healthStatus} in the URL with a status code. In this example, use unhealthy to test the unhealthy scenario, and use either healthy or warning to test the healthy scenario. Create two URLs, one with the unhealthy status, and the other with the healthy status. For example:

```
https://myhc0417webapp.azurewebsites.net/api/healthStatus/
unhealthy?code=hc4Y1wY4AqsskAkVw6WLAN1A4E6aBOh3MbQ3YJRF3XtXgHvooaGOaw==
https://myhc0417webapp.azurewebsites.net/api/healthStatus/
healthy?code=hc4Y1wY4AqsskAkVw6WLAN1A4E6aBOh3MbQ3YJRF3XtXgHvooaGOaw==
```

Save both, as you'll need them to complete this example. You can test the URLs by simply opening them in a browser. You should see a simple page that reads "Status:" followed by the value you put into the URL.

Perform the Second Deployment

To speed things along, the PowerShell scripts that follow will reference an updated service topology and a new rollout file that contains a health check step. This step occurs after the West US deployment, but before the East US deployment. Thus, if the health check fails, the East US service will never be deployed, preventing users in that region from experiencing the failure. This health check is configured to look for "Status: healthy" as a health response from the Azure Function, and anything else as an unhealthy response.

You'll pass the URLs to your Azure Function as parameters when kicking off the script. If you want to download these files to inspect them, you can check out the Service Topology template (bit.ly/2KdzW45) and the Rollout template (bit.ly/31wbP6e).

First, deploy the updated Service Topology. Because the script is rather long, feel free to copy it from here: aka.ms/admhealthdeploy. Here's the script:

```
$projectName = <Enter the same project name used earlier in this example>
$location = <Enter the location (i.e. centralus)>
$resourceGroupName = "${projectName}rg"
$artifactLocation = "http://aka.ms/admtutorialartifactstore" | ConvertTo-
SecureString -AsPlainText -Force
New-AzResourceGroupDeployment `
  -ResourceGroupName $resourceGroupName `
  -TemplateUri "https://armtutorials.blob.core.windows.net/admtutorial/
ADMTemplatesHC/CreateADMServiceTopology.json" `
  -namePrefix $projectName -azureResourceLocation $location `
  -artifactSourceSASLocation $artifactLocation
```

Then verify that the service topology and the underlying resources have been created successfully using the Azure portal, as was shown earlier in **Figure 5**.

Figure 7 **Deploying a Health Integrated Rollout**

```
$projectName = <Enter the same project name used earlier in this example>
$location = <Enter the location (i.e. centralus)>
$managedIdentityID = <Enter the user-assigned managed identity Name>
$healthCheckUrl = <Enter the health check Azure function URL>
$healthCheckAuthAPIKey = $healthCheckUrl.Substring($healthCheckUrl.
IndexOf("?code=")+6, $healthCheckUrl.Length-$healthCheckUrl.
IndexOf("?code=")-6)
$healthCheckUrl = $healthCheckUrl.Substring(0, $healthCheckUrl.
IndexOf("?"))

$resourceGroupName = "${projectName}rg"
$artifactLocation = "http://aka.ms/admtutorialartifactstore" | ConvertTo-
SecureString -AsPlainText -Force

# Create the rollout
New-AzResourceGroupDeployment `
  -ResourceGroupName $resourceGroupName `
  -TemplateUri "https://armtutorials.blob.core.windows.net/admtutorial/
ADMTemplatesHC/CreateADMRollout.json" `
  -namePrefix $projectName `
  -azureResourceLocation $location `
  -artifactSourceSASLocation $artifactLocation `
  -managedIdentityID $managedIdentityID `
  -healthCheckUrl $healthCheckUrl `
  -healthCheckAuthAPIKey $healthCheckAuthAPIKey
```

Next, we'll use the unhealthy Azure Function URL created earlier, along with the Name of the User Assigned Managed identity, to deploy the rollout. As before, feel free to copy this script from aka.ms/admhealthdeploy. **Figure 7** shows the script.

As with the previous rollout, you can check the progress using this script:

```
$projectName = <Enter the same project name used earlier in this example>
$resourceGroupName = "${projectName}rg"
$rolloutName = "${projectName}Rollout"

# Get the rollout status
Get-AzDeploymentManagerRollout `
  -ResourceGroupName $resourceGroupName `
  -Name $rolloutName `
  -Verbose
```

After enough time has passed, checking the status of the rollout will show a failure. This is expected because the unhealthy Azure Function was used! You'll also see that one resource group was created for the West US service, but you won't see the East US resource group because the rollout was automatically stopped before continuing on to other regions. Repeat this deployment with the healthy status URL, and after it has completed, you'll see that both services have successfully deployed!

That's a Wrap

ADM allows you to perform complex, multi-region, multi-subscription deployments that integrate with your existing service health monitoring solutions, using Azure's built-in ARM templates. No additional tools are required, and it's completely free to use. Get started by reviewing the documentation at aka.ms/admdocs, or explore the full tutorial at aka.ms/admtutorial. ■

David Tepper *is a principal program manager at Microsoft on the OneDeploy team in Azure. He has expertise in machine learning and has worked on both shell and deployment technologies for Windows.*

Thanks to the following Microsoft technical experts for reviewing this article: *Cristina del Amo Casado, Jonathan Gao, Devesh Guha Oleti Muni, Sriram Ramaswamy*

TALL™ components

Solid .NET Core PDF Libraries

Q&A with Frank Rem, Founder of TallComponents

Q What gives you satisfaction after 18 years of running your business?

A Feedback from customers who are pleasantly surprised by how quickly we answer their questions and release updates to address issues or feature requests. This is thanks to a great team of engineers.

Q If you release updates quickly, does that mean that you have short release cycles?

A Yes. We have automated building, testing and releasing our products to the maximum extent. Whenever we push commits to our Git repository, our products are built and tested automatically. If successful, our "green button" releases a new version to our website and **nuget.org** in minutes.

Q What is pushing new development of your PDF SDKs?

A Honestly, it is not so much the PDF format. That has been reasonably stable for the last 10 years or so—which is exactly its strength. Dynamic XFA is a notable exception and we have added great support for it. Most new development comes from Microsoft's cross-platform efforts. All of our products are now compatible with .NET Core and Xamarin.

Q Did you have any trouble raising growth capital?

A Actually, we never did. TallComponents has been profitable almost from day one. There is no outside funding or debt. All new investments are paid out of our positive cash flow.

For more information, please visit → www.tallcomponents.com

LIVE! 360
TECH EVENTS WITH PERSPECTIVE

Artificial Intelligence LIVE!

AI FOR DEVELOPERS AND DATA SCIENTISTS

November 17-22, 2019 | ORLANDO
Royal Pacific Resort at Universal

A Practical Introduction To AI & Machine Learning for Enterprise Developers and Data Scientists

Artificial Intelligence Live! is an innovative, new conference for current and aspiring developers, data scientists, and data engineers covering artificial intelligence (AI), machine learning, data science, Big Data analytics, IoT & streaming analytics, bots, and more. You can expect real-world training on the languages, libraries, APIs, tools and cloud services you need to implement real AI and machine learning solutions, today and into the future.

TRACK TOPICS INCLUDE:

- AI Application Development
- Analytics
- Bots
- Data Science and Machine Learning
- Internet of Things
- and more!

SAVE $500 With Summer Savings When You Register by August 16!
Use Promo Code MSDN

A Part of Live! 360: The Ultimate Education Destination

6 Great Events, 1 Low Price!

Visual Studio LIVE! — EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS
SQL Server LIVE! — TRAINING FOR DBAs AND IT PROS
TECHMENTOR — IN-DEPTH TRAINING FOR IT PROS
Office & SharePoint LIVE! — ON-PREMISE, CLOUD, & CROSS-PLATFORM TRAINING
Artificial Intelligence LIVE! — AI FOR DEVELOPERS AND DATA SCIENTISTS
Cloud & Containers LIVE! — CLOUD-NATIVE, PAAS & SERVERLESS COMPUTING

AttendAILive.com

EVENT PARTNER: Microsoft
PLATINUM SPONSOR: accusoft
SILVER SPONSOR: aws
SUPPORTED BY: msdn magazine, Redmond Channel Partner, Redmond Magazine, Virtualization & Cloud Review, Visual Studio Magazine

AGENDA-AT-A-GLANCE

#LIVE360

AI Application Development	Analytics	Bots	Data Science and Machine Learning	Internet of Things

START TIME	END TIME		
		Artificial Intelligence Live! Full Day Hands-On Lab: Sunday, November 17, 2019	
7:15 AM	9:00 AM	Registration • Coffee and Morning Pastries	
9:00 AM	6:00 PM	**AIS01** Hands-On Lab: Build Your Own A.I. Powered Robot - *Henk Boelman*	
2:00 PM	7:00 PM	Pre-Conference Registration - Royal Pacific Resort Conference Center	
START TIME	**END TIME**	**Artificial Intelligence Live! Pre-Conference Workshop: Monday, November 18, 2019**	
7:00 AM	8:30 AM	Registration • Coffee and Morning Pastries	
8:30 AM	5:30 PM	**AIM01** Workshop: AI Application Development - *(Andrew to talk to MSFT)*	**AIM02** Workshop: Big Data and Machine Learning with Hadoop, Spark, and SQL Server 2019 - *Andrew Brust*
6:30 PM	8:00 PM	Dine-A-Round Dinner @ Universal CityWalk - 6:30pm - Meet at Conference Registration Desk to walk over with the group	
START TIME	**END TIME**	**Artificial Intelligence Live! Day 1: Tuesday, November 19, 2019**	
7:00 AM	8:00 AM	Registration • Coffee and Morning Pastries	
8:00 AM	9:00 AM	**ARTIFICIAL INTELLIGENCE LIVE! KEYNOTE: Simplify and Accelerate AI with Azure Machine Learning** *Sujatha Sagiraju, Group Program Manager, Azure Cloud & AI Group, Microsoft*	
9:15 AM	10:30 AM	**AIT01** Make Your App See, Hear and Think with Cognitive Services - *Roy Cornelissen*	**AIT02** Data Insights with End-2-End Azure Analytics - Scott Klein
10:30 AM	11:00 AM	Networking Break • Visit the EXPO - *Pacifica 7*	
11:00 AM	12:00 PM	**LIVE! 360 KEYNOTE: To Be Announced**	
12:00 PM	12:45 PM	Lunch • Visit the EXPO	
12:45 PM	1:30 PM	Dessert Break • Visit the EXPO	
1:30 PM	1:50 PM	**AIT03** Fast Focus: Personal Assistants vs. Bots - *Brian Randell*	**AIT04** Fast Focus: The Sunny Side of AI - *Henk Boelman*
2:00 PM	2:20 PM	**AIT05** Fast Focus: ??? AR vs. VR vs. MR - What Does it All Mean? - *Nick Landry*	**AIT06** Fast Focus: AutoML Crash Course - *Andrew Brust*
2:20 PM	2:45 PM	Networking Break • Visit the EXPO - *Pacifica 7*	
2:45 PM	4:00 PM	**AIT07** Did You Know Cognitive Services Could Do This? A Computer Vision Quest - *Andreas Erben*	**AIT08** AI and Analytics with Apache Spark on Azure Databricks - *Andrew Brust*
4:15 PM	5:30 PM	**AIT09** Bot Building 101 with the Microsoft Bot Framework - *Brian Randell*	**AIT10** Diving Into Deep Learning with Databricks - *Ginger Grant*
5:30 PM	7:30 PM	Exhibitor Reception - *Pacifica 7*	
START TIME	**END TIME**	**Artificial Intelligence Live! Day 2: Wednesday, November 20, 2019**	
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries	
8:00 AM	9:15 AM	**AIW01** An Introduction to HoloLens & Mixed Reality for the Enterprise - *Nick Landry*	**AIW02** Machine Learning 101 for Developers - *Jen Underwood*
9:30 AM	10:45 AM	**AIW03** Building a Holographic AI Assistant with ASP.NET Bots, Natural Language, & Mixed Reality - *Nick Landry*	**AIW04** Getting Started with Azure Machine Learning Services - *Henk Boelman*
10:45 AM	11:30 AM	Networking Break • Visit the EXPO - *Pacifica 7*	
11:30 AM	12:30 PM	**LIVE! 360 KEYNOTE: To Be Announced**	
12:30 PM	1:30 PM	Birds-of-a-Feather Lunch	
1:30 PM	2:00 PM	Dessert Break • Visit the EXPO	
2:00 PM	3:15 PM	**AIW05** Practical Internet of Things for the Microsoft Developer - *Eric D. Boyd*	**AIW06** DevOps for Artificial Intelligence, the Road to Production - *Henk Boelman*
3:15 PM	4:00 PM	Networking Break • Visit the EXPO • Expo Raffle @ 3:30 p.m. - *Pacifica 7*	
4:00 PM	5:15 PM	**AIW07** The Edge of Tomorrow. Use Visual Studio Code and Raspberry Pi to Create IoT Edge Solutions - *Nicholas McCollum*	**AIW08** Visualization Best Practices for Machine Learning Applications - *Jen Underwood*
7:30 PM	9:00 PM	Live! 360 Dessert Luau - *Wantilan Pavilion*	
START TIME	**END TIME**	**Artificial Intelligence Live! Day 3: Thursday, November 21, 2019**	
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries	
8:00 AM	9:15 AM	**AIH01** Machine Learning the .NET Way - *Bri Achtman [TBD]*	**AIH02** Machine Learning with Power BI - *Raj Krishnan*
9:30 AM	10:45 AM	**AIH03** Demystifying User Management for Voice Apps - *Heather Downing*	**AIH04** Launching A Data Science Project: Cleaning Is Half the Battle - *Kevin Feasel*
11:00 AM	12:00 PM	**ARTIFICIAL INTELLIGENCE LIVE! PANEL DISCUSSION: AI: For Data Scientists or Developers?** *Andrew Brust (moderator), Kevin Feasel, Raj Krishnan, Heather Downing*	
12:00 PM	1:00 PM	Lunch - *Oceana Ballroom*	
1:00 PM	2:15 PM	**AIH06** Google vs. Alexa: Battle of the Bots - *Heather Downing*	**AIH06** Data Orchestration and Data Flow with Azure Data Factory - *Raj Krishnan*
2:30 PM	3:45 PM	**AIH07** Build a Facebook Messenger Bot Customer Service Experience with Twilio Autopilot - *Lizzie Siegle [TBD]*	**AIH08** The 8 "C's" of Artificial Intelligence – Being Successful with AI - *Jen Stirrup*
4:00 PM	5:00 PM	Next? Live! 360 Networking Event *Brian Randell, Raj Krishnan, Heather Downing*	
START TIME	**END TIME**	**Artificial Intelligence Live! Post-Conference Workshop: Friday, November 22, 2019**	
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries	
8:00 AM	5:00 PM	**AIF01** Workshop: From Business Intelligence to Business Analytics with the Microsoft Data Platform - *Jen Stirrup*	

Speakers and sessions subject to change

Affairs of State: Serverless and Stateless Code Execution with Azure Functions

Sandeep Alur and Srikantan Sankaran

The serverless architecture pattern has emerged as a backbone of legacy modernization in the public cloud. Most platform services today are conceptualized, packaged and offered with a focus on high throughput, low cost, and superior ease of use and adoption. They generally address economics of scale and provide a platform for solution designers to realize capabilities that would otherwise require significant complexity and infrastructure.

Of course, "Pay as you Go" as a commercial model is one of the key tenets of the public cloud. "Serverless" extends this model further by making key decisions on when and how compute is provisioned, which helps manage costs during runtime. To elaborate on this philosophy and successfully adopt this pattern, it helps to consider Azure Functions as a key service as we lay out design and implementation tenets.

Azure Functions host custom code without requiring the user to specify or provision the compute on which it should run. By provisioning compute resources on demand, it supports unpredictable workloads and exhibits true "pay per use" qualities. Resources scale on the fly to match incoming loads, ensuring high availability and minimizing management overhead.

Azure Functions was envisioned as a service tasked primarily with handling rapid bursts of events at scale in a stateless manner. But the role of the technology has expanded significantly. In this article, we explore the different workload and deployment scenarios of Azure Functions and how they help enterprises implement serverless capabilities.

Some of the technologies discussed in this article are in preview. All information is subject to change.

This article discusses:

- Scenarios to leverage Durable Functions
- Premium tier of Azure Functions and features offered
- Development and configuration best practices

Technologies discussed:

Azure Functions, Durable Functions, Durable Entities

Design Options

The serverless capabilities of Azure Functions can be used across a variety of use-case scenarios, enabling organizations to efficiently deploy functionality, without becoming mired in overhead and infrastructure. There are important design considerations that teams should consider as they embark. Let's explore some of those here.

Stateless vs. Stateful Execution The single most common use case for Azure Functions involves executing rapid bursts of stateless custom code at scale. These scenarios are characterized by their short duration—no more than five minutes—and code that holds no state or locks across requests. There's no requirement to maintain a strict sequence in the execution of requests or implementing a transaction context across requests.

Note that while some of these scenarios aren't meant for execution through Stateless Azure Functions, they can be implemented through additional design elements.

Azure Functions provides hooks for integration with a host of Azure Services through triggers and input and output binding. It can also be used to intersperse process flows implemented using Azure Logic App with custom logic.

Azure Functions can be hosted using either the Consumption tier that provides true Serverless characteristics, or via an App Service plan that uses a dedicated capacity option. In the latter case, some of the limitations in the Consumption tier, like the five-minute timeout per request, will not apply. You can learn more about the differences between these hosting options at bit.ly/2xcCXsO.

> Durable Functions rely on an Orchestrating Function that's responsible for handling state and ensuring reliability across numerous Activity Functions.

What happens when your custom code needs to orchestrate a set of stateless tasks for sequential execution, or calls for parallel execution of stateless tasks at scale with an aggregation of results in a fan-out and fan-in scenario? Or consider long-lived orchestrations that need to be monitored for state changes through external events or need human intervention during the execution lifecycle. Or consider the need for reliable execution that implements the notion of retry from the last failure point, through checkpoints and replay, rather than a regenerative execution of all tasks.

This stuff requires a lot more than a simple stateless Azure Functions service can provide. In these instances, the Azure Durable Functions extension can be used to implement robust design patterns. Durable Functions relies on an Orchestrating Function that's responsible for handling state and ensuring reliability across numerous Activity Functions.

In short, the Activity Function is like the Stateless Azure Functions, in that it handles all the I/O operations through input and output bindings. But unlike Stateless Azure Functions, it's triggered exclusively through an Orchestrating Function.

Figure 1 **Chaining Multiple Activity Functions**

```
public static async Task<object> Run (DurableOrchestrationContext context)
{
  try
  {
    var output1 = await context.CallActivityAsync<object>("ActivityFunction1");
    var output2 = await context.CallActivityAsync<object>(" ActivityFunction2",
      output1);
    var output3 = await context.CallActivityAsync<object>(" ActivityFunction3",
      output2);
    return  await context.CallActivityAsync<object>(" ActivityFunction4", output3);
  }
  catch (Exception)
  {
    // Error handling or compensation goes here.
  }
}
```

The code in **Figure 1** shows an example of an Orchestration Function that implements a chaining of multiple Activity Functions.

Durable Entities, a feature recently added to Durable Functions, implements the Virtual Actor pattern as a Function. Stateful, Durable Entities can be managed using orchestrating Functions, and accessed from Orchestration Clients. Support for Durable Entities is in alpha stage at the time of this writing and is not yet ready for production deployments. To understand what Durable Entities are and how Durable Functions offers them, refer to bit.ly/2RAA12B.

Workloads and Binding We've already discussed that serverless architectures like Azure Functions are inherently suited for unpredictable workloads, where periods of inactivity are punctuated by sudden bursts of demand. For these scenarios, the Consumption tier or the Premium tier would likely be the best choice.

For continuous and predictable loads, however, it might be more cost effective to deploy solutions to pre-provisioned, dedicated resources, using the App Service tier for Azure Functions.

Another thing to consider is the balance between connections baked into code and declarative binding. The latter are design-time configurations for Functions that eliminate the need for the Function code to manage connections to triggers and input and output binding sources. When the function app scales out to multiple instances under load, connections implemented via declarative binding are reused across all instances.

Connections implemented in code, by contrast, require that the developer manage the data source connections across requests. Creating and disposing connections in the Function code for every request limits scalability and performance under load. However, taking this approach is unavoidable, for instance, when the data source isn't supported through design-time bindings. In such a case, care should be taken to reuse client-side connections across requests, using static or singleton client connections.

Guidance on connection management in Functions is available at bit.ly/2ZSf5a3.

Control in Execution

Serverless architectures excel at providing high availability, auto scaling and resiliency, with minimal user intervention. In this regard, they're superior even to Platform-as-a-Service (PaaS) services. However, there are certain configuration options available in serverless environments that can be used to tune the performance characteristics of Azure Functions. Let's walk through those now.

Batching for Performance: The throughput and performance of a function improves when it's configured to receive incoming messages in batches, rather than employing per-message execution. However, this execution is dependent on the Azure Service configured in the binding and whether or not the Azure Function trigger for that service supports batching of messages. For example, Azure Event Hubs supports batching of messages consumed by client applications. The batch size is specified in the host.json file of the function app.

Concurrency of Requests: You can determine how many requests are handled by each instance of a function app by specifying the concurrency of the request triggers. You can tweak this number to suit the nature of the workloads being processed and then let the

LIVE! 360
TECH EVENTS WITH PERSPECTIVE

SQL Server LIVE!
TRAINING FOR DBAs AND IT PROS

November 17-22, 2019 | ORLANDO
Royal Pacific Resort at Universal

Data. Driven.

At SQL Server Live!, we want to help DBAs, analytics experts, systems administrators, and developers like you do more with your SQL Server and Microsoft Data Platform investment. We want to help you improve performance, get insights from your data, step up security, and take advantage of new features across the platform. We want you to master reporting, BI, data integration, and developer tools and techniques. We will show attendees how to adopt new techniques, improve old approaches, explore and visualize their data, and modernize SQL Server infrastructure. We teach and demonstrate how to integrate cloud-based data services, run SQL Server technology in the cloud, use Power BI and Analysis Services, and plan for SQL Server recovery and availability.

TRACK TOPICS INCLUDE:

- Business Intelligence
- SQL Server Administration & Maintenance
- SQL Server for Developers
- SQL Server Features & Components
- SQL Server Performance Tuning & Optimization

SAVE $500 With Summer Savings When You Register by August 16!
Use Promo Code MSDN

A Part of Live! 360: The Ultimate Education Destination

6 Great Events, 1 Low Price!

Visual Studio LIVE! — EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS
SQL Server LIVE! — TRAINING FOR DBAs AND IT PROS
TECHMENTOR — IN-DEPTH TRAINING FOR IT PROS
Office & SharePoint LIVE! — ON-PREMISE, CLOUD, & CROSS-PLATFORM TRAINING
Artificial Intelligence LIVE! — AI FOR DEVELOPERS AND DATA SCIENTISTS
Cloud & Containers LIVE! — CLOUD-NATIVE, PAAS & SERVERLESS COMPUTING

SQLLive360.com

EVENT PARTNER: Microsoft
PLATINUM SPONSOR: accusoft
SILVER SPONSOR: aws
SUPPORTED BY: msdn magazine, Redmond Channel Partner, Redmond Magazine, Virtualization & Cloud Review, Visual Studio Magazine

AGENDA-AT-A-GLANCE

#LIVE360

Business Intelligence	SQL Server Administration & Maintenance	SQL Server Features & Components	SQL Server for Developers	SQL Server Performance Tuning and Optimization

START TIME	END TIME			
SQL Server Live! Full Day Hands-On Lab: Sunday, November 17, 2019				
7:15 AM	9:00 AM	Registration • Coffee and Morning Pastries		
9:00 AM	6:00 PM	**SQS01** Hands-On Lab: The A to Z of Cloud-based SQL Server Solutions - *Allan Hirt*		
2:00 PM	7:00 PM	Pre-Conference Registration - Royal Pacific Resort Conference Center		
START TIME	**END TIME**	**SQL Server Live! Pre-Conference Workshops: Monday, November 18, 2019**		
7:00 AM	8:30 AM	Registration • Coffee and Morning Pastries		
8:30 AM	5:30 PM	**SQM01** Workshop: Advanced Data Protection - Security and Privacy in SQL Server - *Thomas LaRock & Karen Lopez*	**SQM02** Workshop: SQL Server 2019 - *Leonard Lobel*	
6:30 PM	8:00 PM	Dine-A-Round Dinner @ Universal CityWalk - 6:30pm - Meet at Conference Registration Desk to walk over with the group		
START TIME	**END TIME**	**SQL Server Live! Day 1: Tuesday, November 19, 2019**		
7:00 AM	8:00 AM	Registration • Coffee and Morning Pastries		
8:00 AM	9:00 AM	**SQL SERVER LIVE! KEYNOTE: To Be Announced**		
9:15 AM	10:30 AM	**SQT01** AWS Versus Azure: Data Services Comparison - *Thomas LaRock*	**SQT02** Availability Fundamentals for SQL Server - *Allan Hirt*	**SQT03** Power BI: What Have You Done For Me Lately - *Andrew Brust*
10:30 AM	11:00 AM	Networking Break • Visit the EXPO - *Pacifica 7*		
11:00 AM	12:00 PM	**LIVE! 360 KEYNOTE: To Be Announced**		
12:00 PM	12:45 PM	Lunch • Visit the EXPO		
12:45 PM	1:30 PM	Dessert Break • Visit the EXPO		
1:30 PM	1:50 PM	**SQT04** Fast Focus: Automation for the DBA: Embrace Your Inner Sloth - *William Durkin*	**SQT05** Fast Focus: Who's Tinkling in Your Data Lake? - *Karen Lopez*	
2:00 PM	2:20 PM	**SQT06** Fast Focus: From Adaptive to Intelligent: Query Processing in SQL 2019 - *Hugo Kornelis*	**SQT07** Fast Focus: Performance Tuning Without Changing Code - *Thomas LaRock*	
2:20 PM	2:45 PM	Networking Break • Visit the EXPO - *Pacifica 7*		
2:45 PM	4:00 PM	**SQT08** SQL Server 2019 Deep Dive - *Scott Klein*	**SQT09** SQL Server In-Memory Database Objects - *Denny Cherry*	**SQT10** What's New in SQL Server 2019 for Analytics - *Thomas LeBlanc*
4:15 PM	5:30 PM	**SQT11** Microsoft's New Database Experimentation Assistant (DEA) - *Mindy Curnutt*	**SQT12** Common Troubleshooting Techniques for Availability Groups and Failover Cluster Instances - *Allan Hirt*	**SQT13** Power BI implementation guidance: from Pro to Premium and Desktop to Embedded - *Thomas LeBlan*
5:30 PM	7:30 PM	Exhibitor Reception - *Pacifica 7*		
START TIME	**END TIME**	**SQL Server Live! Day 2: Wednesday, November 20, 2019**		
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries		
8:00 AM	9:15 AM	**SQW01** An Introduction to Spatial Data in SQL Server - *Mindy Curnutt*	**SQW02** It's Broken, Now What?! Practical Problem Solving - *William Durkin*	**SQW03** Getting Started with Apache Spark - *Kevin Feasel*
9:30 AM	10:45 AM	**SQW04** Blockchain for the DBA & Data Professional - *Karen Lopez*	**SQW05** SQL Server Open Query Store - *William Durkin*	**SQW06** PolyBase in Action - *Kevin Feasel*
10:45 AM	11:30 AM	Networking Break • Visit the EXPO - *Pacifica 7*		
11:30 AM	12:30 PM	**LIVE! 360 KEYNOTE: To Be Announced**		
12:30 PM	1:30 PM	Birds-of-a-Feather Lunch		
1:30 PM	2:00 PM	Dessert Break • Visit the EXPO		
2:00 PM	3:15 PM	**SQW07** SQL Data Discovery and Classification - *Karen Lopez*	**SQW08** Things You Should Never Do In Microsoft SQL Server - *Denny Cherry*	**SQW09** Deep Dive into Power BI Administration and Security Internals - *Ginger Grant*
3:15 PM	4:00 PM	Networking Break • Visit the EXPO • Expo Raffle @ 3:30 p.m. - *Pacifica 7*		
4:00 PM	5:15 PM	**SQW10** Migrating SSIS Workloads to Azure Data Factory - *Joshua Luedeman*	**SQW11** Improve Your Database Performance in Seven Simple Steps - *Hugo Kornelis*	**SQW12** Performance Tuning Power BI - Ginger Grant
7:30 PM	9:00 PM	Live! 360 Dessert Luau - *Wantilan Pavilion*		
START TIME	**END TIME**	**SQL Server Live! Day 3: Thursday, November 21, 2019**		
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries		
8:00 AM	9:15 AM	**SQH01** Deep Dive into Adaptive Query Processing - *Hugo Kornelis*	**SQH02** Using Modular Scripts to Perform SQL Compliance Audits in Seconds - *Chris Bell*	**SQH03** DataOps and Business Intelligence Testing Approaches - *Jen Stirrup*
9:30 AM	10:45 AM	**SQH04** Data Security and Privacy Techniques for Modern Databases - *Thomas LaRock*	**SQH05** Building Your First SQL Server Container Lab in Docker - *Chris Bell*	**SQH06** To Be Announced
11:00 AM	12:00 PM	**SQL SERVER LIVE! PANEL DISCUSSION: SQL Server is Dead, Long Live SQL Server!** *Thomas LaRock (moderator), Karen Lopez, Bradley Ball, Jen Stirrup, Joseph D'Antoni*		
12:00 PM	1:00 PM	Lunch - *Oceana Ballroom*		
1:00 PM	2:15 PM	**SQH07** Machine Learning with R in Azure SQL Database - *Bradley Ball*	**SQH08** Building a Better Data Solution: Microsoft SQL Server and Azure Data Services - *Joseph D'Antoni*	**SQH09** Data Preparation: A framework with Power BI, Power Query, and Power BI Dataflows - *Jen Stirrup*
2:30 PM	3:45 PM	**SQH10** TimescaleDB on Azure Database for PostgreSQL - *Bradley Ball*	**SQH11** Containers, Pods, and Databases - Learning About the Future of Infrastructure - *Joseph D'Antoni*	**SQH12** Getting Started with Power BI Report Server - *Joshua Luedeman*
4:00 PM	5:00 PM	Next? Live! 360 Networking Event *Thomas LaRock, Bradley Ball, Joseph D'Antoni, Karen Lopez & Jen Stirrup*		
START TIME	**END TIME**	**SQL Server Live! Post-Conference Workshops: Friday, November 22, 2019**		
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries		
8:00 AM	5:00 PM	**SQF01** Workshop: Azure SQL Data Warehouse: A Full Day of the Transformative New NDA Features - *Bradley Ball, Joshua Luedeman, & Gareth Swanepoel*	**SQF02** Workshop: SQL Server High Performance Development - *Joseph D'Antoni*	

Speakers and sessions subject to change

PRODUCED BY
CONVERGE360 an 1105 MEDIA company

CONNECT WITH LIVE! 360
twitter.com/live360 @live360
facebook.com Search "Live 360"
instagram.com @live360_events
linkedin.com Join the "Live! 360" group!

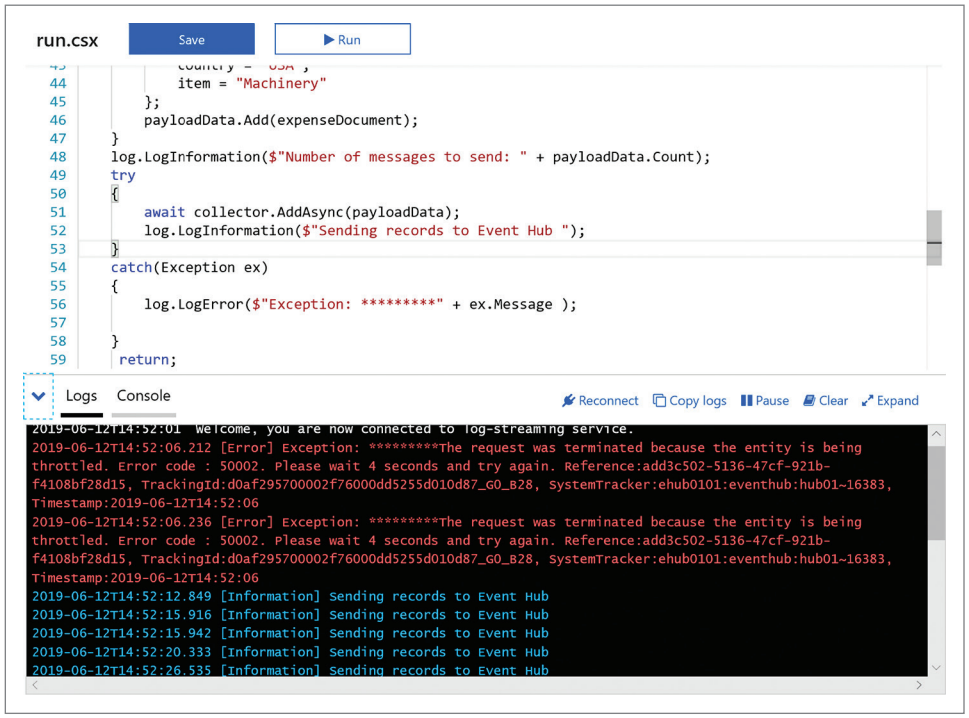


Figure 2 **Handling Exceptions in Output Binding**

function app scale as appropriate. This configuration is specified in the host.json file of the function app.

Scaling Out: Azure Functions automatically scales the infrastructure to handle incoming loads with no user intervention. How steeply the scale out happens depends on the number of events that need to be processed. However, there are certain limits within which a single function app can scale. In the consumption tier, each function app instance can handle multiple concurrent requests, and each function app itself can scale out to at most 200 instances (see bit.ly/2JlU7tV). In addition, the consumption tier doesn't provide the option to choose the capacity of the compute resources for this workload.

How do you scale out beyond these maximum limits? One option would be to clone the function app and have the client applications distribute the load between them. An alternative would be to leverage the premium tier of Azure Functions, which lets you choose from a variety of compute sizes suitable for the workload at hand. Note that the premium tier for function apps is available in public preview at this time and cannot be used in production workloads.

Cold Start Ups: After a period of inactivity when all the instances have timed out, a sudden load applied to a function app can produce a delay as instances must start to actively handle requests again. This delay might not be acceptable when handling certain mission-critical scenarios.

Cost-Performance Trade-offs: The premium tier in function apps provides options to specify the number of warm instances that are always available as a standby, to prevent cold start-up delays. Apart from exhibiting the same serverless capabilities as the consumption tier, it provides additional benefits, like the option to choose from a variety of compute sizes on which to run the instances, the ability to access resources inside a virtual network and exemption from a default timeout duration as in the consumption tier. All these features come at a cost, but the benefits they offer can more than compensate for the trade-offs in mission-critical use cases.

Note that Azure Functions is available both on Linux and Windows compute, but not all the features are available in both offerings. Azure Functions on Linux is currently available in preview.

Portability

Azure Functions comprises a runtime and a scaling component. The former is available as a Docker container, which means Functions can be deployed on-premises, at the edge or on Azure Cloud. The scaling capabilities are achieved by deploying a Functions runtime inside a Kubernetes cluster in conjunction with the open source component, Kubernetes-based Event-driven autoscaling (KEDA).

We've already covered how Azure Functions can run natively on Azure, but Functions can also be deployed to a Kubernetes cluster. On Azure, KEDA can be deployed to a Kubernetes cluster created using Red Hat OpenShift or Azure Kubernetes Service (AKS).

> Azure Functions automatically scales the infrastructure to handle incoming loads with no user intervention.

On the edge or on-premises, KEDA can be installed in a Kubernetes cluster using the Core Tools for Functions. Functions packaged as containers would run in the Kubernetes pods.

Scaling of the containers running Functions happens based on the number of messages in the queues configured as triggers for the function app. Refer to guidance at bit.ly/2J9gYI0 for more on implementing KEDA for Functions and Kubernetes.

Development Best Practices

Covered in this section are some valuable best practices for the development of Azure Functions. Let's start with exception handling from triggers and bindings. Azure Functions provides declarative bindings for triggers, input and output data sources. When there's an error in executing triggers or invoking data sources, these must be caught explicitly in code. The exception handler

should implement the semantics of retries with exponential back-off. Services like Azure Service Bus Queues and Blob Stores implicitly support retries in the triggers, but for other services, these must be explicitly implemented.

To capture runtime exceptions when using other services like Event Hubs, the output binding must use the IAsyncCollector type, instead of an out parameter. Calling the AddAsync method of the collector inside an exception handling block ensures that the runtime exceptions are caught.

In **Figure 2**, a transient exception with code 50002 is thrown by Event Hubs, which indicates that the requests are being throttled. In the exception handler, retry logic that uses an exponential back-off policy should be implemented.

The documentation page at bit.ly/31UDLRl covers how an exponential back-off could be implemented in a durable function app.

Figure 3 shows the metrics from the Event Hubs instance after it's subjected to load, to simulate request throttling.

Authentication and Identities

Azure Functions supports different forms of authentication of client applications. This could be in the form of embedding an authentication key in the HTTP header of the request to the function app, or using an oAuth token when users access the Function through a Web application that implements Azure AD sign-in (bit.ly/2XzrulR). In some cases when the function app is exposed to the external users through Azure API Management, authorization of the user requests could be done at the gateway using the subscription keys embedded in the request.

When the custom code in Azure Functions needs to access other services in Azure, it's a good practice to bootstrap the credentials with it using system-based or user-based managed identities (bit.ly/2IPntI0). Currently, not all services in Azure support access using managed identities. The list of supported services can be found at bit.ly/2ISVUYp.

In the following code snippet, taken from a Visual Studio 2017 Azure Functions project, I show how managed identity can be used to connect to and access Azure SQL Database. For security, you no longer need to store SQL authentication credentials inside the connection string and then store the connection string itself in Azure Key Vault.

```
string connString =
  "Server = tcp:onedbserver.database.windows.net,1433; Database = onedb;";
  using (SqlConnection con = new SqlConnection(connString))
  {
    con.AccessToken = await (
      new AzureServiceTokenProvider()).GetAccessTokenAsync(
        "https://database.windows.net/");
    con.Open();
    try
    {
      using (SqlCommand command =
        new SqlCommand("SELECT * FROM SalesLT.Product", con))
      {
        SqlDataReader reader = command.ExecuteReader();
...
```

Next, publish the function app to Azure, enable Managed Identity for it, and provide database access to it. Refer to the steps documented at bit.ly/2JaGOfr that explain the process of service principal identity creation and assignment to an Azure SQL Database.

Azure App Service and Azure Functions provide convenience options to simplify integration with Azure Key Vault using managed identities to access secrets. Check out the article at bit.ly/2X1f7ed for detailed guidance in implementing this feature.

Wrapping up

Azure Functions provides enterprises with a serverless way to develop, deploy and run custom code that can run anywhere, in Azure, on-premises or on the edge. The advent of additional features to the platform, in the form of durable functions and entities, unlocks the opportunities to extend serverless to a host of new requirements and design patterns. The extensive integration of Azure Functions with other Azure services and development tools ensures that developers are most productive when building applications, regardless of the programming language and platform, using an IDE of their choice. ■

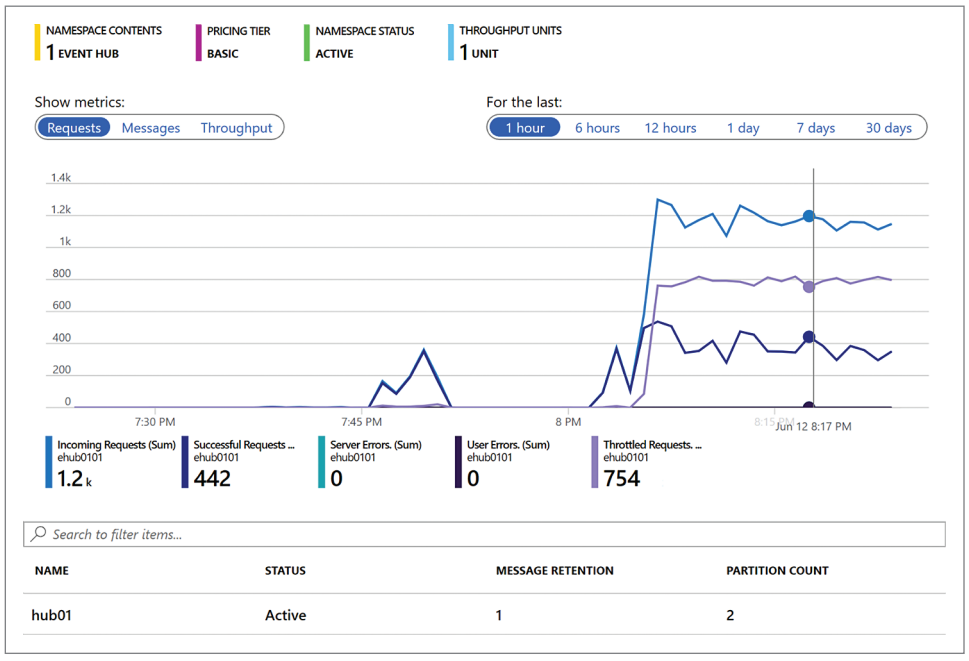


Figure 3 **Monitoring Event Hub for Request Throttling**

Sandeep Alur *is director of Partner Engagements (ISV) at Microsoft. His experience ranges from the original dotcom days to the distributed/cloud computing era to the next generation of intelligent computing. Reach him at Sandeep.Alur@microsoft.com.*

Srikantan Sankaran *is a principal technical evangelist in the One Commercial Partner team in India, based out of Bangalore. He works with numerous ISVs in India and helps them architect and deploy their solutions on Microsoft Azure. Reach him at Srikantan.Sankaran@microsoft.com.*

Thanks to the following Microsoft technical expert for reviewing this article: *Hemant Kathuria*

LIVE! 360
TECH EVENTS WITH PERSPECTIVE

Cloud & Containers LIVE!

CLOUD-NATIVE, PAAS & SERVERLESS COMPUTING

November 17-22, 2019 | ORLANDO
Royal Pacific Resort at Universal

New In 2019!

All Things Cloud-Native

Cloud & Containers Live! has the content you need to understand and leverage Kubernetes, Docker, Azure, AWS, and other cloud-native technologies. Whether you are looking for IT infrastructure, software development, or DevOps information, this conference is for you. Our expert speakers cover topics including cloud-native software design, development, release, deployment, operations, instrumentation/monitoring, and maintenance. We believe the future of most computing is the cloud: containers, PaaS, and serverless. This conference provides you with the knowledge you need to succeed with cloud-based infrastructure and software development.

TRACK TOPICS INCLUDE:

- **Cloud-Native Development**
- **Cloud-Native DevOps**
- **Cloud-Native Infrastructure**

SAVE $500 With Summer Savings When You Register by August 16!
Use Promo Code MSDN

A Part of Live! 360: The Ultimate Education Destination

6 Great Events, 1 Low Price!

Visual Studio LIVE! — EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS
SQL Server LIVE! — TRAINING FOR DBAs AND IT PROS
TECHMENTOR — IN-DEPTH TRAINING FOR IT PROS
Office & SharePoint LIVE! — ON-PREMISE, CLOUD, & CROSS-PLATFORM TRAINING
Artificial Intelligence LIVE! — AI FOR DEVELOPERS AND DATA SCIENTISTS
Cloud & Containers LIVE! — CLOUD-NATIVE, PAAS & SERVERLESS COMPUTING

CCLive360.com

EVENT PARTNER: Microsoft
PLATINUM SPONSOR: accusoft
SILVER SPONSOR: aws
SUPPORTED BY: msdn magazine, Redmond Channel Partner, Redmond Magazine, Virtualization & Cloud Review, Visual Studio Magazine

AGENDA-AT-A-GLANCE

#LIVE360

Cloud-Native Development	Infrastructure	Cloud DevOps

START TIME	END TIME	
Cloud & Containers Live! Full Day Hands-On Lab: Sunday, November 17, 2019		
7:15 AM	9:00 AM	Registration • Coffee and Morning Pastries
9:00 AM	6:00 PM	**CCS01** Hands-On Lab: Hands-On with Cloud-Native .NET Development - *Rockford Lhotka*
2:00 PM	7:00 PM	Pre-Conference Registration - Royal Pacific Resort Conference Center
START TIME	**END TIME**	**Cloud & Containers Live! Pre-Conference Workshops: Monday, November 18, 2019**
7:00 AM	8:30 AM	Registration • Coffee and Morning Pastries
8:30 AM	5:30 PM	**CCM01** Workshop: Building, Running & Continuously Deploying Microservices with Docker Containers on Azure - *Marcel de Vries & René van Osnabrugge* / **CCM02** Workshop: Kubernetes Zero to Hero - Installation, Configuration, and Application Deployment - *Anthony Nocentino*
6:30 PM	8:00 PM	Dine-A-Round Dinner @ Universal CityWalk - 6:30pm - Meet at Conference Registration Desk to walk over with the group
START TIME	**END TIME**	**Cloud & Containers Live! Day 1: Tuesday, November 19, 2019**
7:00 AM	8:00 AM	Registration • Coffee and Morning Pastries
8:00 AM	9:00 AM	**CLOUD & CONTAINERS LIVE! KEYNOTE: To Be Announced**
9:15 AM	10:30 AM	**CCT01** Serverless with Knative - *Mete Atame* / **CCT02** Intro to Kubernetes - *Chris Kinsman*
10:30 AM	11:00 AM	Networking Break • Visit the EXPO - *Pacifica 7*
11:00 AM	12:00 PM	**LIVE! 360 KEYNOTE: To Be Announced**
12:00 PM	12:45 PM	Lunch • Visit the EXPO
12:45 PM	1:30 PM	Dessert Break • Visit the EXPO
1:30 PM	1:50 PM	**CCT03** Fast Focus: Serverless in Azure 101 - *Eric D. Boyd* / **CCT04** Fast Focus: Containers on AWS - *Chris Kinsman*
2:00 PM	2:20 PM	**CCT05** Fast Focus: Deploying Apps to Kubernetes Using Helm - *Rockford Lhotka* / **CCT06** Fast Focus: Building & Debugging Container-based Apps with VS2019 - *Benjamin Day*
2:20 PM	2:45 PM	Networking Break • Visit the EXPO - *Pacifica 7*
2:45 PM	4:00 PM	**CCT07** Building a Better Distributed Job Scheduler on Kubernetes - *Chris Kinsman* / **CCT08** Lock the Doors, Secure the Valuables, and Set the Alarm - *Eric D. Boyd*
4:15 PM	5:30 PM	**CCT09** Microservices and Containers with Service Fabric and Azure Service Fabric Mesh - *Eric D. Boyd* / **CCT10** Practical Container Scenarios in Azure - *Anthony Nocentino*
5:30 PM	7:30 PM	Exhibitor Reception - *Pacifica 7*
START TIME	**END TIME**	**Cloud & Containers Live! Day 2: Wednesday, November 20, 2019**
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries
8:00 AM	9:15 AM	**CCW01** Azure App Service Security for ASP.NET Core MVC and WebAPI - *Benjamin Day* / **CCW02** To Be Announced
9:30 AM	10:45 AM	**CCW03** Microservice Architecture - *Rockford Lhotka* / **CCW04** Automate Your Life with PowerShell on Lambda - *AM Grobelny & Steve Roberts*
10:45 AM	11:30 AM	Networking Break • Visit the EXPO - *Pacifica 7*
11:30 AM	12:30 PM	**LIVE! 360 KEYNOTE: To Be Announced**
12:30 PM	1:30 PM	Birds-of-a-Feather Lunch
1:30 PM	2:00 PM	Dessert Break • Visit the EXPO
2:00 PM	3:15 PM	**CCW05** Architecting .NET Solutions in a Docker Ecosystem - *Alex Thissen* / **CCW06** To Be Announced
3:15 PM	4:00 PM	Networking Break • Visit the EXPO • Expo Raffle @ 3:30 p.m. - *Pacifica 7*
4:00 PM	5:15 PM	**CCW07** Serverless .NET on AWS - *AM Grobelny* / **CCW08** To Be Announced
7:30 PM	9:00 PM	Live! 360 Dessert Luau - *Wantilan Pavilion*
START TIME	**END TIME**	**Cloud & Containers Live! Day 3: Thursday, November 21, 2019**
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries
8:00 AM	9:15 AM	**CCH01** Busy Developer's Guide to the Clouds - *Ted Neward* / **CCH02** Kubernetes Runtime Security - *Jen Tong*
9:30 AM	10:45 AM	**CCH03** Standup Comedy Hour – Why App Folks Love Infrastructure Folks? - *Vishwas Lele & Jack O'Connell* / **CCH04** Implementing Zero Downtime Application Deployments on Azure PaaS - Marcel de Vries
11:00 AM	12:00 PM	**CLOUD & CONTAINERS LIVE! PANEL DISCUSSION: WHAT THE CLOUD-NATIVE EVOLUTION MEANS TO YOU** *Rockford Lhotka (moderator), Vishwas Lele, Marcel de Vries, Nick Pinheiro, & Jen Tong*
12:00 PM	1:00 PM	Lunch - *Oceana Ballroom*
1:00 PM	2:15 PM	**CCH05** Modernize Your App to be Delivered as a SaaS Service - *Nick Pinheiro* / **CCH06** Azure Cloud Transformation Done Right - *Marcel de Vries*
2:30 PM	3:45 PM	**CCH07** Enable External Access to Your Custom Apps with Azure AD B2B - *Nick Pinheiro* / **CCH08** Running 30-Year-Old Software as a Cloud Native SaaS Solution with Docker and Kubernetes on Azure - Roy Cornelissen
4:00 PM	5:00 PM	Next? Live! 360 Networking Event *Rockford Lhotka, Vishwas Lele, & Marcel de Vries*
START TIME	**END TIME**	**Cloud & Containers Live! Post-Conference Workshop: Friday, November 22, 2019**
7:30 AM	8:00 AM	Registration • Coffee and Morning Pastries
8:00 AM	5:00 PM	**CCF01** Workshop: Kubernetes on Azure - *Vishwas Lele*

Speakers and sessions subject to change

JAMES MCCAFFREY

The UCB1 Algorithm for Multi-Armed Bandit Problems

Imagine you're in a casino standing in front of three slot machines. You have 30 tokens. Each machine wins according to a different probability distribution, and these distributions are unknown to you. Your goal is to quickly find the best machine so you can maximize the amount of money you win. This is an example of a multi-armed bandit problem—named as such because slot machines are informally referred to as one-armed bandits.

In your day-to-day work environment, it's unlikely you'll have to deal with casino slot machines. But the multi-armed bandit scenario corresponds to many real-life problems. For example, a pharmaceutical company that has three new drugs for a medical condition has to find which drug is the most effective with a minimum number of clinical trials on human subjects. And an online advertising campaign with several new approaches needs to find which approach maximizes revenue as quickly as possible.

There are many different algorithms that can be used for a multi-armed bandit problem. The UCB1 (upper confidence bound, version 1) algorithm is one of the most mathematically sophisticated, but somewhat surprisingly, one of the easiest algorithms to implement. A good way to understand what the UCB1 algorithm is and to see where this article is headed is to take a look at a demo run in **Figure 1**.

The demo sets up three 0-based index machines, each with a different probability of winning. The three probabilities are (0.3, 0.7, 0.5), so machine [1] is the best machine. On each pull, if a machine wins it pays out $1 and if it loses it pays $0. The UBC1 algorithm starts by playing each machine once. In the demo, machines [0] and [1] won but machine [2] lost.

The UCB1 algorithm is iterative. The demo specifies six trials after the initialization pulls. In the first trial, the algorithm computes the average reward for each machine. Because machines [0] and [1] won in the initialization phase, their current average reward = $1.00 / 1 pull = $1.00. Because machine [2] lost, its current average reward = $0.00 / 1 pull = $0.00.

The current average rewards and the current trial number are used in an ingenious way to compute a decision value for each machine. For trial No. 1, the decision values are the same as the average rewards. The arm/machine to play is the one with the largest decision value. At this point machines [0] and [1] are tied with the largest decision value. Machine [0] is arbitrarily selected rather than machine [1]. Machine [0] is then played but loses.

Code download available at msdn.com/magazine/0819magcode.

In trial No. 2, the updated average reward for machine [0] is $1.00 / 2 pulls = $0.50. The average rewards for machines [1] and [2] are still $1.00 and $0.00, respectively, because neither machine was played. The decision values are computed as (1.33, 2.18, 1.18), so machine [1] is selected and it wins.

The process continues through trial No. 6. At that point, the UCB1 algorithm appears to be successful because machine [1], the best

```
C:\BanditUCB\bin\Debug\BanditUCB.exe

Begin UCB1 bandit demo

Three arms with true means u1 = 0.3, u2 = 0.7, u3 = 0.5
Playing each machine/arm once to start:
[0]: win
[1]: win
[2]: lose
-------------
trial #1
curr cum reward: 1.00 1.00 0.00
curr arm counts:    1    1    1
curr avg reward: 1.00 1.00 0.00
decision values: 1.00 1.00 0.00
Selected machine = [0]
result: a LOSS
-------------
trial #2
curr cum reward: 1.00 1.00 0.00
curr arm counts:    2    1    1
curr avg reward: 0.50 1.00 0.00
decision values: 1.33 2.18 1.18
Selected machine = [1]
result: a WIN
-------------
trial #3
curr cum reward: 1.00 2.00 0.00
curr arm counts:    2    2    1
curr avg reward: 0.50 1.00 0.00
decision values: 1.55 2.05 1.48
Selected machine = [1]
result: a WIN
-------------
trial #4
curr cum reward: 1.00 3.00 0.00
curr arm counts:    2    3    1
curr avg reward: 0.50 1.00 0.00
decision values: 1.68 1.96 1.67
Selected machine = [1]
result: a LOSS
-------------
trial #5
curr cum reward: 1.00 3.00 0.00
curr arm counts:    2    4    1
curr avg reward: 0.50 0.75 0.00
decision values: 1.77 1.65 1.79
Selected machine = [2]
result: a WIN
-------------
trial #6
curr cum reward: 1.00 3.00 1.00
curr arm counts:    2    4    2
curr avg reward: 0.50 0.75 0.50
decision values: 1.84 1.70 1.84
Selected machine = [0]
result: a LOSS
-------------

End bandit UCB1 demo
```

Figure 1 **UCB1 Algorithm Demo Run**

machine, has been played the most times (4) and has the highest average reward ($0.75).

The UCB1 algorithm is quite clever. Look at trial No. 5 in **Figure 1**. The cumulative rewards are ($1.00, $3.00, $0.00) and the numbers of times the machines have been played are (2, 4, 1). Therefore, machine [2] hasn't been tried since its initial loss in the initialization phase. An unsophisticated algorithm would continue by selecting machine [0] or [1], but UCB1 balances exploration of machine characteristics with exploitation of the best machine found and selects machine [2].

The UCB1 algorithm is designed specifically for bandit problems where the payout values are 0 or 1. This is called a Bernoulli process. UCB1 can be applied to other types of problems, such as one where the payout follows a Gaussian distribution. But the more different a payout distribution is from Bernoulli, the worse the UCB1 algorithm will perform. I don't recommend UCB1 for non-Bernoulli problems, but some of my colleagues believe that UCB1 can be successful if used conservatively.

> The key to the UCB1 algorithm is a function that converts a set of average rewards at trial t into a set of decision values, which are then used to determine which machine to play.

The information in this article is based on the 2002 research paper titled "Finite-Time Analysis of the Multiarmed Bandit Problem" by P. Auer, N. Cesa-Bianchi and P. Fischer. In addition to UCB1, the paper presents an algorithm named UCB-Normal intended for use with Gaussian distribution multi-armed bandit problems.

This article assumes you have intermediate or better programming skills with C# or a C-family language such as Python or Java, but doesn't assume you know anything about the UCB1 algorithm. The demo is coded using C#, but you shouldn't have any trouble refactoring the code to another language if you wish. The complete demo code is presented in this article. The source code is also available in the accompanying download.

Understanding the UCB1 Algorithm

The key to the UCB1 algorithm is a function that converts a set of average rewards at trial t into a set of decision values, which are then used to determine which machine to play. The equation is shown in **Figure 2**. In words, at trial t, select the arm, a, from all arms, that has the largest average reward (the r-hat) plus the upper confidence bound (which is the square root term). Here, n(a) is the number of times arm a has been pulled.

$$a(t) = argmax_{a \in A} \left\{ \hat{r}_a + \sqrt{\frac{2 * \ln(t)}{n_a}} \right\}$$

Figure 2 **The Key Equation for UCB1**

The equation is simpler than it appears and is best explained by example. Suppose, as in the demo, the algorithm is at trial t = 5, and the cumulative rewards are (1.00, 3.00, 0.00), and the arm counts are (2, 4, 1). The first step is to compute the average reward for each arm: (1.00 / 2, 3.00 / 4, 0.00 / 1) = (0.50, 0.75, 0.00). Then, the decision value for arm [0] is computed as:

```
decision[0] = 0.50 + sqrt( 2 * ln(5) / 2 )
            = 0.50 + sqrt(1.61)
            = 0.50 + 1.27
            = 1.77
```

Similarly, the decision values for arms [1] and [2] are:

```
decision[1] = 0.75 + sqrt( 2 * ln(5) / 4 )
            = 0.75 + sqrt(0.80)
            = 0.75 + 0.90
            = 1.65

decision[2] = 0.00 + sqrt( 2 * ln(5) / 1 )
            = 0.00 + sqrt(3.22)
            = 0.00 + 1.79
            = 1.79
```

Because the number of times an arm has been played is in the denominator of the fraction part of the upper confidence bound term, small values increase the decision value. This allows rarely pulled arms to eventually get a chance against arms with a high average reward. Very nice.

The Demo Program

The complete demo program, with a few minor edits to save space, is presented in **Figure 3**. To create the program, I launched Visual Studio and created a new console application named BanditUCB. I used Visual Studio 2017, but the demo has no significant .NET Framework dependencies.

After the template code loaded, at the top of the editor window I removed all unneeded namespace references, leaving just the reference to the top-level System namespace. In the Solution Explorer window, I right-clicked on file Program.cs, renamed it to the more descriptive BanditProgram.cs and allowed Visual Studio to automatically rename class Program.

All normal error checking has been omitted to keep the main ideas as clear as possible. All the control logic is contained in the Main method. There's a single helper function named ArgMax that returns the index of the largest value in a numeric array. For example, if an array holds values (5.0, 7.0, 2.0, 9.0), then ArgMax returns 3.

Setting Up UCB1

The demo program begins with these statements:

```
Random rnd = new Random(20);
int N = 3;
int trials = 6;
double p = 0.0;
double[] means = new double[] { 0.3, 0.7, 0.5 };
```

The Random object is used to determine whether a selected machine wins or loses. The seed value, 20, is used only because it gives a representative demo. The array named means could have

Visual Studio LIVE!
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

Sept 29 - Oct 3, 2019 | Westin Gaslamp Quarter
EXPERIENCE TECH TRAINING IN THE WARM CALIFORNIA SUN

INTENSE DEVELOPER TRAINING CONFERENCE

In-depth Technical Content On:

- AI, Data and Machine Learning
- Cloud, Containers and Microservices
- Delivery and Deployment
- Developing New Experiences
- DevOps in the Spotlight
- Full Stack Web Development
- .NET Core and More

New This Year!

On-Demand Session Recordings Now Available

Get on-demand access for one full year to all keynotes and sessions from Visual Studio Live! San Diego, including everything Tuesday – Thursday at the conference.

ondemand.vslive.com

Save $300 When You Register by August 30

Your Adventure Starts Here!

vslive.com/sandiego

SUPPORTED BY

Microsoft | Visual Studio | msdn magazine | Visual Studio MAGAZINE

PRODUCED BY

CONVERGE 360 An 1105 MEDIA company

AGENDA AT-A-GLANCE

#VSLive

DevOps in the Spotlight | Cloud, Containers and Microservices | AI, Data and Machine Learning | Developing New Experiences | Delivery and Deployment | .NET Core and More | Full Stack Web Development

START TIME	END TIME				
		Pre-Conference Full Day Hands-On Labs: Sunday, September 29, 2019 *(Separate entry fee required)*			
7:30 AM	9:00 AM	Pre-Conference Hands-On Lab Registration - Coffee and Morning Pastries			
9:00 AM	6:00 PM	**HOL01** Full Day Hands-On Lab: Develop an ASP.NET Core 2 App with EF Core 2 in a Day - *Philip Japikse*		**HOL02** Day Hands-On Lab: Building a Modern DevOps Pipeline on Microsoft Azure with ASP.NET Core and Azure DevOps - *Brian Randell & Mickey Gousset*	
START TIME	**END TIME**	**Pre-Conference Workshops: Monday, September 30, 2019** *(Separate entry fee required)*			
7:30 AM	8:00 AM	Pre-Conference Workshop Registration - Coffee and Morning Pastries			
8:00 AM	5:00 PM	**M01** Workshop: Modern Security Architecture for ASP.NET Core - *Brock Allen*	**M02** Workshop: Kubernetes on Azure - *Vishwas Lele*	**M03** Workshop: Cross-Platform C# Using .NET Core, Kubernetes, and WebAssembly - *Rockford Lhotka & Jason Bock*	
6:45 PM	9:00 PM	Dine-A-Round			
START TIME	**END TIME**	**Day 1: Tuesday, October 1, 2019**			
7:00 AM	8:00 AM	Registration - Coffee and Morning Pastries			
8:00 AM	9:15 AM	**T01** MVVM and ASP.NET Core Razor Pages - *Ben Hoelting*	**T02** Azure, Windows and Xamarin: Using the Cloud to Power Your Cross-platform Applications - *Laurent Bugnion*	**T03** .NET Core and Azure, Microservices to Messaging - *Brady Gaster*	**T04** What's New in C# 7.X And C# 8 - *Philip Japikse*
9:30 AM	10:45 AM	**T05** Angular 101 - *Deborah Kurata*	**T06** Securing Web APIs from Mobile and Native Applications - *Brock Allen*	**T07** SQL Server 2019 Deep Dive - *Scott Klein*	**T08** How Microsoft Does DevOps - *Mickey Gousset*
11:00 AM	12:00 PM	**Keynote:** AI for the Rest of Us - *Damian Brady, Cloud Developer Advocate, Microsoft*			
12:00 PM	1:00 PM	Lunch			
1:00 PM	1:30 PM	Dessert Break - Visit Exhibitors			
1:30 PM	2:45 PM	**T09** Securing Single Page Applications (SPAs) - *Ben Hoelting*	**T10** To Be Announced	**T11** Data Pipelines with End-2-End Azure Analytics - *Scott Klein*	**T12** .NET Standard, .NET Core, why and how? - *Laurent Bugnion*
3:00 PM	4:15 PM	**T13** Managing Your Angular Async Data Effectively - *Deborah Kurata*	**T14** Cross-Platform Development with Xamarin, C#, and CSLA .NET - *Rockford Lhotka*	**T15** What Every Developer Ought to Know About #deeplearning and #neuralnetwork - *Vishwas Lele*	**T16** Azure Pipelines - *Brian Randell*
4:15 PM	5:30 PM	Welcome Reception			
START TIME	**END TIME**	**Day 2: Wednesday, October 2, 2019**			
7:30 AM	8:00 AM	Registration - Coffee and Morning Pastries			
8:00 AM	9:15 AM	**W01** Diving Deep Into ASP.NET Core 2.x - *Philip Japikse*	**W02** Building End-to-End ML Solution Using Azure Machine Learning Service - *Raj Krishnan*	**W03** Go Serverless with Azure Functions - *Eric Boyd*	**W04** Exceptional Development: Dealing With Exceptions in .NET - *Jason Bock*
9:30 AM	10:45 AM	**W05** WebAssembly: the Browser is your OS - *Jeremy Likness*	**W06** Azure Data Explorer—An in-depth Look at the New Microsoft PaaS Offering - *Raj Krishnan*	**W07** Keep Secrets with Azure Key Vault - *Eric D. Boyd*	**W08** Testability in .NET - *Jason Bock*
11:00 AM	12:00 PM	**General Session:** Moving .NET Beyond Windows = James Montemagno, Principal Program Manager - *Mobile Developer Tools, Microsoft*			
12:00 PM	1:00 PM	Birds-of-a-Feather Lunch			
1:00 PM	1:30 PM	Dessert Break - Visit Exhibitors - Exhibitor Raffle @ 1:15pm (Must be present to win)			
1:30 PM	1:50 PM	**W09** Fast Focus: Getting Started with ASP.NET Core 2.0 Razor Pages - *Walt Ritscher*	**W10** Fast Focus: The Four Flavors of Power BI - *Thomas LeBlanc*	**W11** Fast Focus: Serverless of Azure 101 - *Eric D. Boyd*	
2:00 PM	2:20 PM	**W12** Fast Focus: Sharing C# Code Across Platforms - *Rockford Lhotka*	**W13** Fast Focus: Running .NET on Linux: What's Different? - *Steve Roberts*	**W14** Fast Focus: Lessons about DevOps from 3D Printing - *Colin Dembovsky*	
2:30 PM	3:45 PM	**W15** Angular vs React - a Comparison - *Gregor Dzierzon*	**W16** Effective Visualizations with Power BI - *Thomas LeBlanc*	**W17** Serverless .NET on AWS - *Steve Roberts*	**W18** Better Azure DevOps - Security 101 - *Brian Randell*
4:00 PM	5:15 PM	**W19** Object Oriented Programming Using TypeScript - *Gregor Dzierzon*	**W20** Transition from SSIS to Azure Data Factory - *Thomas LeBlanc*	**W21** Cloud Debugging – A Revolutionary Approach - *Alon Fliess*	**W22** DevOps for Machine Learning - *Damian Brady*
7:00 PM	8:30 PM	VSLive!'s City Lights at Night Trolley Tour			
START TIME	**END TIME**	**Day 3: Thursday, October 3, 2019**			
7:30 AM	8:00 AM	Registration - Coffee and Morning Pastries			
8:00 AM	9:15 AM	**TH01** Advanced Serverless Workflows with Durable Functions - *Jeremy Likness*	**TH02** Stunning Mobile Apps with the Xamarin Visual Design System - *James Montemagno*	**TH03** Building a Stronger Team, One Strength at a Time - *Angela Dugan*	**TH04** Creating Business Applications Using Blazor (Razor Components) - *Michael Washington*
9:30 AM	10:45 AM	**TH05** What's New in Bootstrap 4 - *Paul Sheriff*	**TH06** How to Have Better Business Intelligence through Visualizations - *Walt Ritscher*	**TH07** How Do You Measure Up? Collect the Right Metrics for the Right Reasons - *Angela Dugan*	**TH08** Building Business Applications Using Bots - *Michael Washington*
11:00 AM	12:15 PM	**TH09** Signal R: Real-time for All Things - *Brady Gaster*	**TH10** Microsoft Power Platform, RAD Done Right: Building Dynamic Mobile Apps with PowerApps - *Walt Ritscher*	**TH11** Past, Present & Future of C# Debugging - *Alon Fliess*	**TH12** Automate Your Life with PowerShell on Lambda - *Steve Roberts*
12:15 PM	1:30 PM	Lunch			
1:30 PM	2:45 PM	**TH13** Advanced Fiddler Techniques - *Robert Boedigheimer*	**TH14** Building UWP Apps for Multiple Devices - *Tony Champion*	**TH15** To Microservice or Not to Microservice? How? - *Alon Fliess*	**TH16** Testing in Production Using Azure and Visual Studio Team Services (VSTS) - *Colin Dembovsky*
3:00 PM	4:15 PM	**TH17** Improving Web Performance - *Robert Boedigheimer*	**TH18** Building Cross Device Experiences with Project Rome - *Tony Champion*	**TH19** Getting Started with Unit Testing in Visual Studio - *Paul Sheriff*	**TH20** Modernizing Your Source Control: Migrating to Git from Team Foundation Version Control (TFVC) - *Colin Dembovsky*

Speakers and sessions subject to change

CONNECT WITH US

twitter.com/vslive – @VSLive

facebook.com – Search "VSLive"

linkedin.com – Join the "Visual Studio Live" group!

been named something like probsWin instead. But because each machine pays either $1 or $0, the average (mean) value for each machine is the same as the probability of winning. For example, if the probability of winning for a machine is 0.6 and you play the machine 1,000 times, you'd expect to win $1 about 600 times. The mean value per pull is $600 / 1000 = 0.60.

Computing the Decision Values

The demo program computes decision values using a direct mapping to the UCB1 equation in **Figure 2**:

```
for (int i = 0; i < N; ++i) {
  decValues[i] = avgReward[i] +
    Math.Sqrt( (2.0 * Math.Log(t) / armCounts[i]) );
  Console.Write(decValues[i] + " ");
}
```

The computation will throw an exception if t = 0 (the log of 0 is negative infinity) or if any arm count is 0 (division by 0). However, the UCB1 initialization phase where each machine is tried once prevents either of the exception conditions from occurring.

After the decision values have been computed, the machine to play is determined by this statement:

```
int selected = ArgMax(decValues);
Console.WriteLine("Selected machine = [" + selected + "]");
```

The ArgMax function returns the index of the largest decision value. If two or more decision values are tied as largest, ArgMax returns the first index encountered. This introduces a slight bias toward smaller-indexed machines. One approach to eliminate this bias would be to refactor ArgMax so that if there's a tie, one of the tied indexes is chosen randomly.

The Epsilon-Greedy Algorithm

The UCB1 algorithm is closely related to another multi-armed bandit algorithm called epsilon-greedy. The epsilon-greedy algorithm begins by specifying a small value for epsilon. Then at each trial, a random probability value between 0.0 and 1.0 is generated. If the generated probability is less than (1 - epsilon), the arm with the current largest average reward is selected. Otherwise, an arm is selected at random. An epsilon-greedy implementation based on the demo program structure could look like:

```
//int selected = ArgMax(decValues);  // UCB1
double epsilon = 0.05;  // Epsilon-greedy
int selected;
double pr = rnd.NextDouble();  // [0.0, 1.0)
if (pr < (1 - epsilon))
  selected = ArgMax(avgReward);  // Usually
else
  selected = rnd.Next(0, N);  // 5% of the time
```

Figure 3 **The UCB1 Algorithm Demo Program**

```
using System;
namespace BanditUCB
{
  class BanditProgram
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Begin UCB1 bandit demo \n");
      Console.WriteLine("Three arms with true means u1 = " +
       "0.3, u2 = 0.7, u3 = 0.5");
      Random rnd = new Random(20);
      int N = 3;
      int trials = 6;
      double p = 0.0;

      double[] means = new double[] { 0.3, 0.7, 0.5 };
      double[] cumReward = new double[N];
      int[] armCounts = new int[N];
      double[] avgReward = new double[N];
      double[] decValues = new double[N];

      // Play each arm once to get started
      Console.WriteLine("Playing each arm once: ");
      for (int i = 0; i < N; ++i) {
        Console.Write("[" + i + "]: ");
        p = rnd.NextDouble();
        if (p < means[i]) {
          Console.WriteLine("win");
          cumReward[i] += 1.0;
        }
        else {
          Console.WriteLine("lose");
          cumReward[i] += 0.0;
        }
        ++armCounts[i];
      }

      Console.WriteLine("-------------");
      for (int t = 1; t <= trials; ++t) {
        Console.WriteLine("trial #" + t);

        Console.Write("curr cum reward: ");
        for (int i = 0; i < N; ++i)
          Console.Write(cumReward[i].ToString("F2") + " ");

        Console.Write("\ncurr arm counts: ");
        for (int i = 0; i < N; ++i)
          Console.Write(armCounts[i] + " ");

        Console.Write("\ncurr avg reward: ");
        for (int i = 0; i < N; ++i) {
          avgReward[i] = (cumReward[i] * 1.0) / armCounts[i];
          Console.Write(avgReward[i].ToString("F2") + " ");
        }

        Console.Write("\ndecision values: ");
        for (int i = 0; i < N; ++i) {
          decValues[i] = avgReward[i] +
            Math.Sqrt( (2.0 * Math.Log(t) / armCounts[i]) );
          Console.Write(decValues[i].ToString("F2") + " ");
        }

        int selected = ArgMax(decValues);
        Console.WriteLine("\nSelected machine = [" +
          selected + "]");
        p = rnd.NextDouble();
        if (p < means[selected]) {
          cumReward[selected] += 1.0;
          Console.WriteLine("result: a WIN");
        }
        else {
          cumReward[selected] += 0.0;
          Console.WriteLine("result: a LOSS");
        }
        ++armCounts[selected];
        Console.WriteLine("-------------");
      } // t

      Console.WriteLine("End bandit UCB1 demo ");
      Console.ReadLine();
    } // Main

    static int ArgMax(double[] vector)
    {
      double maxVal = vector[0];
      int result = 0;
      for (int i = 0; i < vector.Length; ++i) {
        if (vector[i] > maxVal) {
          maxVal = vector[i];
          result = i;
        }
      }
      return result;
    }
  } // Program
} // ns
```

One of several variations of the basic epsilon-greedy algorithm is to slowly decrease the value of epsilon over time. This has the effect of concentrating on exploration early in the run and then emphasizing exploitation of the best arm found, later in the run. The biggest problem with epsilon-greedy is that there's no easy way to determine a good value for epsilon.

In my opinion, the research results comparing UCB1 and epsilon-greedy and many other multi-armed bandit algorithms are inconclusive. Based on my experience, there's no single, consistently best algorithm to use and, if possible, it's good practice to run a few experiments with different algorithms using a simulation of your real problem.

In my opinion, the research results comparing UCB1 and epsilon-greedy and many other multi-armed bandit algorithms are inconclusive.

The standard way to compare different multi-armed bandit algorithms is to compute a regret metric. Regret is the difference between the expected value of the system, assuming you know the best arm, and the actual value of the system in experiments. For example, suppose you played the three machines of the demo system 10 times and you won six times and lost four times. The total reward is $6.00. But if you hypothetically pulled the best arm (with probability of winning equal to 0.7) all 10 times, the average total reward would be $7.00. Therefore, the regret is $7.00 - $6.00 = $1.00.

Wrapping Up

I mentally organize machine learning into three categories: supervised learning, where you have training data with known, correct answers; unsupervised learning, where you have data without correct answers; and reinforcement learning (RL), where a correct or incorrect result is called a reward (which could be negative) and comes from a function instead of data. Multi-armed bandit problems are usually considered a part of RL, but some of my research colleagues consider the multi-armed bandit a distinct type of problem.

There are dozens of algorithms for multi-armed bandit scenarios. Based on my experience, in addition to the UCB1 and epsilon-greedy algorithms described in this article, the most common algorithm used in practice is called Thompson Sampling. You can learn about that algorithm in the February 2018 issue of *MSDN Magazine* at msdn.com/magazine/mt829274. ■

Dr. James McCaffrey *works for Microsoft Research in Redmond, Wash. He has worked on several key Microsoft products including Azure and Bing. Dr. McCaffrey can be reached at jamccaff@microsoft.com.*

Thanks to the following Microsoft technical experts who reviewed this article: *Chris Lee, Ricky Loynd*

dtSearch®

Instantly Search Terabytes

dtSearch's **document filters** support:
- popular file types
- emails with multilevel attachments
- a wide variety of databases
- web data

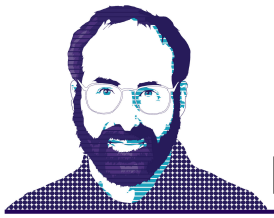
Over 25 search options including:
- efficient multithreaded search
- easy multicolor hit-highlighting
- forensics options like credit card search

Developers:
- SDKs for Windows, Linux, macOS
- Cross-platform APIs for C++, Java and .NET with .NET Standard / .NET Core
- FAQs on faceted search, granular data classification, Azure and more

Visit dtSearch.com for
- hundreds of reviews and case studies
- fully-functional enterprise and developer evaluations

The Smart Choice for Text Retrieval® since 1991

dtSearch.com 1-800-IT-FINDS



DAVID PLATT

Change of Plan

Pigs are flying. Hell is freezing over. And Microsoft is telling users they don't need to periodically change their passwords anymore.

That sounds surprising, considering all the harangues we've received over the years about diligently performing password changes. But it's true. Aaron Margosis described on the Microsoft Security Guidance blog (bit.ly/2Lq4jUB) how the security baseline for Windows 10 and Windows Server is being changed, stating that Microsoft is "Dropping the password-expiration policies that require periodic password changes." Holy Toledo.

Margosis explains: "When humans are forced to change their passwords, too often they'll make a small and predictable alteration to their existing passwords, and/or forget their new passwords." He continues with a thoughtful discussion of the costs of periodic password expirations versus the marginal security gain from them. I especially admire how Margosis puts himself in the shoes of his human users, instead of hoping, futilely, that they'll morph into something more logical. I urge you to read the entire piece.

He's wrong about one thing, though. Users don't use that shortcut "too often." Every single person I've ever asked admits to doing it every single time. That's what happens when you exceed a user's "hassle budget," which I discussed in my April 2013 column (msdn.com/magazine/dn166939.aspx).

I especially admire how Margosis puts himself in the shoes of his human users, instead of hoping, futilely, that they'll morph into something more logical.

It's about time we got rid of periodic resets. They're not benign. For a brutal description of the deadly consequences of too-frequent password resets, read the last chapter of "Do No Harm: Stories of Life, Death and Brain Surgery," by English neurosurgeon Henry Marsh (St. Martin's Press, 2014). It should be required reading for all security architects.

I'm glad to see Microsoft implementing ideas from what I've named the Rational Security Movement. I've been advocating this since at least 2003, in my newsletter at bit.ly/2xiK6I7 (which also announced the birth of my daughter Lucy, who will probably have her driver's license by the time you read this). Bruce Schneier wrote about balancing effort versus return in his excellent book, "Beyond Fear: Thinking Sensibly About Security in an Uncertain World" (Copernicus Books, 2003). And Cormac Herley, of Microsoft Research, examined how users make those tradeoffs in a superb paper entitled, "So Long, and No Thanks for the Externalities: The Rational Rejection of Security Advice by Users" (bit.ly/2LzdySL). In it, Herley decries "the profound irony that much security advice, not only does more harm than good (and hence is rejected), but does more harm than the attacks it seeks to prevent, and fails to do so only because users ignore it."

These changes happen slowly, though. I attended a Harvard lecture given by Lorrie Cranor, a professor at Carnegie Mellon who studies the uneasy coexistence of security and usability (check out the video at bit.ly/2xfmV1t). She was then a visiting scholar at the U.S. Department of Commerce, at which she had to use six separate systems, with separate login credentials. She managed to convince the administrators of two of those systems that password resets were not cost-effective, but despite her research on the topic (see bit.ly/2RFKz0v), she couldn't convince the others. She didn't tell us, but I'll bet you anything that she just bumped up the last digit of her expired password on those systems, like everyone else in the universe.

I'm wondering when and if this change will work its way into Microsoft's internal systems. My friends there say that so far it hasn't. But interestingly, most of them said, "I hardly ever use my password to log in anymore. I just use the camera on my laptop." By this they mean Windows Hello, which authenticates users via facial recognition. As I've always said: Users are lazy, a natural and inevitable consequence of being human. (See my very first column, msdn.com/magazine/ee309884.) If you make something easier to do, people will jump on it. (In their haste, surrendering their biometric data without worrying about privacy, but that's a topic for another day.)

I'm as human as anyone else. I'm going to make sure my next laptop has a camera that supports it, so I can stop using passwords, too. ■

DAVID S. PLATT *teaches programming .NET at Harvard University Extension School and at companies all over the world. He's the author of 11 programming books, including "Why Software Sucks" (Addison-Wesley Professional, 2006) and "Introducing Microsoft .NET" (Microsoft Press, 2002). Microsoft named him a Software Legend in 2002. He wonders whether he should have taped down two of his daughter's fingers so she would learn how to count in octal. You can contact him at rollthunder.com.*

Address the ELEPHANT IN THE ROOM

Bad address data costs you money, customers and insight.

Melissa's 30+ years of domain experience in address management, patented fuzzy matching and multi-sourced reference datasets power the global data quality tools you need to keep addresses clean, correct and current. The result? Trusted information that improves customer communication, fraud prevention, predictive analytics, and the bottom line.

- Global Address Verification
- Digital Identity Verification
- Email & Phone Verification
- Location Intelligence
- Single Customer View

+ .NET
+ Microsoft® SSIS
+ Microsoft® Dynamics CRM

See the Elephant in Your Business -

Name it and Tame it!

melissa

www.Melissa.com | 1-800-MELISSA

Free API Trials, Data Quality Audit & Professional Services.

Progress® Telerik®

Modern UI Made Easy

Building a modern UI for Web, Desktop and Mobile apps has never been easier with our .NET, JavaScript & Productivity Tools

www.telerik.com/msdn