A Broadcom Company
technologies

CA Gen™

The CA Gen DevOps Story

White Paper
Building a z/OS CICS Blockmode Application with CA Endevor® and Zowe

Table of Contents

Chapter 1: Introduction

In many companies today, there has been a push to embrace more modern DevOps processes for their applications, including on z/OS platforms. Broadcom® is already a major provider of Mainframe DevOps products that enable our customers to solve their complex operational needs with robust solutions. CA Gen™ previously sat outside of the traditional DevOps pipeline while still providing a robust and integrated solution for designing and constructing your most critical business applications. While such a comprehensive approach does have many advantages, it can sometimes be too restrictive to satisfy an organization's operational requirements.

Have you ever had to explain to your organization why it is challenging to integrate CA Gen with a standard DevOps tool? If so, we aim to provide a template that shows how to integrate a CA Gen CICS block-mode application (now referred to as *reference application type*) with the most popular mainframe DevOps tool around—CA Endevor®.

It is worth mentioning that DevOps is a very broad topic, so there are many valid variations to any approach. Some of the items covered in this paper will be applicable to other paths. In the spirit of DevOps, feel free to experiment to make it work for you. Broadcom is trying to add tools for your DevOps tool belt when working with CA Gen.

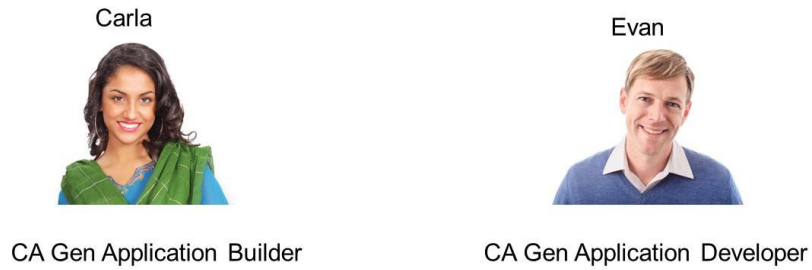
So, what value can be provided by building your CA Gen applications with CA Endevor? Both Host Construction and the Implementation Toolset perform similar functions. Utilizing CA Endevor to build applications provides several unique benefits and extension points that are not available with Host Construction or the Implementation Toolset:

- Track and version source code artifacts
- Uniform skills leveraged with a common software change management (SCM) tool
- Automate deployment on promotion
- Custom scripting with the Zowe CLI
- Integrate with Jenkins or other CI/CD tools
- Build applications without interacting with the "green screen"

1.1 Design Thinking

When designing solutions, Broadcom teams want to understand the users—what their everyday challenges are, their expectations, frustrations, and desires. Looking at the individual profiles of people working in Broadcom customers' organizations and observing patterns has enabled the creation and definition of specific personas.

Figure 1: CA Gen Personas



1.1.1 Carla – CA Gen Application Builder

Carla is an expert in CA Gen model management and has been building CA Gen applications for many years. She is experienced with ISPF and CA Gen's green-screen applications, like the Host Encyclopedia and Implementation Toolset. While she can develop CA Gen applications, it is not her day-to-day responsibility and is not something she focuses on now.

Carla's daily responsibilities require that she work with many different CA Gen developers to ensure their changes are properly scoped and to ensure the proper subsets are assigned to those developers. Once CA Gen developers are ready to check in their changes, Carla takes over. This involves managing changes from multiple developers and building applications from the CA Gen models to ensure the changes work correctly together. Changes may involve user exits or other external resources that need to be considered alongside the normal CA Gen construction activities.

Carla is responsible for keeping the encyclopedia in a consistent state, so it's up to her to identify when something isn't working correctly. If something isn't working like it should, she must track down the problem and identify who was responsible for creating the error. To accomplish this, she must run tests against the CA Gen application, which were created using a homegrown testing framework.

Once the application is building correctly, it's Carla's job to post build results to any external parties and start the process of moving the model changes from the development model into a QA model. She must build the application for the QA environment once the changes to the model have been migrated so that more rigorous QA testing can occur.

Finally, when QA testing is completed, she must migrate the changes to the production model and build the application again so it can be deployed into production. The deployment of the application into product is a more controlled process, but Carla is still responsible for building the load modules that run in production.

1.1.2 Evan – CA Gen Application Developer

As a CA Gen application developer, Evan is responsible for making changes to CA Gen models and creating new applications using existing models. He has extensive experience with CA Gen and is an expert in modeling systems to meet business requirements. He can write code to support his CA Gen applications but doesn't normally write full applications.

Evan doesn't interact with the encyclopedia and so doesn't have a lot of familiarity with z/OS and the green screen tools that CA Gen provides. Evan does use a 3270 emulator to test his application changes but is not a z/OS or CICS expert. This means that he is reliant on Carla to build and deploy his applications on z/OS.

Evan's day-to-day work requires him to use the Toolset UI to make changes to models and create new CA Gen applications, as business requirements dictate. This is done with an agile or scrum methodology, where he works with product management to meet the needs of the end users. To complete his work, Evan interacts with Carla regularly to scope and check out a subset of his model so he can isolate his changes. Testing his changes on z/OS requires him to wait for his changes to be built and deployed by Carla, which sometimes means waiting for other developers' changes to be integrated as well.

Ideally, Evan would like to automate some of the repetitive tasks that he must do each day. He would get more done, in less time, and with fewer mistakes if he was able to test his applications himself. Avoiding interacting with z/OS is critical in meeting these goals. Because he's not an experienced mainframer, he'd also need fewer cheat sheets with infrequently used z/OS commands as reference.

Chapter 2: Welcome to DevOps CA Gen Edition

2.1 What is DevOps?

What does DevOps actually mean? It is definitely a buzzword, and as such, can take on many different meanings depending on whom you talk to. For the purposes of this document, DevOps refers to a cycle of streamlined steps that are repeatable and designed to increase the efficiency of and collaboration between traditionally siloed development and operations teams.

Generally, when an organization adopts a DevOps process, there is an effort to automate as much of the process as possible. In addition to adding automation, having a simple process to follow at each step of the DevOps cycle makes it easier to train new employees and to share responsibilities between team members. This new CA Gen with CA Endevor DevOps process aims to show a path to realize both of those goals.

CA Gen plans to provide a template that enables building applications using CA Endevor. With some customization, it is possible to integrate with CA Endevor to build CA Gen applications. It is possible to use similar techniques with other SCMs but because CA Endevor is a highly configurable system, providing a template that works for everyone isn't feasible. The templates and instructions apply only to CA Endevor.

Also, depending on input from customers, the CA Gen team will be happy to explore more of these DevOps templates for other supported technologies, platforms, and application types.

There are many advantages to using CA Endevor to build applications. It makes it easier to integrate with existing DevOps practices. Instead of CA Gen being an exception to the rule, CA Gen applications can be built like any other application.

Through the utilization of CA Endevor sandbox environments, developer changes are isolated. This is an important first step for following more agile development practices that are becoming more commonplace throughout the industry.

2.2 Traditional CA Gen Workflow

CA Gen developer teams implement changes to CA Gen generated code by modifying CA Gen models and regenerating and rebuilding the code from them. The lifecycle of a change consists of three versions of the same model, each used at a specific stage—development, QA, and production. This model ensures that changes do not impact production without rigorous testing.

A coding change starts with copying a production model to create its development model version. This development model is further divided into subsets, each of which is checked out from the Encyclopedia repository and given to an application developer. The checkout process brings the subset to the CA Gen Toolset where the developer makes the actual change.

After a modification is complete, you can build and implement the changed code in one of two ways:

- Check-in the subset into the original development model and generate it from the repository. Either cross-generate COBOL code from a CSE/HE or generate COBOL code using Host Construction.
- Cross-generate the COBOL code from the Toolset.

COBOL code that is cross- generated is then transferred as a remote file to z/OS and installed by the z/OS Implementation Toolset. COBOL code that is generated by Host Construction can be installed by the Host Construction. Installing generated code consists of compiling, link-editing, and DB2 Bind (if appropriate) to create application load modules. You can then deploy and test application load modules on target platforms. These tests often include changes from multiple subsets.

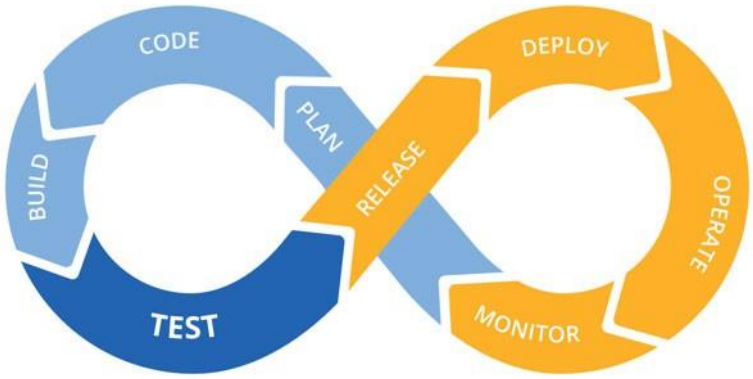
When the testing is successful, you migrate the changes in the development model to the QA model where the building and testing process is repeated. The change eventually migrates into the production model, where the process is repeated one more time, enabling the change to be deployed to the production environment.

Throughout this process, you must realize that the CA Gen model is the artifact that is kept and that the source code is treated only as a means to transform the model into a working application.

2.3 DevOps Workflow

As mentioned before, DevOps is a blanket term and there are many valid ways to structure a DevOps pipeline. However, there are some commonalities to the DevOps lifecycle. The following graphic shows a typical depiction of the DevOps lifecycle in an application. An important feature of this graphic is that the process is iterative, so once you reach the “end” of the cycle, it starts all over again.

Figure 2: DevOps Lifecycle in an Application



One of the key benefits of CA Endevor-built CA Gen applications is automating the compiling, linking, and possibly even deployment of the application with CA Endevor processors. Today, CA Gen customers can organize their development practices to follow a model like this, and some have done that on their own. Adding the CA Endevor into the development process will automate and standardize building and releasing applications. In the following sections we’ll go through the DevOps process stage by stage to explain how everything fits together.

2.3.1 Prerequisites

Consider the following:

- **Zowe and CA Brightside**

  The new DevOps workflow makes heavy use of the new open-sourced Zowe framework to provide automation for many operations. In a nutshell, Zowe offers modern ways to interact with z/OS much like a cloud platform. CA Brightside provides a Zowe environment that is fully supported by Broadcom. The installation process for Zowe and various plugins for other Broadcom products are streamlined when using CA Brightside. If you would just like to get started with open-source Zowe, the details on how to install it are available on the Zowe homepage at https://zowe.org.

  Zowe CLI is required to make full use of the process automation, however, it is possible to make use of the CA Endevor approach without using Zowe. The difference is that manual steps are required instead of being able to run simple scripts to automate things like uploading newly generated COBOL files. For the purpose of this document, though, the following steps assume that Zowe and the CA Endevor plugin for Zowe are configured and ready to use.

- **CA Endevor Configuration**

  The new DevOps workflow relies on CA Endevor processors to compile, link-edit, and deploy the CA Gen COBOL generated application.

  Complete these tasks by modifying an existing CA Endevor environment and processors or creating new ones. Note that the processors are very specific to CA Gen and the build environment, therefore, we don't provide an example SCL.

  These are the necessary items to continue:

  - A processor to invoke the COBOL compiler and use the embedded co-processor
  - A processor to invoke the linkage editor to create and bind the load module

  A template is provided to specify the COBOL SQL statements required by the co-processor. More templates are provided for LNKINC and SIDEDECK for different application types as built by CA Gen. These templates are used by the sample code, described in the next section of this paper.

  An additional, but optional, processor is provided to enable deployment of applications. This processor uses a template to update the CICS DFHCSD by adding definitions for transactions and programs and copies the programs to a data set that can be added to the CICS DFHRPL concatenation. It may be necessary to issue a NEWCOPY or PHASEIN command in order to utilize those new load modules.

NOTE: This processor is not technically necessary because those things can be done by hand, but they can save developer time by automating manual steps.

- **Sample Code**

  A collection of scripts created to process a CA Gen remote file for the reference application type. The scripts rely on CA Endevor and the CA Endevor plugin for Zowe, which must be configured as explained in the previous sections.

  The sample code must be provided with a valid CA Gen remote file generated from either the Toolset or one of the CA Gen Encyclopedias. The system where the scripts are run must have the Zowe CLI installed and configured. In addition, the CA Endevor SCM Plugin for Zowe must be installed and configured. There are also a number of configuration parameters to customize to your environment. Finally, once all the prerequisite tools are installed and everything is configured properly, the processing runs without requiring any user input. We have created a collection of scripts in support of the background work for this paper and are working through the process of making them available.

2.3.2 Plan

The DevOps process starts with a requirement for a change. This change can be something as small as adding a new field to an existing entity or something a bit larger like a new screen. The process of developing, testing, and deploying an application should be the same regardless of the scope of the work. But, the first step in the process is determining the scope of the work. The CA Gen subset feature is used by many teams to ensure that the piece of the model that is being worked on, and subsequently locked, is appropriately sized.

This is a job that Carla will be responsible for and she will need to work with Evan to identify the proper scope for the change being implemented. Gathering the initial requirement and identifying the proper scope for the subset of the model fits nicely into the planning stage of our DevOps cycle.

2.3.3 Code

After Evan has his subset scoped, he can proceed to download the subset. Historically, downloading a model or a subset required him to use FTP or CA Gen's seamless upload/download functionality of the Windows Toolset and Host Encyclopedia. However, Evan can quickly accomplish the download by using the Zowe files API.

Once the subset or model has been transferred to his workstation, he can load it into the Toolset for editing. Evan can make any edits that are necessary to complete his work. The actual process of editing the model does not change with the new DevOps process.

Once Evan's changes are complete, he generates the application source code using the Toolset generators. Code generation is only required where model changes were made. So, if a procedure step or action block were modified, they must be generated. Also, if any new model objects were added, they must be generated as well. Evan must also be sure to choose the Install option when generating so a CA Gen remote file is created.

2.3.4 Build

The build step for the reference application normally leverages either Host Construction or the z/OS Implementation Toolset. Those are traditional green-screen applications and must be run on the mainframe through a 3270 emulator. The sample code allows Evan to have access to similar functionality using a combination of CA Endevor SCM, Zowe CLI, and custom processing from CA Gen to build the reference application.

A key benefit of the sample code is that it can run on multiple platforms. As explained before, sample code scripts create (or update) the necessary CA Endevor elements to build the application. Evan can avoid having to open up a 3270 terminal and can just build from source code he generates on his Workstation. While it is not required, we recommend using developer sandboxes in CA Endevor to isolate source code changes.

As the model is the source of truth for any CA Gen application, generated code is an intermediate artifact for building applications there is no built-in functionality to preserve application source code. However, source code can be an important artifact for certain organizations, especially for auditing purposes.

Utilizing CA Endevor for building applications comes with the ancillary benefit that the source code CA Gen generates is managed by CA Endevor and is version controlled. Now the CA Gen application looks like any other COBOL application. Carla can easily satisfy any auditing requirements that require the source code to be available for any application running in production.

There is another option for running the CA Endevor scripts off of a personal workstation. Jenkins is a leading CI/CD server and can execute the sample code on multiple platforms, further automating the build process. Instead of using the sample code on a local workstation, a Jenkins job can invoke the same processing in the CI/CD environment. It's worth noting that Jenkins is only one option for a CI/CD server and almost any CI/CD tool will be able to run the sample code. No matter the tool, the opportunity to easily integrate CA Gen with a CI/ CD solution is great news for automating DevOps processes.

2.3.5 Test

One of the most important stages of the DevOps cycle is also one of the trickiest to solve—testing. Testing a CA Gen block-mode application on z/OS generally requires a terminal emulator scripting tool. Many organizations use a third-party product or their own homegrown solution for this purpose. There is a recently created open source project named Galasa, that may open up opportunities to use a widely available platform that would enable more standardized automation testing on z/OS. As of the time of this writing, Galasa's 3270 emulator support is still too early-in-development to easily use it without running into bugs. However, after the Galasa manager reaches a beta or GA stage, it will be a universally available option for writing automated tests for block mode screens.

The DevOps process works best when testing is done early and often during the development process. Even though the testing stage of the DevOps cycle comes after coding, the two activities often go hand-in-hand. Getting feedback that shows changes work as expected and don't negatively impact other parts of the application should be done as close to making the change as possible. This practice reduces the time it takes to fix bugs. Being able to leverage automated tests to get that feedback is ideal because it reduces the amount of manual work that has to be done by the developer. Tests are more likely to be run if they are automated and will be run in a deterministic manner.

Another benefit of running tests early in the development process is that Evan can go back to the Toolset and make any changes without having to promote the model changes to the next stage. Since the build process is easier to initiate from his local workstation, this process is streamlined, saving developer time.

To be able to actually test the application, it needs to be deployed to a CICS region. Deployment can be done automatically using a CA Endevor Move Processor as described earlier. Alternatively, the application's load modules must be manually copied into the appropriate dataset. After copying the load modules, a NEWCOPY command may be necessary to activate the changes.

2.3.6 Release and Deploy

The release and deploy processes are closely linked when using the new DevOps workflow. Also, depending on the model management philosophy, there may be another wrinkle DevOps workflow. It is common for customers to use multiple copies of the model to represent different stages of development. So, there may be a development, QA, and production copy of the model. Otherwise, similar behavior can be implemented using a typical configuration of the CA Endevor map to model the different stages of development.

2.3.6.1 CA Endevor Map

The CA Endevor map can be utilized to move changes through the different lifecycle stages. This is similar to the traditional approach to application development. In this case, releasing and deploying the application will be done through CA Endevor actions. With this approach, CA Endevor packages group changes as a logical unit and those changes flow through the different CA Endevor lifecycle stages. This approach allows the full utilization of CA Endevor's processors to deploy the application, bind to different databases for different environments, or any other custom processing that is enabled by CA Endevor.

2.3.6.2 CA Gen Model Migration

Typically, customers have multiple versions of their model stored in their encyclopedia. There will be a copy of the model for different environments, like development, QA and production. Since the model is the source of truth for any CA Gen application, model changes must be migrated between the different model copies before the changes reach production.

This means that part of the DevOps process will have to be repeated each time a migration from one level to another is done. In other words, when Carla migrates model changes from the development model to the QA model, she'll need to build, test, and migrate the changes until they reach the production environment.

Additionally, there will need to be a CA Endevor environment for each model stage that is responsible for building the application for that stage. When model changes are migrated from one stage to another, the change elements must be promoted. Then the target CA Endevor environment will need to be reloaded with new versions from the promoted elements. This process could be somewhat automated but will need some manual intervention from Carla.

Chapter 3: Summary

3.1 What Does this Mean for Carla?

Carla now gets to use the same SCM tool for CA Gen applications as the rest of her organization. This means that CA Gen is no longer an exception to the rule and can follow development processes that are more standard in the industry.

Carla can now focus more on her model management duties and spend more time managing those things rather than mechanical issues like running builds. She can leave the build management to CA Endevor and only worry about the processes that can't be automated.

In addition, Carla may have auditors from her organization that want her to provide source code for all CA Gen applications. CA Endevor provides the audit trail for the application source code with no additional effort.

Another exciting opportunity that presents itself with a DevOps-focused mindset is the opportunity to integrate with CI/CD solutions like Jenkins. In Carla's organization there is likely already a group that is ready to integrate CA Gen with the organization's choice of CI/CD server. Now Carla has a solution at hand when she is asked to integrate with the corporate CI/CD server.

3.2 What Does this Mean for Evan?

Evan likely has the most visible changes to his daily workflow. He can now be in control of building his own changes and let CA Endevor take care of building the application. Tightening this cycle of making a change in the Toolset, building those changes, and subsequently doing developer testing can save him a significant amount of time. Also shortening the time between making a change and testing a change makes it easier to develop.

Evan no longer needs to log in to complete any tasks when he uses the sample code scripts. These scripts allow him to sidestep that requirement and just focus on modeling business requirements in CA Gen.

3.3 What Does this Mean for Your Organization?

Carla and Evan are pleased that the tools they are already licensed for can integrate with each other without much effort. CA Gen has not always been thought of as a player in the DevOps space, but through the ingenuity of the CA Gen community, many DevOps problems have been solved for years. Standardizing some of the techniques that have been employed by organizations for years can yield a very robust DevOps solution.

As we have seen, combining tools from the Broadcom portfolio can result in a powerful combination of industry leading tools, leading to a very full-featured DevOps solution. Streamlining processes, reducing overhead, satisfying audit requirements, and leveraging common skills are just a few of the potential benefits for an organization looking at integrating their CA Gen portfolio with their DevOps pipeline.

The CA Gen team hopes that you have found value in this whitepaper and can utilize some of the techniques described to integrate with your organization's DevOps toolchain. We are committed to providing common-sense solutions to our customers to enable them to do more with their valuable Broadcom software portfolio. We intend to create more white papers outlining solutions for other platforms and application types in the future. Please join the CA Gen EDGE community and let us know where we can make CA Gen an even more valuable solution for you and your organization.

Revision History

CA-Gen-DevOps-WP100; September 21, 2020

- Initial release.