

E-GEN:

A DevOps Approach to Building *Batch Workflow Automation*

*Authored by S. Michael Benson
Owner and Principal Architect
EmpriZe IT Consulting, LLC*

*Sponsored by
International Software Company*



Treating Artifacts as Code in DevOps

The DevOps process relies heavily on automation of the software development lifecycle stages to achieve maximum agility while drastically shortening the time to market with new and higher quality business applications and functions.

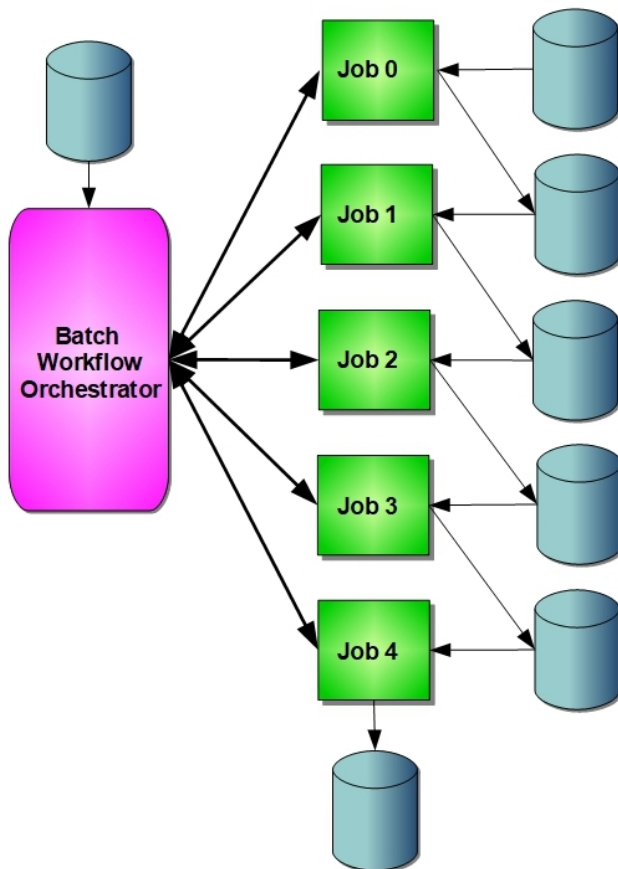
Combining the development and operations teams can help reduce the time it takes to deploy new code from development through test and into production. But there are still manual processes that take time and are error prone.

Some have tried to address this problem by treating many of the non-code artifacts as if they were just a different type of code. While this can work if they follow a simple grammar, it doesn't often work well with batch workflows.



“...there are still manual processes that take time and are error prone.”

What is a Batch Workflow?



Batch programs may evoke a picture of 1960's mainframes using punch cards and magnetic tape drives. In reality batch has modernized along with most other IT systems and is used by companies that process very large amounts of data that must remain consistent without any end user interaction.

Because the data may be loaded from different sources, the online systems are usually prevented from making changes during the batch run to ensure consistency.

Batch workflow is another way of saying background application programs as jobs that run in a controlled sequence. You may have heard this called application workflow orchestration.

“Batch workflow is another way of saying background applications programs as jobs that run in a controlled sequence.”

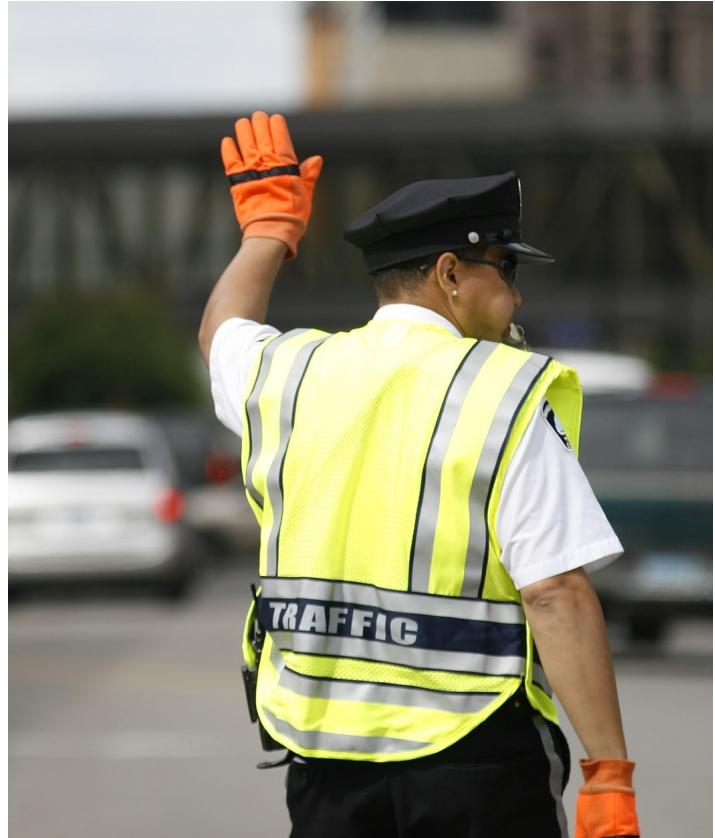
Batch Workflow Schedulers

Tools that orchestrate the batch workflows are called schedulers. Schedulers use a sequence of instructions that tells them when and how to run the batch workflow.

The language of batch workflow schedulers varies based upon the vendor and the target system. Many job schedulers have their own integrated development environments to specify the instructions.

The process of creating a batch workflow starts with creating individual batch job definitions. A job can have many application job steps that are run sequentially and require very specific resources as input and output files.

Once the individual job definitions are completed, they can be combined into workflows that are executed by the batch workflow scheduler.



“Schedulers use a sequence of instructions that tell them when and how to run the batch workflow.”

Defining the Batch Jobs

```
//SAMPJOB1 JOB 'SAMP',CLASS=6,  
//  MSGCLASS=X,REGION=8K,  
//  NOTIFY=&SYSUID  
//*  
//STEP010 EXEC PGM=ICETOOL,  
//  ADDRSPC=REAL  
//*  
//INPUT1 DD DSN=SAMP.SORT.INPUT1,  
//  DISP=SHR  
//INPUT2 DD DSN=SAMP.SORT.INPUT2,  
//  DISP=SHR,  
//  UNIT=SYSDA,  
//  VOL=SER=(1243,1244)  
//OUTPUT1 DD DSN=MYFILES.SAMPLE.OUTPUT1,  
//  DISP=(,CATLG,DELETE),  
//  RECFM=FB,  
//  LRECL=80,  
//  SPACE=(CYL,(10,20))  
//OUTPUT2 DD SYSOUT=*
```

“On IBM's zOS operating system, the jobs are defined using Job Control Language (JCL).”

While most scheduler programs offered by software vendors provide useful tools to define the interactions between the jobs that make up the batch workflow, often the actual job definitions are left to be created manually by the operations team.

Job definitions include the name of the program to run and the resources needed to run it. On IBM's z/OS operating system, the jobs are defined using Job Control Language (JCL).

JCL for a single job can include multiple steps that invoke different programs with unique resource requirements.

Manual creation of jobs is cumbersome and error prone, thus becoming an obstacle to the agility and quality expected in a DevOps environment.



Imagine having a single place to define not only batch jobs, but all of the scripts and scheduling artifacts needed to automate batch workflows.

Imagine instead of manual processes to define each of these components, you can define rules and variables that are used to generate the batch workflow artifacts.

Imagine the freedom to choose the batch workflow scheduler that is best suited for you without having to learn new development tools.

E-GEN, a product from International Software Company, is all of this and more. It eliminates many of the roadblocks to successfully using DevOps tools for agility and quality in batch workflows.

E-GEN uses functional descriptions and your company's in-house rules to automate release management. It can use this information to create any type of script, JCL, scheduler input and other components, regardless of the scheduler, to greatly simplify batch workflow changes.

Defining Global Variables and Rules

By using environment variables and rules as functional descriptions, E-GEN can reduce errors that are typically introduced by having to create the individual batch workflow artifacts manually.

By reducing manual errors in batch workflow artifacts, you can save considerable expense in having to perform problem determination to uncover where the error was injected and fix it.

By storing these environment variables and rules inside a secure central repository, you can quickly make changes to target artifacts.

By generating the batch workflow artifacts automatically, you can save the time it would take to create a change request and pass that along to the operations team to be manually built.

You can see a sample of the variables in the E-GEN screen shot below.

The screenshot shows the E-GEN Environment variables editor. The main window displays a table of environment variables. The table has columns for Variable, Description, DEV : Test, QA : Quality Assurance, and REF : Production. The table lists several variables, including APP-PFX, DB2-SUBSYS, ENV-ID, HLQ1-SFX, HLQ2, JOB-INCL-CDN, JOB-PFX, JOB-USR, JOBLIB1, OPC-SUBSYS, and PDSLIB. The table also shows the Type, Description, and Prefixed status for each variable.

Variable	Description	DEV : Test	QA : Quality Assurance	REF : Production
APP-PFX	First letter of I/VS Applications	T	#	\$
DB2-SUBSYS	DB2 Subsystem	DB2T	DB2I	DB2P
ENV-ID	Environment letter	T	#	\$
HLQ1-SFX	First qualifier suffix	DEV	INT	PRD
HLQ2	Second qualifier	TST	STD	STD
JOB-INCL-CDN	Standard Include	DEVLIB	QALIB	PRDLIB
JOB-PFX	First letter of Jobs	T	#	\$
JOB-USR	User Jobcard	S###	S###	S999
JOBLIB1	Standard Joblib	EXPDEV.SAMPLE.JOBLIB	EXPINT.SAMPLE.JOBLIB	EXPPRD.SAMPLE.JOBLIB
OPC-SUBSYS	I/VS Subsystem	OICT	OICT	OPCA
PDSLIB	Libraries Alias	EXPDEV.SMPTST	EXPINT.OPCLIB	EXPPRD.OPCLIB

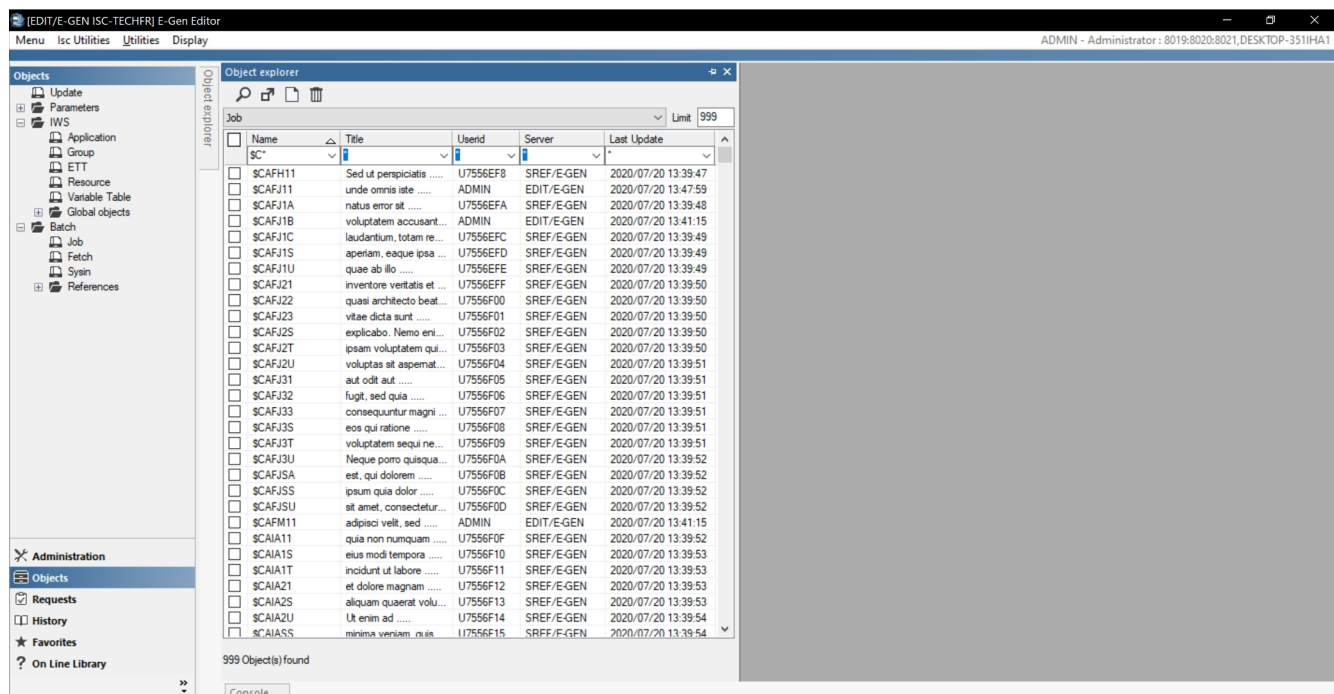
Creating and Editing Objects

You can use the E-GEN Object explorer tool to browse and select objects for editing. You can also use it to create new objects of selected types.

The partial screen shot below shows the Object explorer being used to browse a list of jobs that have already been created. These jobs are not in JCL format yet, but rather contain a functional description about parts of the job that will be used to generate the JCL.

When you select an object to be edited, you are taken to a screen that shows the various components that make up the object. From that screen, you can select individual components and make updates.

Each change that is made to the object components is logged so that you can see the exact change and who made it in case you need to roll it back for some reason.



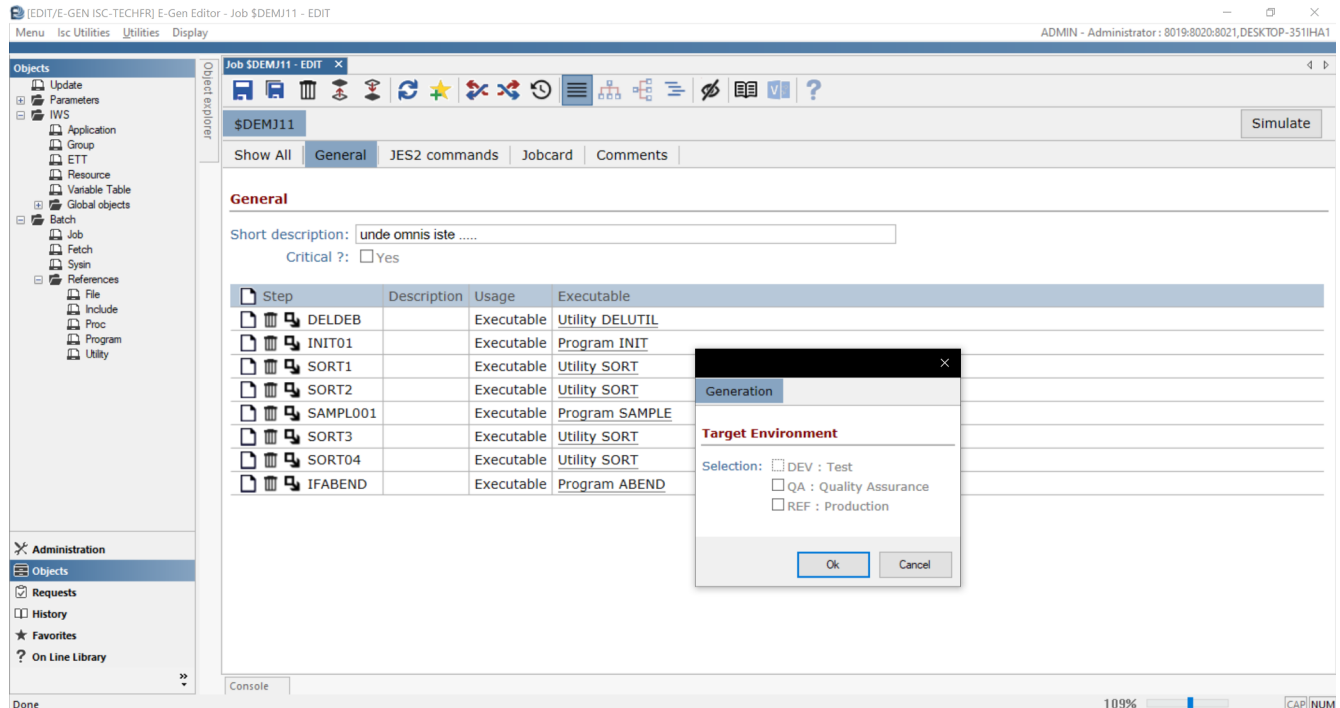
Generating Batch Workflow Artifacts

You can create complex batch workflows, including the JCL, scripts, and other scheduler artifacts for many different scheduler types, including IBM Workload Scheduler, BMC Control-M, CA/7, and others.

Not only does E-GEN support mainframe schedulers, it can also generate purely distributed scheduler artifacts like CA AutoSys, \$Universe Workload Automation, Visual TOM, and more.

You can define as many physical environments and software development lifecycle (SDL) stages you wish to support through E-GEN. For instance, you may wish to define multiple test stages in your SDL.

The E-GEN partial screen shot below shows an example of generating a batch job with several steps for two target physical environments: QA (target: Quality Assurance LPAR) and REF (target: Production LPAR).



Looking at a Generated Batch Job's JCL

The output shown below is the result of generating the JCL for the Quality Assurance environment. The generated JCL for the Production environment would be similar but with name changes as a part of the generation process. These change were specified in the global environment variables for each environment. For example, the job name prefix for QA is specified as # so the generated job name is #CAFJ11.

Other generated names in the sample output below include dataset names, INCLUDE member names, and the DB2 subsystem name.

Just imagine having to enter all of these fields manually and you can easily see the value of specifying environment variables and then using them to generate jobs conforming to your internal standards. The quality of your JCL is automatically increased by generating it.

Env	Type	PDS	Name
QA : Quality Assurance	Job	EXPINT.OPCESAI.JOBLIB	#DEM111
REF : Production	Job	EXPPRD.OPCESAP.JOBLIB	\$DEM111

Logfile

```

//**OPC SCAN
//**OPC SEARCH NAME=(GLOBAL,%JOBNAME.)
//**$DEM111 JOB (DEM),CLASS=B,MSGCLASS=M,MSGLEVEL=(1,1),USER=S999
//-----*
//**          JOB GENERATED BY E-GEN
//**   - DATE   : 2020/08/26           - TIME   : 10:42:02
//**   - USER  : ADMIN                - STAGE  : EDIT
//**-----*
//**          INCLUDE MEMBER=JOBLIB
//**
//**
//**OPC RECOVER CALLEXIT=EXIT01, ERRSTEP=DELDEB, RESSTEP=DELDEB
//**OPC RECOVER ERRSTEP=INIT01, RESSTEP=INIT01
//**OPC RECOVER ERRSTEP=SORT1, RESSTEP=SORT1
//**OPC RECOVER ERRSTEP=SORT2, RESSTEP=SORT2
//**OPC RECOVER ERRSTEP=SAMPL001, RESSTEP=SAMPL001
//**OPC RECOVER ERRSTEP=SORT3, RESSTEP=SORT3
//**OPC RECOVER ERRSTEP=SORT04, RESSTEP=SORT04
//**-----*
//** STEP: DELDEB PROGRAM : DELUTIL
//**-----*
//** DELDEB EXEC PGM=DELUTIL
//**-----*
//** STEP: INIT01 PROGRAM : INIT
//**-----*
//** INIT01 EXEC PGM=INIT,
//**          REGION=0M
//**          DD SYSOUT=*
//**          DD SYSOUT=*
//**-----* DASHBOARD FILE
//** OUTREP DD SYSOUT=*
//** DB2INP DD *
//** DB2P
//**
//**SYSACC DD *
//**$SAMJ.
//**
//**FILEMREF DD DSN=XXXPRD.STD.DEMREFS.DEMEXT01.J%JMA..$DEM111,
//**          DISP=(NEW,CATLG,DELETE),
//**          VOL=(,6),
//**          SPACE=(CYL,(50,50),RLSE),
//**          DCB=(RECFM=FB,LRECL=300,BLKSIZE=27900)
//**-----*

```

Putting the Jobs together in a Workflow

The screen shot below is showing an IBM Workload Scheduler for z/OS application that is a set of Jobs that are to be run in a particular order as specified in the application definition. Other schedulers may use different terminology for a batch workflow.

You set this up by specifying internal and external predecessors for each specific job in the workflow application. The actual component jobs have already been defined.

The batch workflow application can also be shown graphically using the E-GEN editor. This allows you to visualize the flow of jobs and their relationships.

In addition to generating the actual batch workflow application, E-GEN gives you the ability to generate technical documentation such as your operations run books and Request for Change (RFC) documents for DevOps change management.

Application DEMJ1RDJ - EDIT

Description: INTERPRETER
Status: ☒ Active ☐ Pending
Valid from (AA/MM/JJ): 20/07/2020
Priority:
Calendar ID: ...Empty...
Owner text:
Owner ID: DEM
Group definition: ...Empty...
Auth. Group ID: MVS

Runcycle

Period/RG or rule name	Period / Run Cycle Group	Run type	Free-day Rule	Input arrival time	Deadline day	Deadline time	Spec. Env.
DAILY		R: Regular (RULE)	1	19:00	01	06:59	<input type="checkbox"/> INT <input type="checkbox"/> HML <input type="checkbox"/> PRD

Operation

Operation	Description	Job	Int. predecessor	NB Ext. predecessor	ExNo	<input type="checkbox"/> INT <input type="checkbox"/> HML <input type="checkbox"/> PRD
BR14-001	START			1		<input type="checkbox"/> INT <input type="checkbox"/> HML <input type="checkbox"/> PRD
CPUA-015	INTERPRETATION	\$DEM1RA	001			<input type="checkbox"/> INT <input type="checkbox"/> HML <input type="checkbox"/> PRD
CPUA-020	BN ACCOUNT	\$DEM1RB	015			<input type="checkbox"/> INT <input type="checkbox"/> HML <input type="checkbox"/> PRD
CPUA-025	DEM COLLECTION	\$DEM1RC	020			<input checked="" type="checkbox"/> INT <input type="checkbox"/> HML <input type="checkbox"/> PRD
CPUA-030	DENOTING	\$DEM1RD	020			<input checked="" type="checkbox"/> INT <input type="checkbox"/> HML <input type="checkbox"/> PRD
CPUA-035	GROUPED EDITIONS	\$DEM1RE	025 030			<input type="checkbox"/> INT <input type="checkbox"/> HML <input type="checkbox"/> PRD
CPUA-040	FILE TRANSFER	\$DEM1RF	025			<input type="checkbox"/> INT <input type="checkbox"/> HML <input type="checkbox"/> PRD

Looking at a Generated IWS Application

The partial screen shot below shows the IBM Workload Scheduler for z/OS application that was generated. This application was built from the functional model that was previously built through a simple E-GEN dialog.

You can see that the IWS application has a very specific syntax and many control statements and parameters. Getting these correctly specified the first time is very difficult. Making changes can be challenging.

E-GEN takes the risk out of making manual changes to the IWS application. E-GEN has been shown to require 75% fewer manual activities which results in much higher quality.

The IWS application is a scheduler artifact that is used to automate batch workflows. Other schedulers use their own artifacts and terminology but E-GEN takes care of that for you when you specify the target scheduler.

Env	Type	PDS	Name
REF : Production	Tws-application	OPCA	\$DEMJ1RDJ
REF : Production	Job	EXPPRD.OPCESAP.JOBLIB	\$DEMJ1RA
REF : Production	Job	EXPPRD.OPCESAP.JOBLIB	\$DEMJ1RB
REF : Production	Job	EXPPRD.OPCESAP.JOBLIB	\$DEMJ1RC
REF : Production	Job	EXPPRD.OPCESAP.JOBLIB	\$DEMJ1RD
REF : Production	Job	EXPPRD.OPCESAP.JOBLIB	\$DEMJ1RE
REF : Production	Job	EXPPRD.OPCESAP.JOBLIB	\$DEMJ1RF
REF : Production	Job	EXPPRD.OPCESAP.JOBLIB	\$DEMJ1RG
REF : Production	Job	EXPPRD.OPCESAP.JOBLIB	\$DEMJ1RH
REF : Production	Job	EXPPRD.OPCESAP.JOBLIB	\$DEMJ1RI
REF : Production	Job	EXPPRD.OPCESAP.JOBLIB	\$DEMJ1RJ
REF : Production	Job	EXPPRD.OPCESAP.JOBLIB	\$DEMJ1RK
REF : Production	Job	EXPPRD.OPCESAP.JOBLIB	\$DEMJ1RL

```

OPTIONS DURUNIT(SECONDS)
ADSTART ACTION(ADD)
ADID($DEMJ1RDJ) ADVALFROM(200720)
DESCR('INTERPRETER')
ADTYPE(A)
OWNER('DEM')
ADSTAT(A)
GROUP(MVS)
ADRUN ACTION(ADD)
NAME(DAILY) RULE(1) VALFROM(200101) VALTO(711231)
DESCR('A completer')
TYPE(R)
IATIME(1900) DLDAY(01) DLTIME(0659)
ADRULE
EVERY
DAY(DAY)
MONTH
ADOP ACTION(ADD)
OPNO(001) JOBNREP(DEBBLOC) WSID(BR14) ADOPCATM(N)
CONDRJOB(N)
DESCR('START')
DURATION(37)
AEC(Y)
AJSUB(Y) AJR(Y) TIME(N) CLATE(N)
PSNUM(1)
ADOPPHOTO(N) MONITOR(N)
ADOPJOBCRT(N)
ADOPUSRSYS(N)
ADOPEXPJCL(Y) CSCRIPT(N) USEXTNAME(N) USEXTSE(N)
USesai(N)
ADOPMH(N)
ADOPNOP(N)
ADDEP ACTION(ADD)
PREADID($DEMJ1RDJ)
PREWSID(CPUA) PREOPNO(255)
PRESEL(C)
PREMAND(N)
ADOP ACTION(ADD)
OPNO(015) JOBN($DEMJ1RA) WSID(CPUA) ADOPCATM(N)
CONDRJOB(N)

```

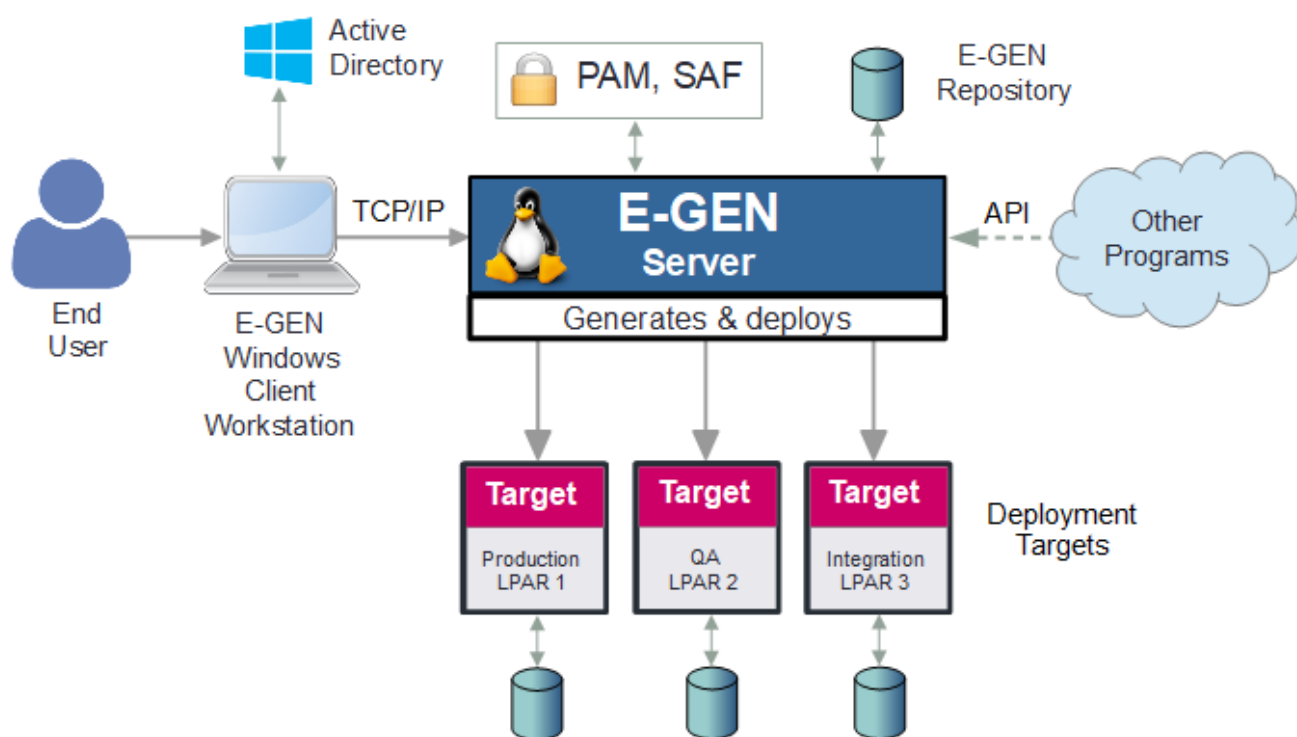

E-GEN's Client-Server Architecture

E-GEN's architecture is based on a client-server model. The client workstation runs in a Windows environment and connects with the E-GEN server over TCP/IP. You can deploy clients on individual workstations, shared directories, or through Citrix.

You can also have clients that use the E-GEN APIs to communicate requests programmatically to enable DevOps pipeline integration.

Servers can be deployed to z/OS, Linux on IBM Z, Unix, Linux, or Windows. The diagram below shows the server on a Linux system.

Deployment targets do not have to reside on the same platform as the server. For instance, you can deploy to a z/OS platform from a server running on Linux as shown in the diagram. Agent and iPack are other ISC products that can help make the deployment process even simpler.



Using iCAN for Initial Data Creation

With ISC's premier product for automating the creation of critical operational documentation, iCAN can help you collect all of the batch workflow artifacts that are used as input to the E-GEN process.

iCAN also analyzes dependencies and relationships so that you do not miss critical elements of the batch process definitions as you are defining your environment variables and functional definitions.

iCAN works with many different schedulers to provide you with a complete view and not just a scheduler-centric picture.

The iCAN screen shot below shows some of the schedulers and other products that can have data collected for documentation. In this example, data is being collected for IBM Workload Scheduler, Control-M, \$Universe and other JCL objects.

The screenshot shows the iCAN Editor interface. The main window is titled "Collect Configuration DEFAULT". It has a menu bar (Menu, Isc Utilities, Utilities, Display) and a toolbar. On the left is an "Object explorer" pane with a tree view showing "Administration" (Definitions, Security, Generation, Collect configuration, Collect Configuration, Object List, Syntax, Syntax Properties) and "Administration" (Objects, Requests, History, Favorites, On Line Library). The main area displays the "Collect Configuration DEFAULT" window with tabs for "Show All", "Description", and "Advanced". The "Description" tab is active, showing a "Description" field with "Default configuration", "Last Run" with "2020/07/10 10:39:35", "Last Return Code" with "0", and "Source Directory" with "/data/sample/inputs/". Below this is a table of collectors.

Collectors	Active	Verbose	Last Run	Last Return Code
TWS Scheduling	<input checked="" type="checkbox"/> Yes	<input type="checkbox"/> Yes	2020/07/10 10:26:27	0
TWSd Scheduling	<input type="checkbox"/> Yes	<input type="checkbox"/> Yes	Unavailable	0
CA/7 Scheduling	<input type="checkbox"/> Yes	<input type="checkbox"/> Yes	Unavailable	0
Control/M Scheduling	<input checked="" type="checkbox"/> Yes	<input type="checkbox"/> Yes	2020/07/10 10:30:52	0
Zeke Scheduling	<input type="checkbox"/> Yes	<input type="checkbox"/> Yes	Unavailable	0
\$Universe	<input checked="" type="checkbox"/> Yes	<input type="checkbox"/> Yes	2020/07/10 10:33:18	0
CA AutoSys	<input type="checkbox"/> Yes	<input type="checkbox"/> Yes	Unavailable	0
Visual TOM	<input type="checkbox"/> Yes	<input type="checkbox"/> Yes	Unavailable	0
Program Sources	<input type="checkbox"/> Yes	<input type="checkbox"/> Yes	Unavailable	0
Ims Definitions	<input type="checkbox"/> Yes	<input type="checkbox"/> Yes	Unavailable	0
Cics Definitions	<input type="checkbox"/> Yes	<input type="checkbox"/> Yes	Unavailable	0
Cft Definitions	<input type="checkbox"/> Yes	<input type="checkbox"/> Yes	Unavailable	0
List Catalog	<input type="checkbox"/> Yes	<input type="checkbox"/> Yes	Unavailable	0
JCL Objects	<input checked="" type="checkbox"/> Yes	<input type="checkbox"/> Yes	2020/07/10 10:39:35	0

The status bar at the bottom shows "Done", "Console", "100%", and "CAP|NUM".

5 Ways E-GEN Provides Value

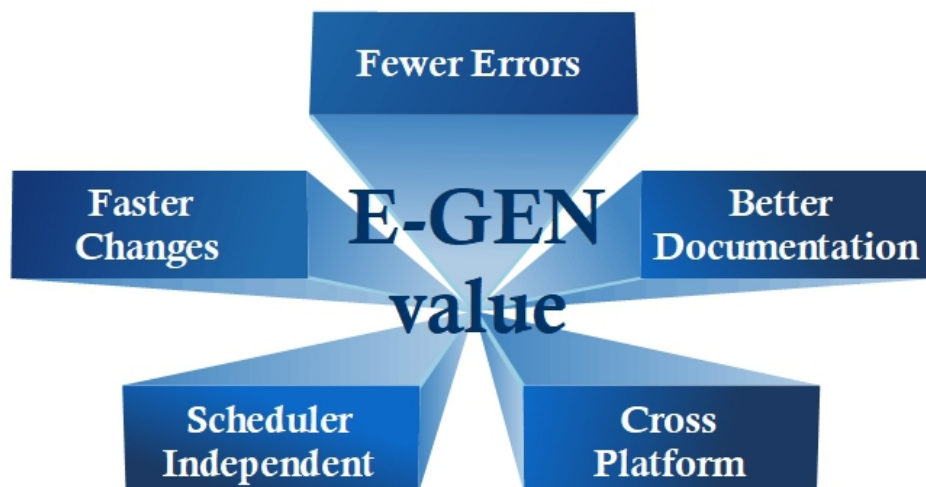
1 *Faster Changes* – E-GEN integrates with most DevOps pipeline automation tools for lightening fast deployment.

2 *Fewer Errors* – On average there are 75% fewer manual activities which are the leading cause of deployment errors, thereby reducing development expense.

3 *Better Documentation* – Documentation is automated and takes 90% less time on average to create using E-GEN and iCAN.

4 *Cross Platform* – You can run the server on any z/OS, Linux on IBM Z, other Linux and Unix distributions, or Windows environment and deploy to any server in your enterprise.

5 *Scheduler Independent* – You no longer have to rely on the definitional tools of your batch automation scheduler vendor. You can create the rules and global variables that can generate artifacts for any of the most popular scheduler products.



About International Software Company

International Software Company (ISC) is a leader in designing innovative solutions for IT Operations. Companies around the globe rely on our products to increase the efficiency and decrease the costs of their IT production. For 25 years, we have specialized in automating the change-management process, and in providing tools that help software engineers understand the complexities of their information systems. ISC is a privately held company with offices in Belgium and France. You can find more by visiting us at <http://www.iscsoftware.com/?lang=en>

About EmpriZe IT Consulting, LLC

EmpriZe IT Consulting, LLC offers several different consulting and freelance services predominately for IBM mainframe clients. S. Michael (Mike) Benson, the founder and principal architect, has over 40 years of IT experience. Mike left IBM after 30 years of service as an Executive IT Architect in technical sales where he worked with numerous clients creating IT solution architectures. You can find more by visiting us at <http://emprizeitconsulting.com>