

Digital Transformation with IBM Application Discovery

Suman Gopinath





Digital Transformation with IBM Application Discovery

This IBM® Redpaper™ publication describes how IBM Application Discovery (AD) complements IBM z/OS® Connect Enterprise Edition and IBM Developer for z Systems® in making older mainframe applications available to the digital world.

By using a sample scenario, this publication primarily focuses on how the functionality of AD can be used to discover application programming interface (API) candidates and how z/OS Connect can easily create an API out of the mainframe program. It also describes how IBM Developer for z Systems acts as the tool that links the entire transformation.

API enablement

Many enterprises successfully use the improvements in mainframe technologies, such as Java, web services, connectivity options by using JDBC, and IBM MQ to connect the mainframe to the outside world. The advent of the mobile era meant that these organizations had to go one step further.

In 2016, mobile web usage overtook desktop for the first time and the expectation is that Internet of Things will increase this usage by many times. Large enterprises with mainframe back-end systems use methods to make available that function to mobile applications by using APIs.

APIs provide a standard interface definition to a particular service. Over the years, we used other terms, such as *service-oriented architecture*, to describe the method of breaking up your system into more manageable parts that can be called independently. This process evolved from web services to more lightweight protocols.

Today, that interface definition is primarily managed in terms of REST calls with a JSON schema to provide the abstraction layer to the back-end function. The REST interface makes it easy to manage devices or mobile applications.

APIs are behind many of the daily functions we perform, such as when you book an airline ticket, interact with your bank by using your mobile phone, or use any mobile application that is using an API to access a mainframe application. With the growing standardization of APIs by using REST, an API economy is developing to allow new business to be created by using these APIs from other companies.

IBM provides z/OS Connect Enterprise Edition to simplify creating APIs for the applications and data that is in z/OS. z/OS Connect provides the tools with which developers easily can use the functionality for new possibilities. You can then use the capabilities of an API manager to make these services available to the broader community.

Unfortunately, many of today's mainframe applications are not designed as APIs and grew to be monolithic applications. These applications were updated and expanded over the last 30 years. They likely were made available by using web services or IBM MQ over the years but can still contain the original 3270 display logic intermixed into the application. The question becomes how to find and make available the correct capabilities as APIs.

In addition, today's market requires that organizations respond faster than ever to changes and business demands. This ability requires a change in the way application development is done. This change is the reason that mainframe applications must be included in the overall DevOps transformation for development practices.

However, even with a full DevOps pipeline, you still must understand the application. The application developers that built it might no longer be available and they are not aware of all of the variances the application now includes because it grew over so many years. It is here in this process where IBM's Application Discovery and Delivery Intelligence (ADDI) plays a key role. For this IBM Redpaper publication, we focus on the Application Discovery portion of ADDI.

Scenario

General Insurance Application (GenApp) is an application in Insurance Company AXZ. The application has a 3270 interface that can be accessed from a terminal. The application uses a BMS map to control the screen layout for the 3270 interface. It enables businesses to create policies for motor insurance, endowment, house insurance, and commercial property. It also allows the Business Operations to access the customer's profile and add customers.

The Business Operations personnel of GenApp raised a request for a better and modern GUI that can be accessed from their mobile devices and tablets, starting with the customer inquiry screens. The IBM CICS® green screen transaction that is entered by operations to access the customer information is SSC1.

Tip: The general insurance application is a working COBOL CICS application that is based on VSAM files and IBM DB2®. It is publicly available as a CICS SupportPac CB12 and can be downloaded from the [CB12: General insurance application \(GENAPP\) for IBM CICS TS page](#) of the IBM Support website.

Discovering your applications

The prerequisite to embarking on the digitalization journey is to understand your mainframe applications. IBM ADDI empowers architects and developers to understand their application landscape. It can rapidly analyze and visualize applications across languages and environments and provide detailed graphical views, program and transaction flow diagrams, impact analysis reports, usage reports, and help size changes by using industry-standard metrics.

Note: ADDI can connect to various mainframe source code management repositories and schedulers for analysis. The components of an application or a set of applications are organized into a project in ADDI. The project is then parsed and built into metadata. ADDI includes an eclipse-based client that developers can use to run reports, graphs, and impact analysis (see Figure 1).

The build process can be scheduled as required; for example, every weekend, or every release of code into a QA environment or production. ADDI can be scheduled to detect changed components and perform what is known as a “make” or a build of the changed components only.

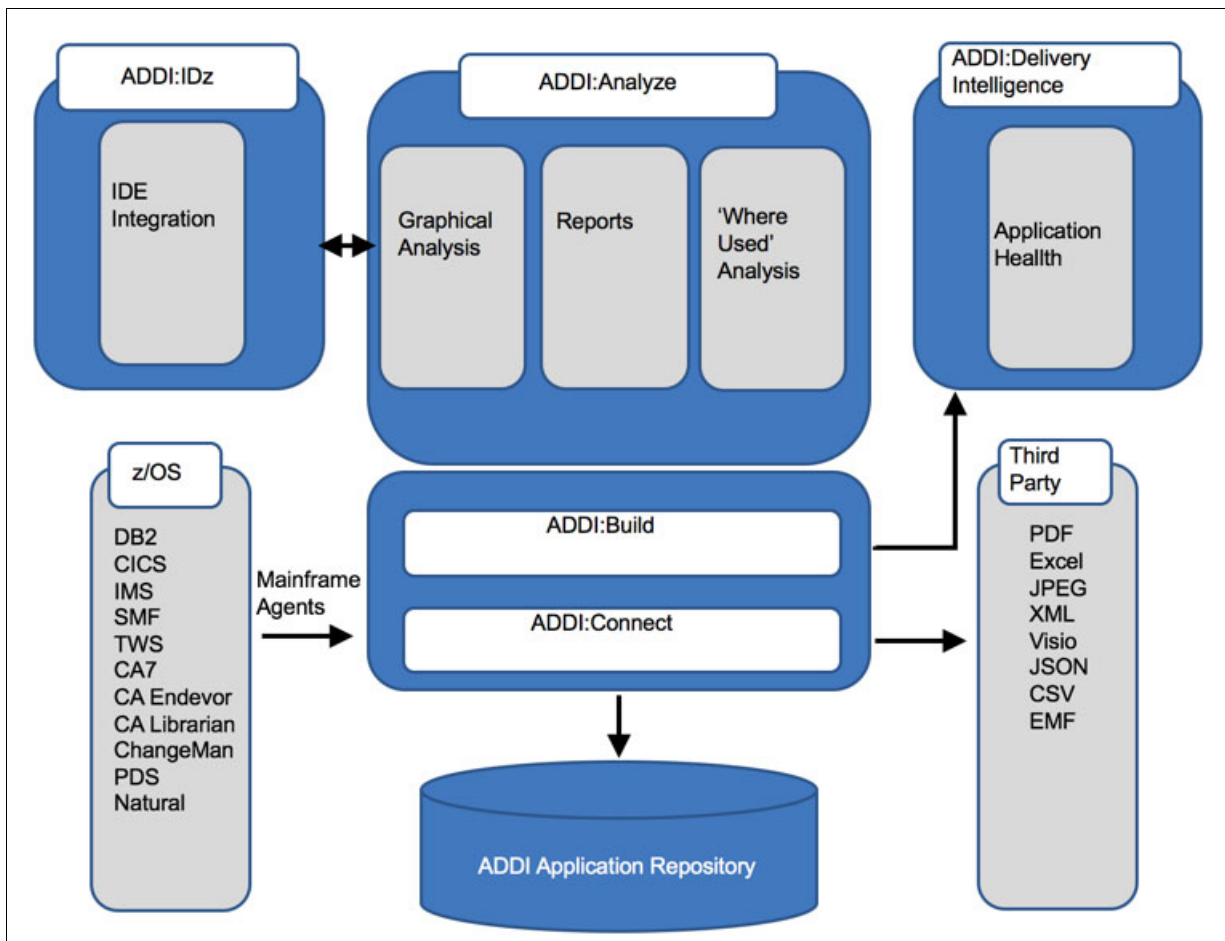


Figure 1 ADDI architecture

The Application Delivery Intelligence (ADI) part of ADDI serves as a dashboard for the applications that are analyzed by using AD. ADI can also perform a trend analysis of the static performance of these applications. This Redpaper publication describes the discovery (or AD) part of ADDI.

This section describes an example of how ADDI can be used to identify the program that can be made available as an API.

Analyzing an application

If the requirement to create an API is specific and points to a green screen transaction (SSC1 in this case), ADDI can be used to analyze the transaction, identify the programs that make up the transaction, their interfaces, copybooks, and so on. This feature simplifies the process of identifying the program that can be made available as an API.

The following steps describe one of the many ways to use ADDI to analyze the transaction:

1. Use transaction call graphs to visualize the application and identify the COBOL program that is started by the transaction (see Figure 2).

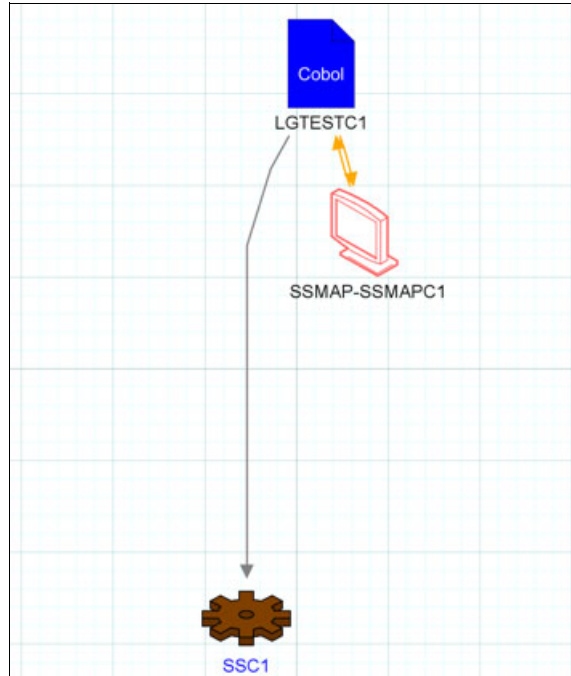


Figure 2 Transaction Call graph from IBM ADDI

2. Use Screen Layout to visualize the CICS screen. Compare this visualization against the actual SSC1 green screen that is shown in Figure 3.

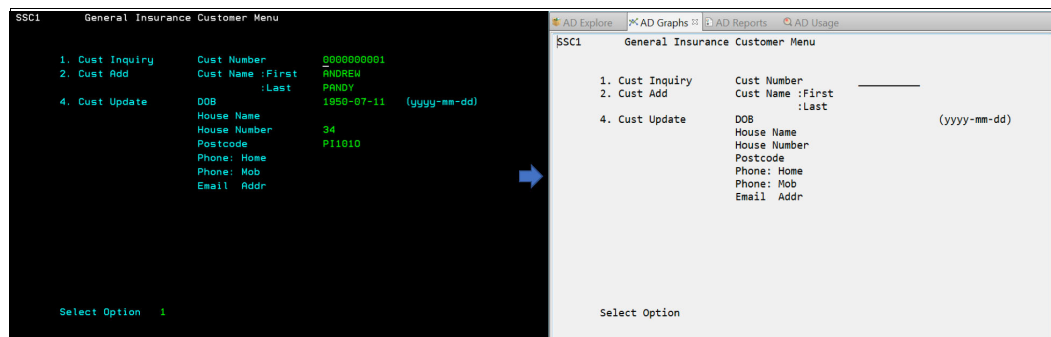


Figure 3 CICS screen

- Trace the functional flows from the main program to a data source by drawing a Program Call Graph, as shown in Figure 4.

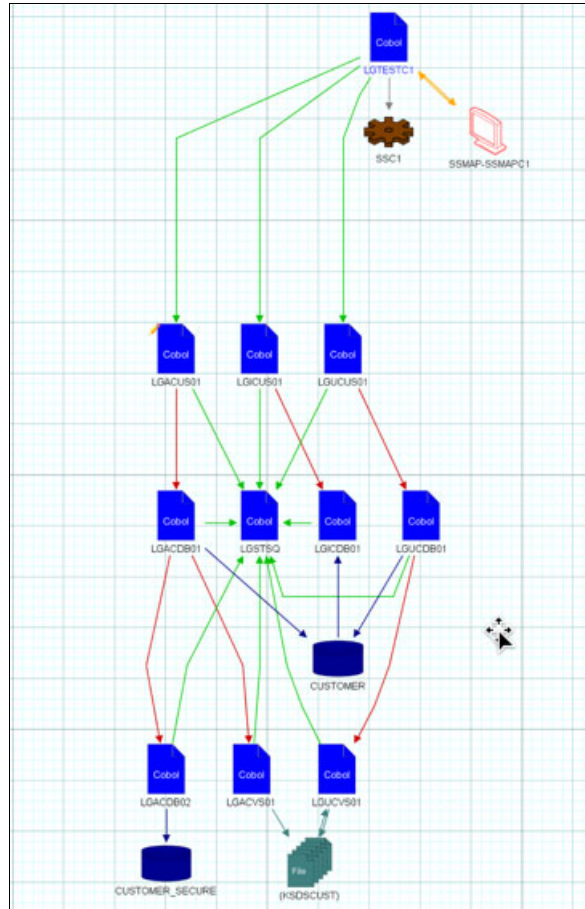


Figure 4 Program Call Graph example

Note: The different symbols that are shown in the figures in this section indicate programs, databases, transactions, and screens. The color-coded lines refer to database calls, CICS links, and dynamic calls. Arrows that lead into a database or file indicate an update or insert, whereas arrows that lead out indicate a select or a read.

4. Isolate the required flows and eliminate the noise by using the Filter functions (see Figure 5).

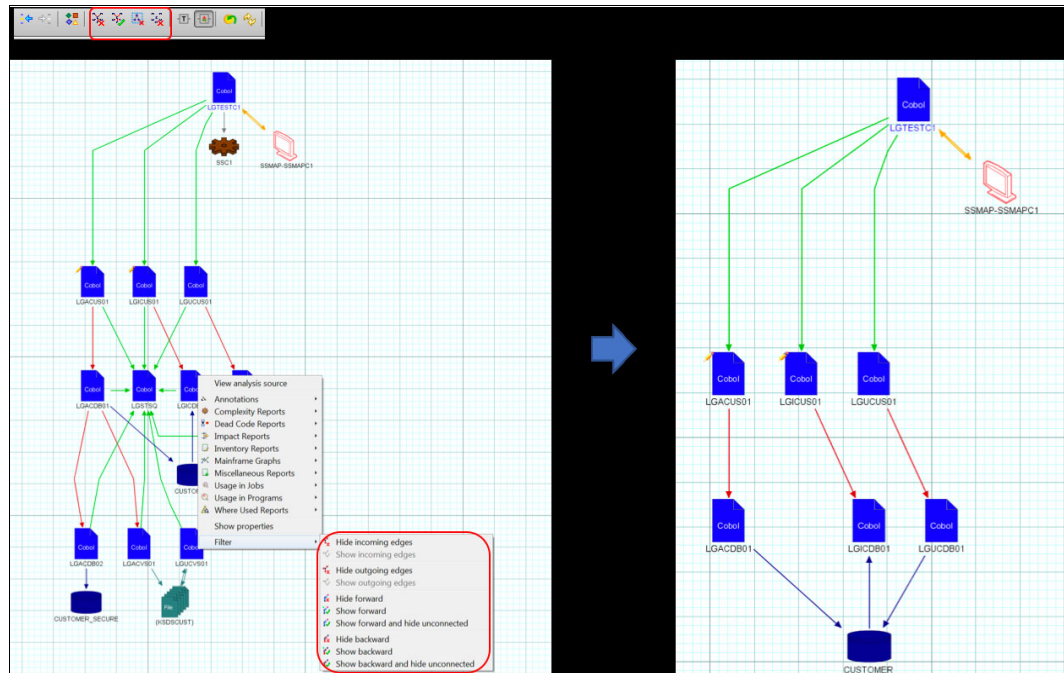


Figure 5 Filtering and Isolating required flows

In an application where a distinction is made between the presentation layer and the rest of the logic, the program that is closest to the main program that is starting the screen is the best candidate for an API because this process encompasses the required functionality. In this example, LGICUS01 is an ideal candidate.

Note: An API call cannot display screens; therefore, screen-based programs cannot be made available as an API.

5. Identify the program that can be used as an API, its interfaces, and drill-down to the invocation method in the code, as shown in Figure 6.

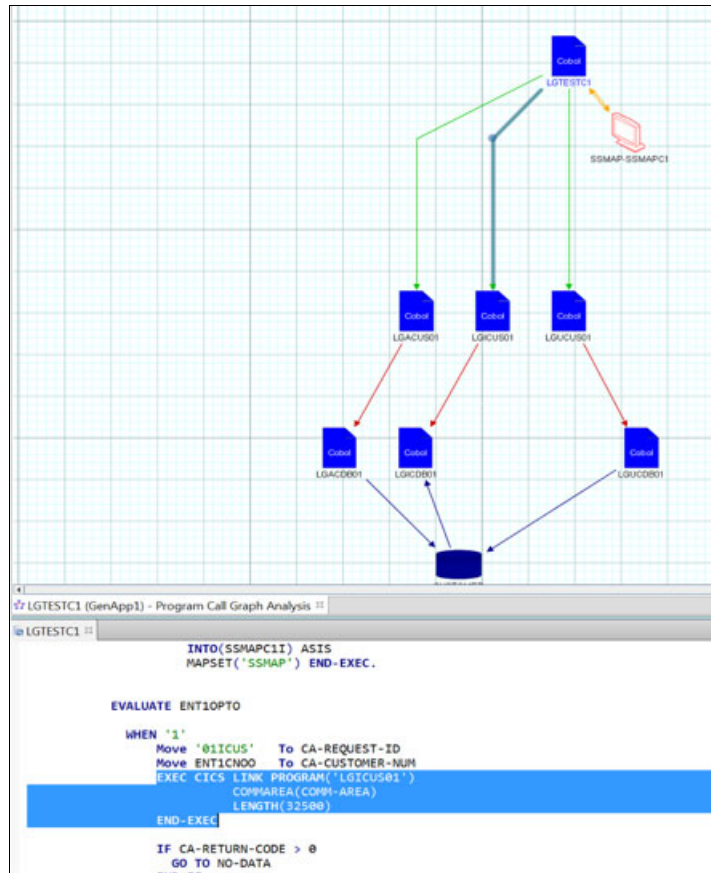


Figure 6 Identifying the interface to API candidate program

Note: The code that is behind the interfaces in the Program Call Graph can be seen by double-clicking the interface lines. Similarly, other entities, such as variable names in Usage reports and paragraph names in Program Flow graphs, can be double-clicked to drill down to the corresponding source code.

6. Drill down to the interface structures that can be used to create an API by using variable usage reports (see Figure 7).

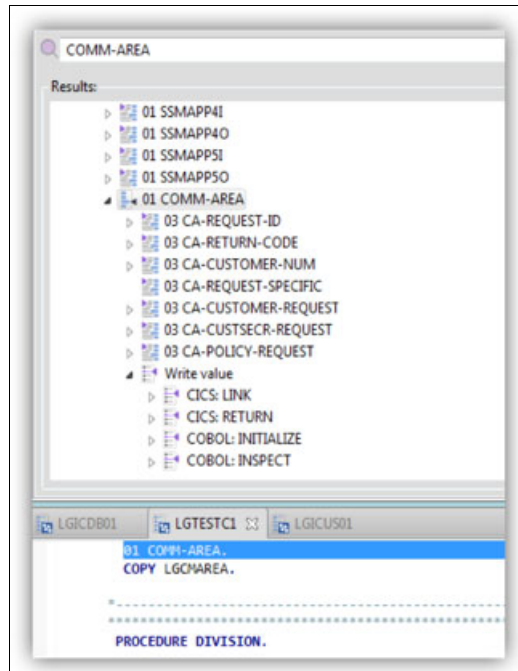


Figure 7 Identifying the interface copybook or structure

Discovering APIs

Sometimes the requirement to create an API is not specific to any screen or transaction. ADDI can still be used to discover and isolate functionalities or potential candidates for an API.

By using the circular view and running a transaction or program call graph for the entire application, parts of the application that are closely related can be identified. This view groups components that are accessing similar tables, programs, and so on. This feature helps to break down a complex application into sections, which makes it easier to choose a starting point for creating an API (see Figure 8).

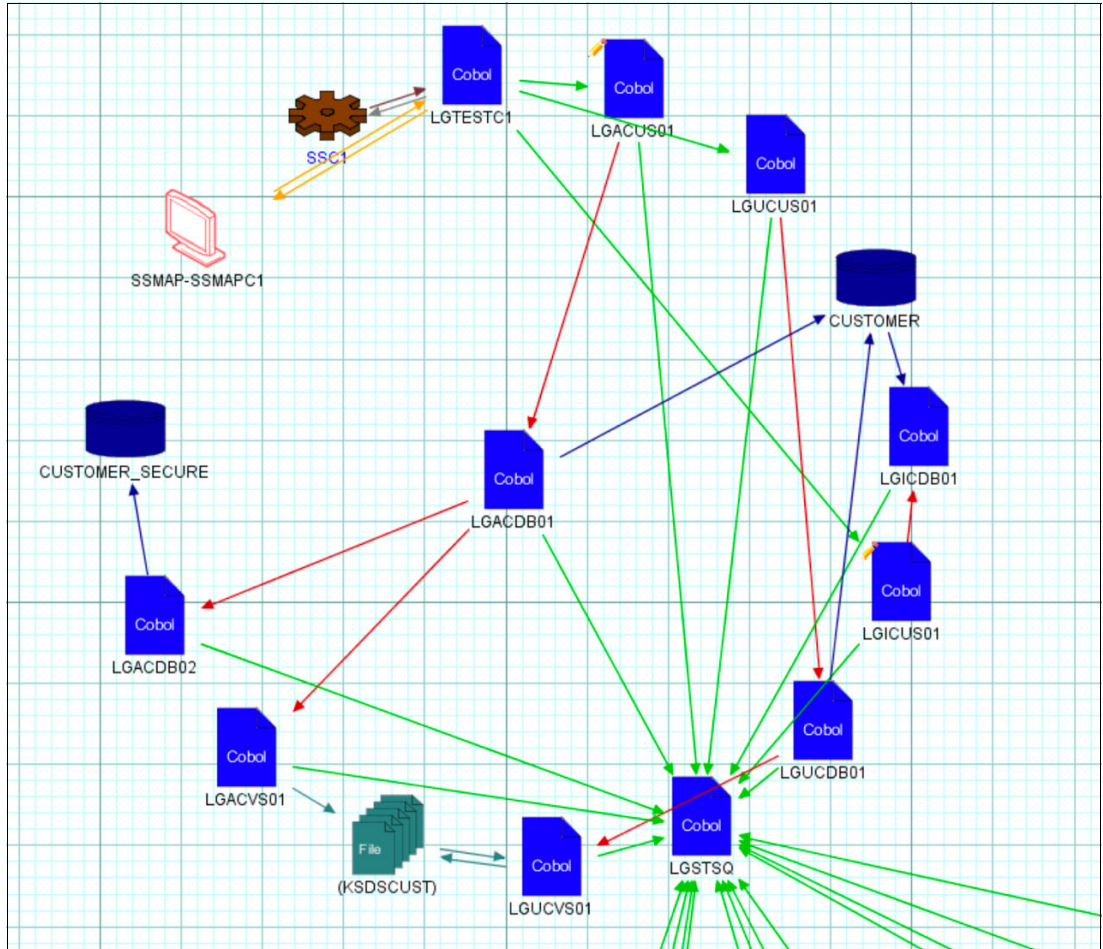


Figure 8 Close-up view of the circular view that groups programs that access common artifacts

ADDI parses the source code and subsystem information into its own SQL Server-based metadata. This parsing gives a developer the ability to create reports that are based on the metadata that is provided, and extend the large repertoire of graphs and reports that are ready for use.

This feature, which is known as Rule Based Analysis, can also be used to create custom reports that help identify candidates for creating an API. For example, one place to look for an API is among the set of programs that are accessed by the map or presentation layer.

Another potential API for policy-related applications is the program that includes the maximum rules on POLICY. SQL queries, such as the query that is shown in Figure 9, can be created on the metadata tables. This query retrieves all programs with IF/EVALUATE statements on variables that include the word POLICY in them, which is ordered by the programs with the highest conditions.

```

Select progr.ProgramName,Varb1.VarName, stat.description as stat from EZ_GenApp1.dbo.StatementReference statref
inner join EZ_GenApp1.dbo.Statements stat
on Stat.StatementType = statref.StatementType
inner join EZ_GenApp1.dbo.OccurrencesStat Occ
on Occ.OccurID = statref.OccurID
inner join EZ_GenApp1.dbo.Programs progr
on progr.ProgramID = Occ.ProgID
INNER JOIN EZ_GenApp1.dbo.Variables Varb1
ON StatRef.ResourceID = Varb1.VarID
and IsCopy = -1
inner join EZ_GenApp1.dbo.Occurrences Occ2
on Occ2.OccurID = Varb1.OccurID
inner join EZ_GenApp1.dbo.Paths paths2
on paths2.PathID = occ2.pathid
where statref.statementType in (18,23)
and
VarName in (select varb.varname from EZ_GenApp1.dbo.Variables varb
inner join EZ_GenApp1.dbo.Occurrences Occ1
on Occ1.OccurID = varb.OccurID
inner join EZ_GenApp1.dbo.Paths paths1
on paths1.PathID = Occ1.PathID
where IsCopy = -1
and varb.varname like '%POLICY%')
order by ProgramName
  
```

Figure 9 Creating queries on the AD metadata

For more information about the database schemas that can be queried, see the ADDI documentation.

The queries that were created can be entered as a rule in a custom rules properties file, which standardizes the queries across the different developers that are accessing AD. Different queries that were created can be clustered into groups with appropriate descriptions. A sample custom rules file is shown in Figure 10.

```

custom Rules.properties - Notepad
File Edit Format View Help
2.param.default.value = 4000

##group definition
##groups are optional

#<group.id>.group.name = <text : group name in gui>
#<group.id>.group.[parent] = [(0|{include.key}):]{group.id}
#<group.id>.group.[description] = <text : group description in gui>

##rule definition
##at least one of .class or .query must exist
##.query and .selectiveQuery can be used together for efficiency reasons.
## Only one is executed depending on user selection

#<rule.id>.rule.name = <identifier: rule name>
#<rule.id>.rule.[sourceBased] = true|false (default true)
#<rule.id>.rule.[weight] = <number> (default 1)
#<rule.id>.rule.[query] = <path : to query file> (default none)
#<rule.id>.rule.[selectiveQuery] = <path : to selective query file> (default none)
#<rule.id>.rule.[class] = <class> (default none)
#<rule.id>.rule.[groups] = [(0|{include.key}):]{group.id}[ , [(0|{include.ke}
(default none)
#<rule.id>.rule.[description] = <text: rule description> (default none)
  
```

Figure 10 Creating custom rules

The Rule-based analysis can then be started from the AD client, as shown in Figure 11.

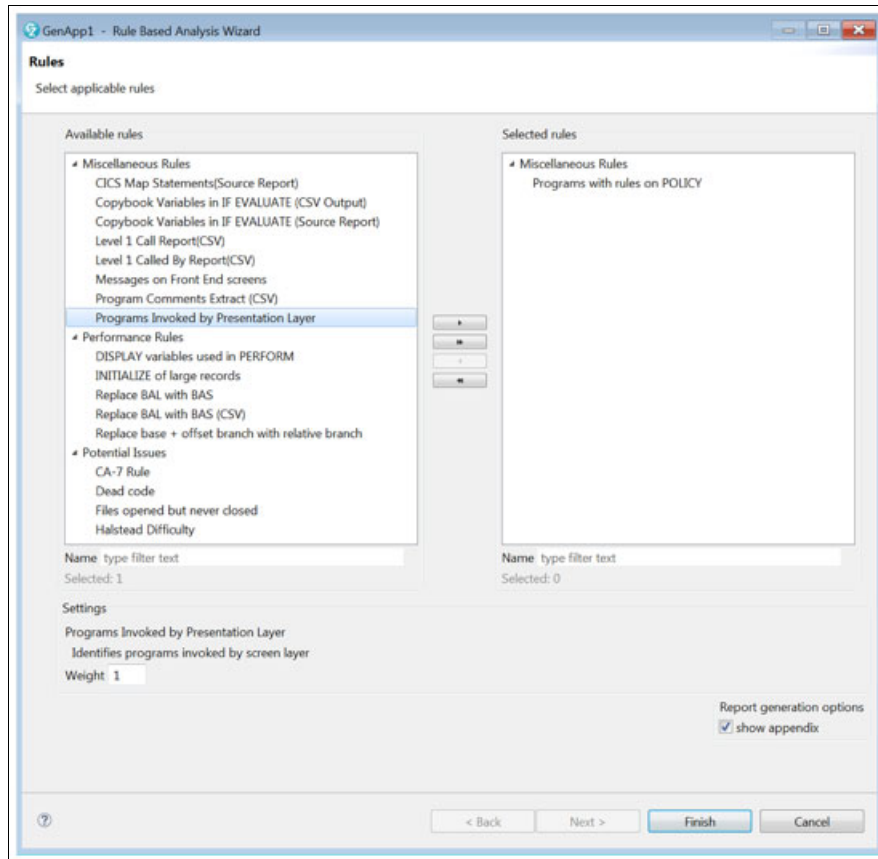


Figure 11 Starting the rules-based analysis from the browser

The results are shown in Figure 12.

2	LGICDB01	DB2-POLICYNUMBER	COBOL: EVALUATE
3	LGIPDB01	DB2-POLICYNUMBER	COBOL: IF
4	LGIPDB01	DB2-POLICYNUMBER	COBOL: IF
5	LGIPDB01	DB2-POLICYNUMBER	COBOL: IF
6	LGIPDB01	DB2-POLICYNUMBER	COBOL: IF
7	LGIPDB01	DB2-POLICYNUMBER	COBOL: IF
8	LGIPDB01	DB2-POLICYNUMBER	COBOL: IF
9	LGTESTP1	CA-POLICY-DATA	COBOL: EVALUATE
10	LGTESTP2	CA-POLICY-DATA	COBOL: EVALUATE
11	LGTESTP3	CA-POLICY-DATA	COBOL: EVALUATE

Figure 12 Results

Simplifying application changes by using IBM Developer for z Systems

For most enterprise mainframe customers, the programs are not always classified into a presentation and business layer. The copybooks also do not always conform to a definitive standard. Therefore, some amount of refactoring is required to ensure that the presentation logic is separated out of the program from which an API must be created.

Because AD and the IBM z/OS Connect Enterprise Edition tools are on Eclipse, these products integrate seamlessly with IBM Developer for z Systems (IDz). A developer can easily switch between the analysis view and the development view of a program in IDz.

Starting IDz functionality from the AD perspective is shown in Figure 13.

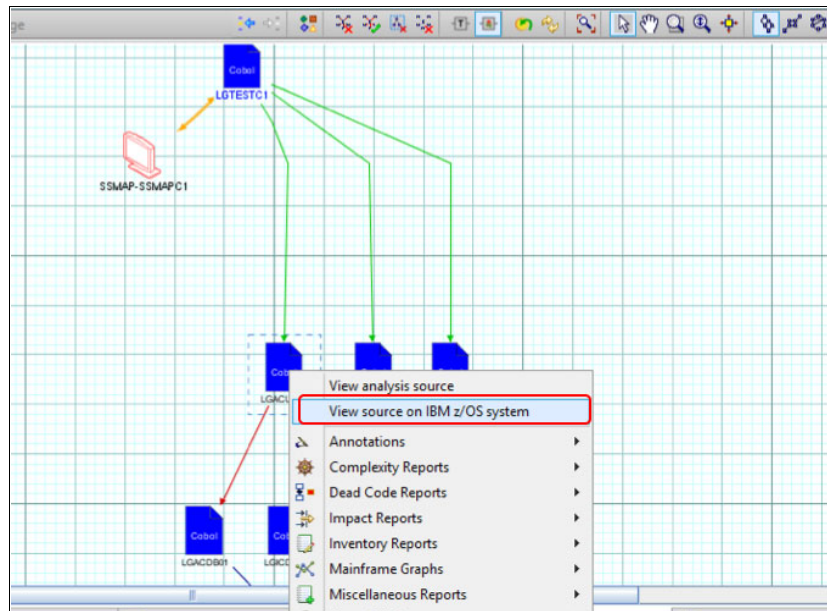


Figure 13 Starting IDz functionality from the AD perspective in Eclipse

Starting AD from the z/OS perspective is shown in Figure 14.

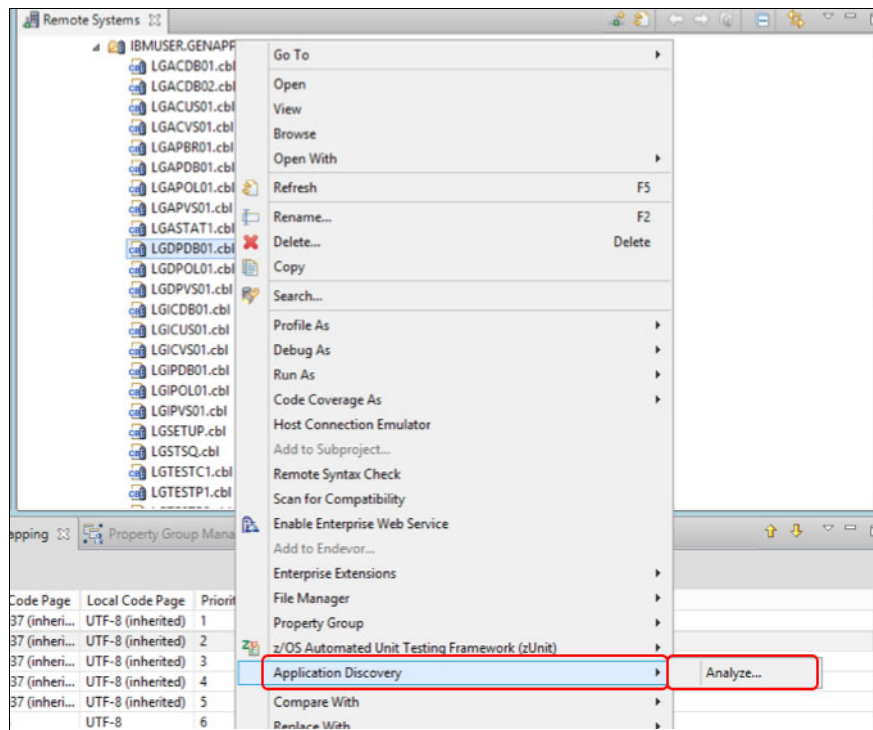


Figure 14 Starting AD from within the z/OS perspective in IDz

Refactoring source code

If the program that is required to create an API is monolithic (as in most cases), the code must be refactored to separate out the business logic from the presentation logic. In combination with AD, IDz provides accelerators with which this process can be done.

The following features are available:

- ▶ The Filter feature from the Program Flow graph can be used to isolate parts of the business logic. As shown in Figure 15, the calls to LGICDB01 are isolated.

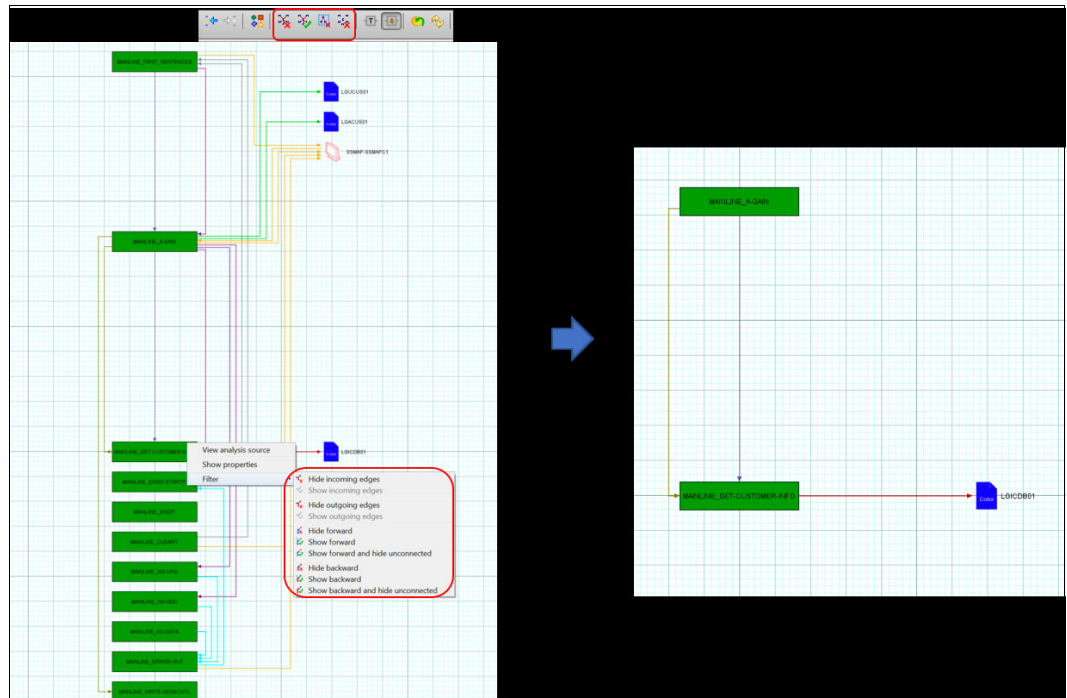


Figure 15 Filtering specific logic from the Program Flow graph

- ▶ Based on Eclipse, IDz includes a feature to create snippets that contain portions of code, which can be inserted into any program that is being developed. This feature also can be used to create snippets out of paragraphs that can be moved to a new program (see Figure 16).

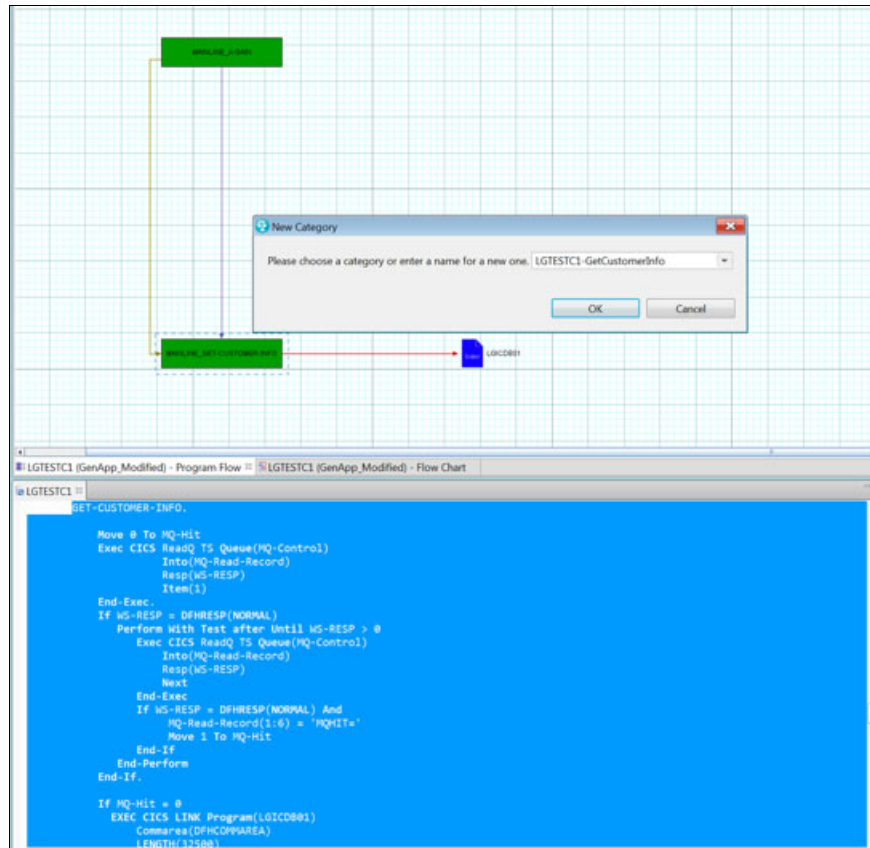


Figure 16 Drilling down from isolated flows to source code to create snippets

- IDz can create programs from defined templates. With the combination of these features and snippets, a program that features extracted logic can be easily created (see Figure 17 and Figure 18).

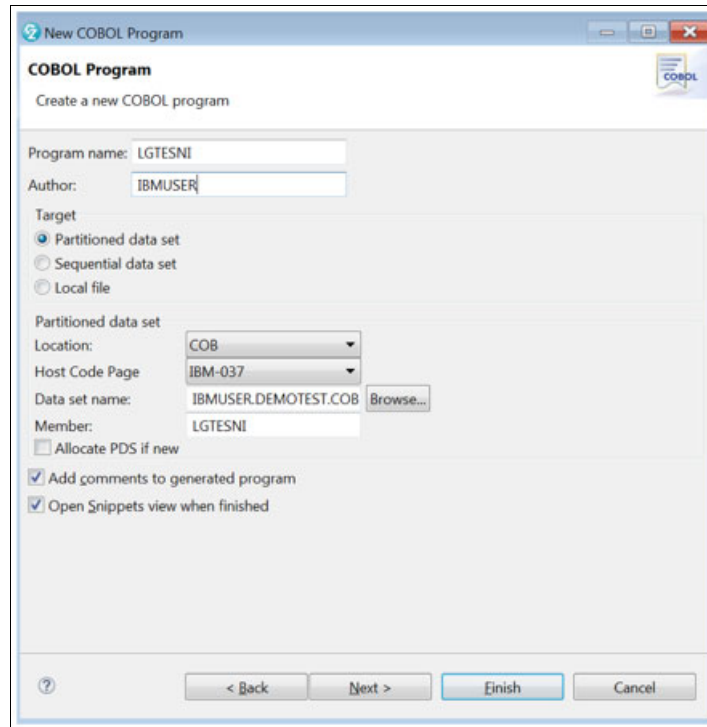


Figure 17 Creating a program by using templates

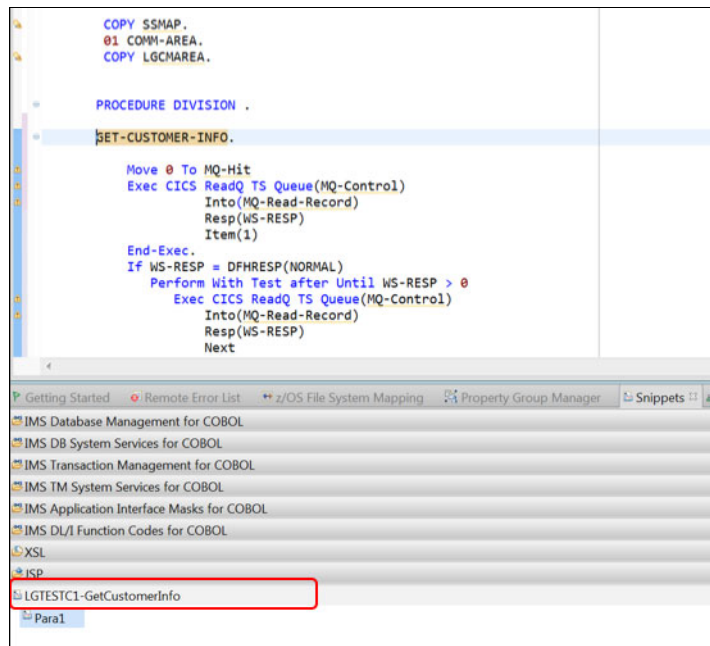


Figure 18 Adding snippets that were captured from AD into the new code

- ▶ The newly created programs and the modified programs can be easily made available for testing and deployment (see Figure 19) by using one of the following methods:
 - Filtering out unused variables that are identified by the zero Reference counts in the Data Elements Report
 - The Generate, compile, and link JCL feature to quickly build programs
 - Software Analyzer feature to perform coding standards reviews

Name	Level	References
WS-RESP	1	5
READ-MSG	1	1
READ-CUST-HIGH	3	0
F24	77	0
COMM-AREA	1	0
WS-ITEM-COUNT	1	0
WRITE-MSG	1	0
STSQ-NAME	3	0
WS-CUST-HIGH	1	0
WRITE-MSG-LOW	3	0
MSGEND	77	0

Figure 19 Filtering out and removing unused variables

By using the ADDI and IDz integration, refactoring code becomes much easier. The refactored code and its interface copybook can then be used to create APIs.

Creating an API by using IBM z/OS Connect Enterprise Edition

IBM z/OS Connect Enterprise Edition provides a framework that enables z/OS based programs and data to participate fully in the new API economy for mobile and cloud applications. The framework provides concurrent access by using a common interface to multiple z/OS subsystems.

z/OS Connect EE includes Eclipse tools that help developers create APIs from interface structures, such as the copybook of a CICS program as an input. z/OS Connect EE runs in the IBM WebSphere® Liberty z/OS environment.

Creating an API from a CICS program that is started by using COMMAREA

z/OS Connect Enterprise Edition includes the following features:

- Provides a mainframe utility BAQLS2JS to create Service Archive files (also known as SAR files) from the COBOL copybook, as shown in Figure 20.

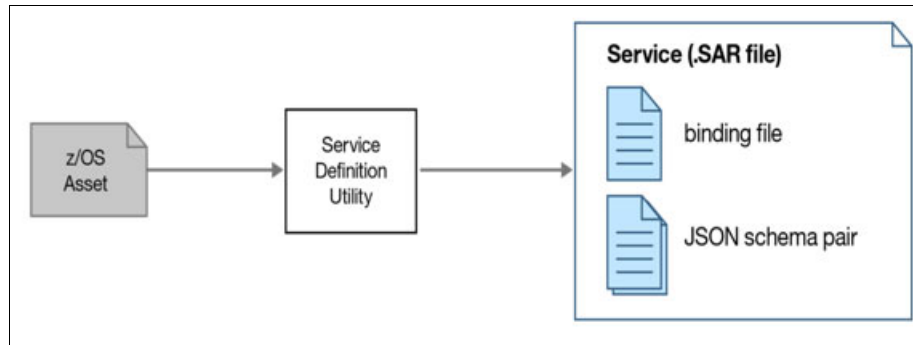


Figure 20 Creating artifacts for a SAR file

Note: Bind files are binary format files that contain information about the present data fields and how the JSON fields relate to the COMMAREA fields. IBM z/OS Connect Enterprise Edition reads these bind files and uses the information that is contained in them to perform the data conversion between JSON and COMMAREA.

JSON schema provides information about how the JSON is to be formatted to match the fields in the COMMAREA.

SAR files are compressed-format files that are produced by the `baqls2js` utility that contain JSON schema files and metadata that is related to the service. These files are provided as the input to the IBM z/OS Connect Enterprise Edition API Editor for it to understand the service and data fields.

- ▶ Its own wizard-based tools can be plugged into IDz. The tools include API editors that help to select the API methods, as shown in Figure 21.

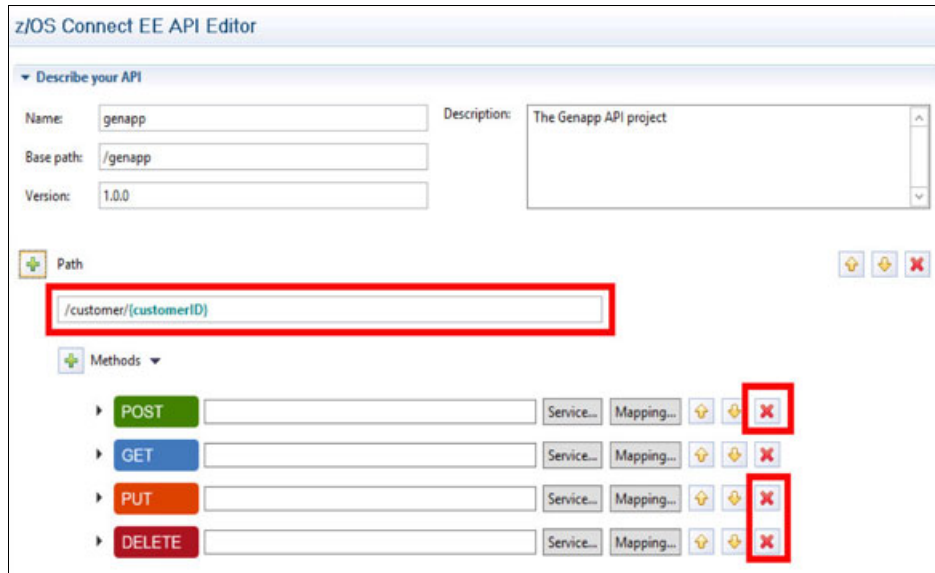


Figure 21 IBM z/OS Connect Enterprise Edition API Editor

- ▶ JSON structures are available to copybook variables (see Figure 21).

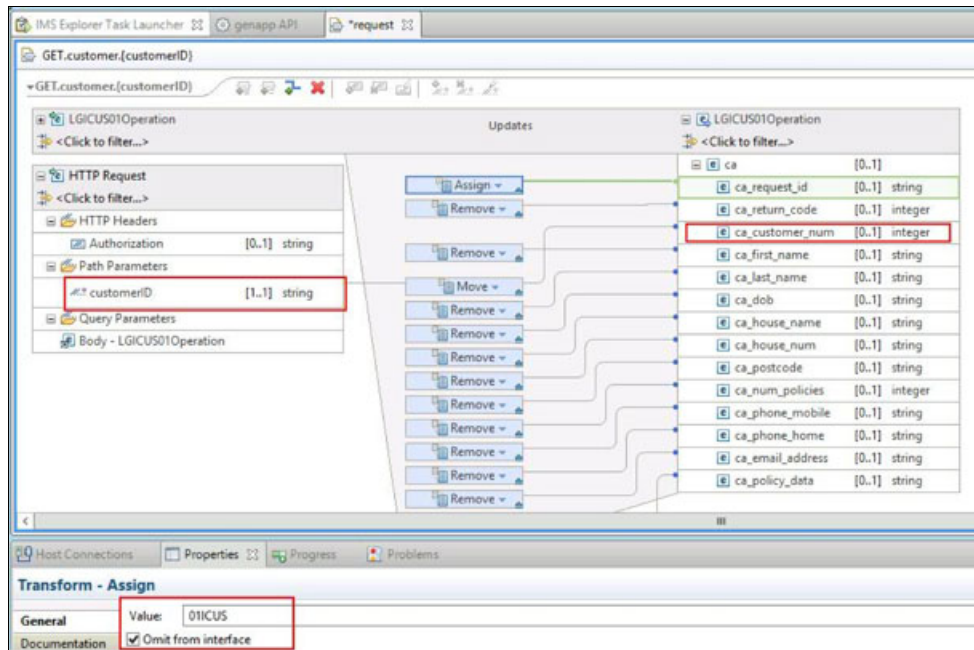


Figure 22 Mapping the API JSON schema and the COBOL copybook

- ▶ Because z/OS Connect Enterprise Edition runs in Liberty, artifacts that are created (such as schema mapping bind files) can be easily configured in .xml files, as shown in Figure 23.

```

<zoscconnect_zosConnectService
  id="InquireCustomer"
  invokeURI="/InquireCustomer"
  serviceName="InquireCustomer"
  dataXformRef="xformJSON2Byte"
  serviceDescription="Get customer details"
  serviceRef="wolaGA" />

<zoscconnect_localAdaptersConnectService
  id="wolaGA"
  registerName="ZCONNREG"
  serviceName="LGICUS01"
  connectionFactoryRef="wolaCF" />

<zoscconnect_zosConnectDataXform
  id="xformJSON2Byte"
  bindFileLoc="/var/zoscconnect/servers/server1/dataXform/bind"
  bindFileSuffix=".wsbind"
  requestSchemaLoc="/var/zoscconnect/servers/server1/dataXform/json"
  responseSchemaLoc="/var/zoscconnect/servers/server1/dataXform/json"
  requestSchemaSuffix=".json" responseSchemaSuffix=".json" />

```

Figure 23 Example of a Liberty server.xml file configuration for an API InquireCustomer

The tools also provide options to easily deploy the application into CICS (see Figure 24).

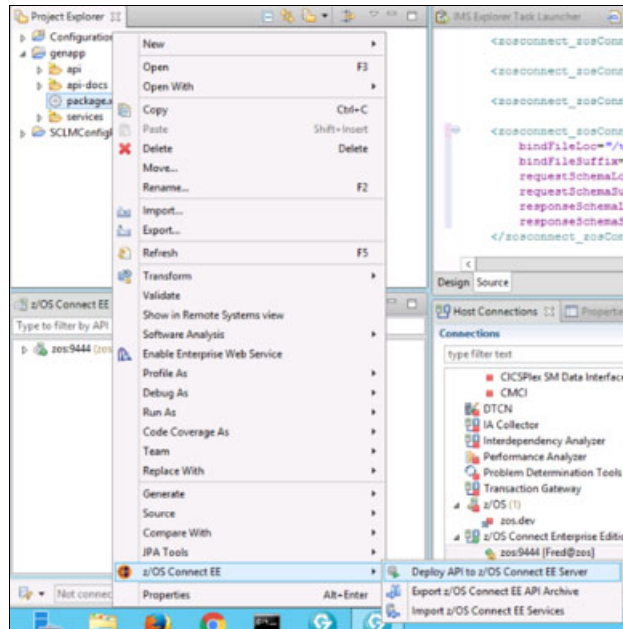
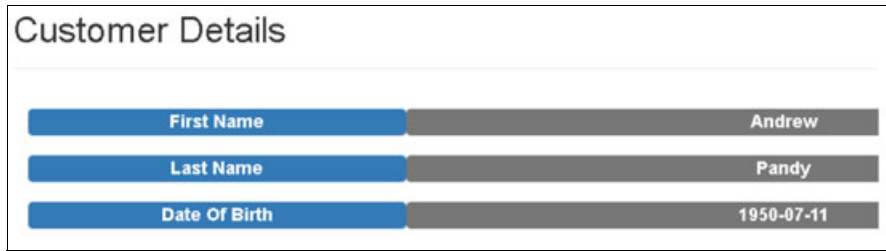


Figure 24 Deploying an API from the IBM z/OS Connect Enterprise Edition tools


The Customer Details web page (see Figure 25) shows the same functionality as the CICS screen (see Figure 26).



Customer Details

First Name	Andrew
Last Name	Pandy
Date Of Birth	1950-07-11

Figure 25 Customer details web page



```
SSC1      General Insurance Customer Menu

  1. Cust Inquiry      Cust Number      0000000001
  2. Cust Add          Cust Name :First  ANDREW
                      :Last   PANDY
  4. Cust Update       DOB             1950-07-11  (yyyy-mm-dd)
                      House Name
                      House Number  34
                      Postcode      PI1010
                      Phone: Home
                      Phone: Mob    (44) 0754 6745433
                      Email Addr

Select Option  1
```

1A B 04/051

Figure 26 CICS screen

Summary

This paper described how you can easily understand your mainframe assets, refactor them by using IBM Developer for z Systems, and make them available to new uses through APIs by using z/OS connect Enterprise Edition in Application Discovery.

By using the latest DevOps practices with a modern pipeline, including IBM Rational® Team Concert™, IBM UrbanCode® Deploy, and Rational Test Workbench, you now have the tools to find and make available the APIs to easily test any refactoring that is done. This testing ensures that the applications continue to function as expected while the service interface is added.

Additional resource

For more information about the topics that are covered in this Redpaper publication, see the [IBM z Systems Trial Program page](#) of the IBM IT infrastructure website.

Author

This paper was produced by a team of specialists from around the world.

Suman Gopinath is a Solution Architect for DevOps for IBM zEnterprise® Systems in India. She has 15 years of experience in the mainframe space, specializing in automation and DevOps on mainframe applications for the last six years. She holds a degree in Bachelor of Technology from Cochin University of Science and Technology, India. Her areas of expertise include the development, design, and architecture of mainframe applications; DevOps tooling (particularly Application Discovery and Delivery Intelligence); IBM Developer for z Systems; Urbancode Deploy on IBM Z, and integrating IBM products with Open Source tools.

Thanks to the following people for their contributions to this project, valuable inputs, and diligent review:

- ▶ Madhu B A
IBM India
- ▶ Russell Bonner
IBM UK
- ▶ Paul Pilotto
IBM Belgium
- ▶ Rosalind Radcliffe
IBM US

Thanks also to the following people:

- ▶ Mark Indermaur and Teresa Ge for discussions and inputs about finalizing this IBM Redpaper publication topic.
- ▶ Larry Hart and Ambal S Balakrishnan for their reviews, coordination, and helping to see this project to its completion.

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new IBM Redbooks® publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.


COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

CICS®	Redbooks®	z Systems®
Concert™	Redpaper™	z/OS®
DB2®	Redbooks (logo)  ®	zEnterprise®
IBM®	UrbanCode®	
Rational®	WebSphere®	

The following terms are trademarks of other companies:

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other company, product, or service names may be trademarks or service marks of others.



REDP-5452-00

ISBN 0738456152

Printed in U.S.A.

Get connected

