

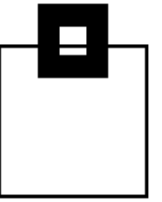


The zGui (r)evolution

—

First hands on experience and best practices

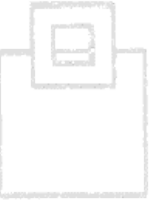
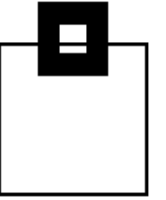
Ulf Heinrich  
SEGUS Inc



# Agenda

---

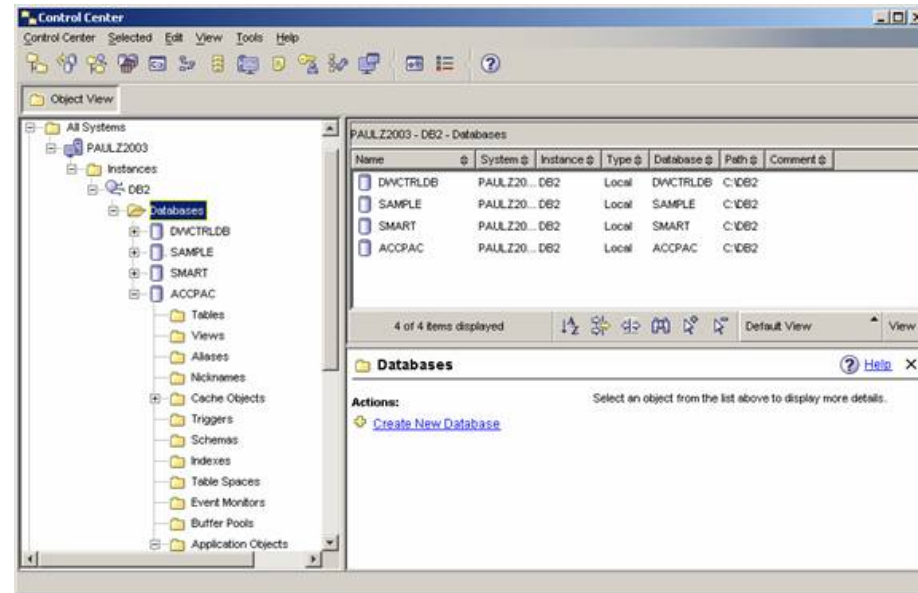
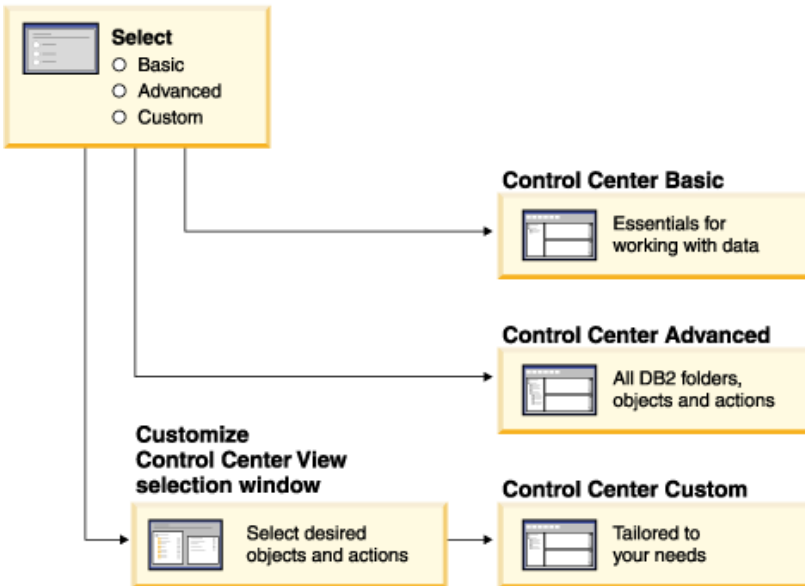
- GUIs in the past
- Zowe ecosystem overview
- Zowe differentiation to prior GUIs
- Zowe components
- Zowe examples
- Hands on usage based on a cloning example
- Summary of experience



# GUIs in the past

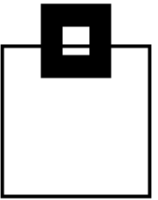
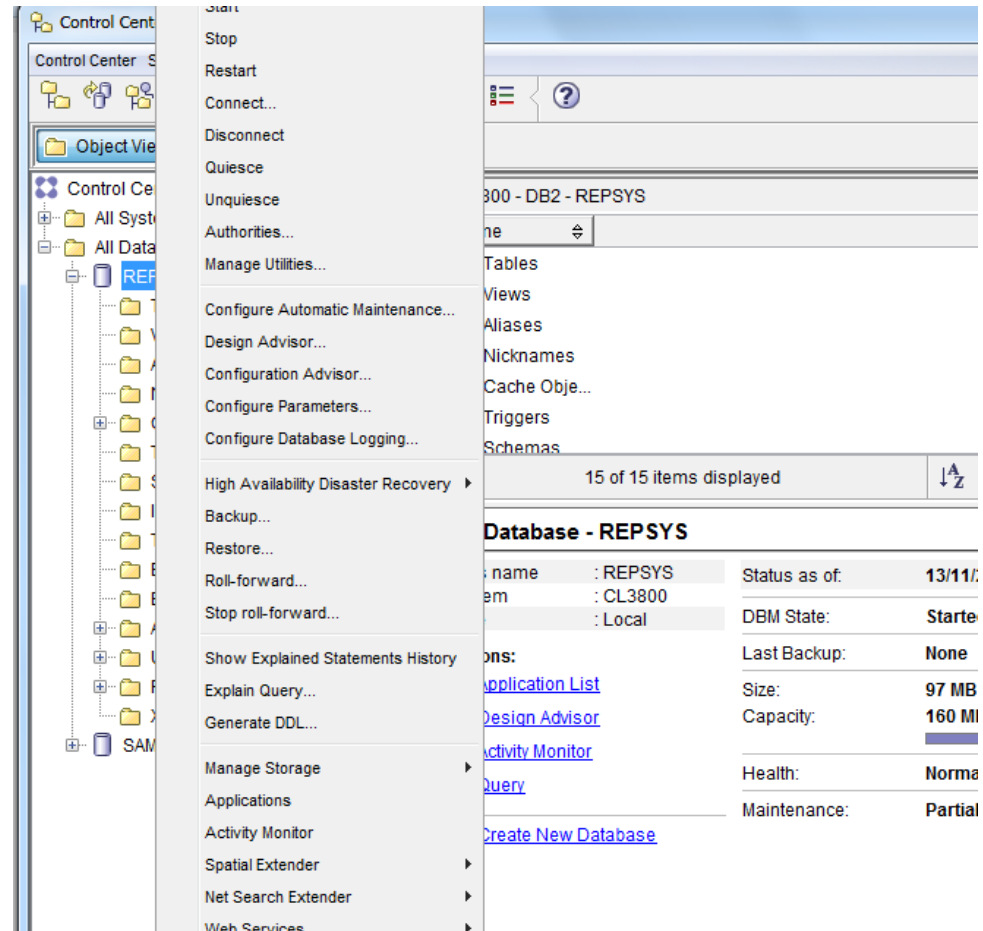
- Db2 Control Center (Db2cc)
  - Introduced with Db2 LUW 5, but also able to connect to Db2 z/OS
  - A Windows/Linux fat client using Db2 connect and stored procedures
  - Manages and administers Db2 systems and objects

Control Center View selection window



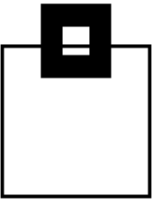
# GUIs in the past

- Db2 Control Center (Db2cc)
  - Can also open other centers to
    - optimize queries, jobs, and scripts
    - perform data warehousing tasks
    - create stored procedures
    - work with DB2 and IMS commands

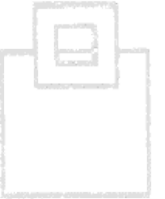


# GUIs in the past

---

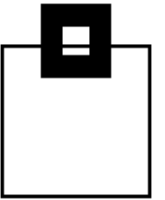


- Db2 Control Center (Db2cc)
  - More and more features and functions added over time:
    - Activity Monitor
    - Command Editor
    - Configuration Assistant
    - Control Center and associated wizards and advisors
    - Control Center plug-in extensions
    - Event Analyzer
    - Health Center
    - Indoubt Transaction Monitor
    - Journal
    - License Center
    - Memory Visualizer
    - Query Patroller Center
    - Satellite Administration Center
    - Task Center
    - User interface to access Spatial Extender functionality
    - User interface to Visual Explain



# GUIs in the past

---

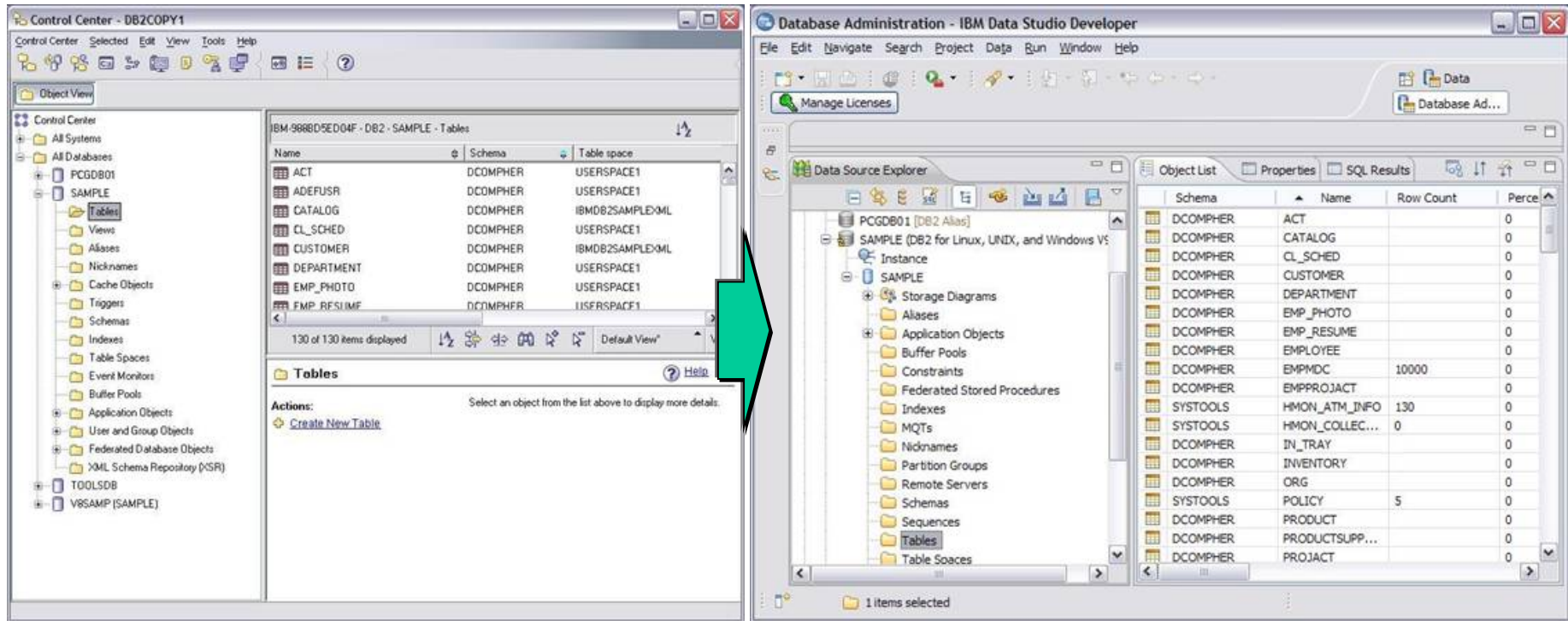


- Db2 Control Center (Db2cc)
  - ...along with wizards and advisors:
    - Control Center and associated wizards and advisors
      - Alter Database Partition Group wizard
      - Backup wizard
      - Configuration advisor
      - Configure Database Logging wizard
      - Configure Multisite Update wizard
      - Create Cache Table wizard
      - Create Database wizard
      - Create Federated Objects wizard (Also known as Create Nicknames wizard)
      - Create Table Space wizard
      - Create Table wizard
      - Design advisor
      - Drop Partition launchpad
      - Health Alert Notification
      - Health Indicator Configuration launchpad
      - Load wizard
      - Recommendation advisor
      - Redistribute Data wizard
      - Restore wizard
      - Set Up Activity Monitor wizard
      - Set Up High Availability Disaster Recovery (HADR) Databases wizard
      - Storage Management Setup launchpad
      - Troubleshooting wizard

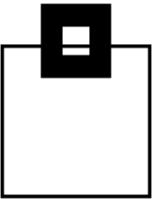


# GUIs in the past

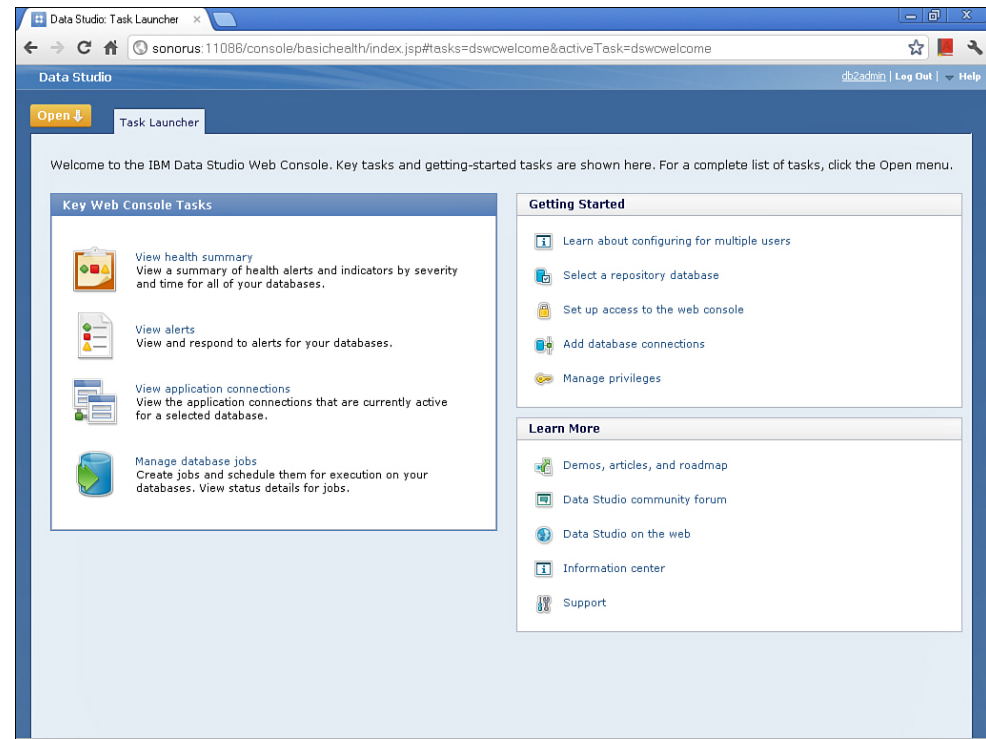
- Db2 Control Center (Db2cc)
  - Deprecated with Db2 LUW 9.7 and Db2 z/OS 10.1
  - Db2cc successor: Data Studio



# GUIs in the past



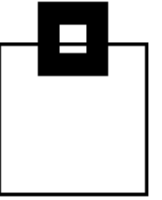
- Db2 Data Studio (Db2DS)
  - A Windows/Linux EclipsePlugin using Java Db2 connection
  
- Db2 Data Studio Web Console (Db2DSWC)
  - A Client/Server architecture, that enables web browser access





# GUIs in the past

---

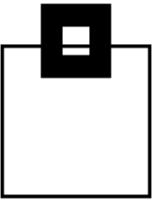


- Db2cc successor: Data Studio
  - True for most of the Db2cc tools, except:
    - Activity Monitor, Event Analyzer, Health Center, Web Console, Memory Visualizer, Query Patroller Center  
→ InfoSphere Optim Performance Manager
    - Configuration Assistant  
→ InfoSphere Optim Configuration Manager
  - With more complex licensing associated:
    - InfoSphere Optim Performance Manager Extended Insight is a separately priced feature for InfoSphere Optim Performance Manager (part of InfoSphere Optim Performance Manager EE)
    - Data Studio consist of three components
    - The Index Advisor and Query Advisor require an InfoSphere Optim Query Workload Tuner license
    - Db2 Data Studio (Db2DS) renamed and bundled into Optim in 2009

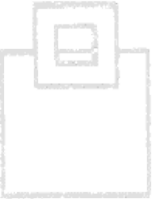


## GUIs in the past

---



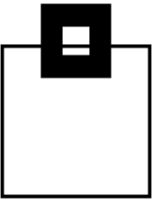
- Then Db2 Data Server Manager was introduced\* and customers were confused whether this is a DS successor/replacement
  - Some IBMers said yes, some insisted they address different people:
    - DS is intended for developers
    - DSM is intended for DBAs
  - Unfortunately some DS features are not maintained with Db2 12 CD
- Digging deeper indicates lots of the prior GUI Eclipse stuff and components "borrowed" from Db2DSWC
- However, the labs are saying it is "very much a rewrite of the front end, but the smarts have been passed onto this next generation"



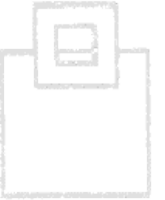
\*in July 2010 also z/OS Management Facility for system programmers

# GUIs in the past

---

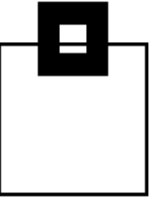


- Bottom line/downside for ISVPs and customers:
  - Familiar UIs continue to be changed
  - Used features deprecated, or slightly shifted into other UIs
  - No single/common point of control
    - ISPF still the one and only true (Db2) z/OS UI that stays reliably solid over the years
    - ISPF still the one and only true (Db2) z/OS UI that is supported by IBM AND ISVs



# Zowe ecosystem overview

---

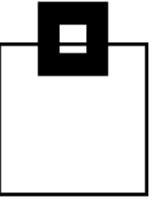


- At the SHARE 2018 conference, IBM, Rocket Software and CA Technologies (now BROADCOM) announced Zowe – THE z ecosystem
  - Open source project licensed under EPL 2.0
  - Extensible framework
  - Fuses and unites „old“, solid mainframe UI (tn3270, VT) with latest UI (HTML5, JS, TS, CLI)
  - Based on and exploiting proven, rock solid technology (RLF, SAF, USS)
  - Introduces REST APIs, ESM microservices, discovery services, ...
- Addresses
  - Application Developers
  - System Programmers
  - DBAs
  - DevOps Architects



# Zowe ecosystem overview

---



- Zowe is four major components:

1. Application Framework

The web UI that works with the underlying REST APIs presenting and bundling information in a modern, powerful full screen mode

2. z/OS Services

Providing z/OS RESTful web service and deployment architecture for z/OS microservices

3. Zowe CLI

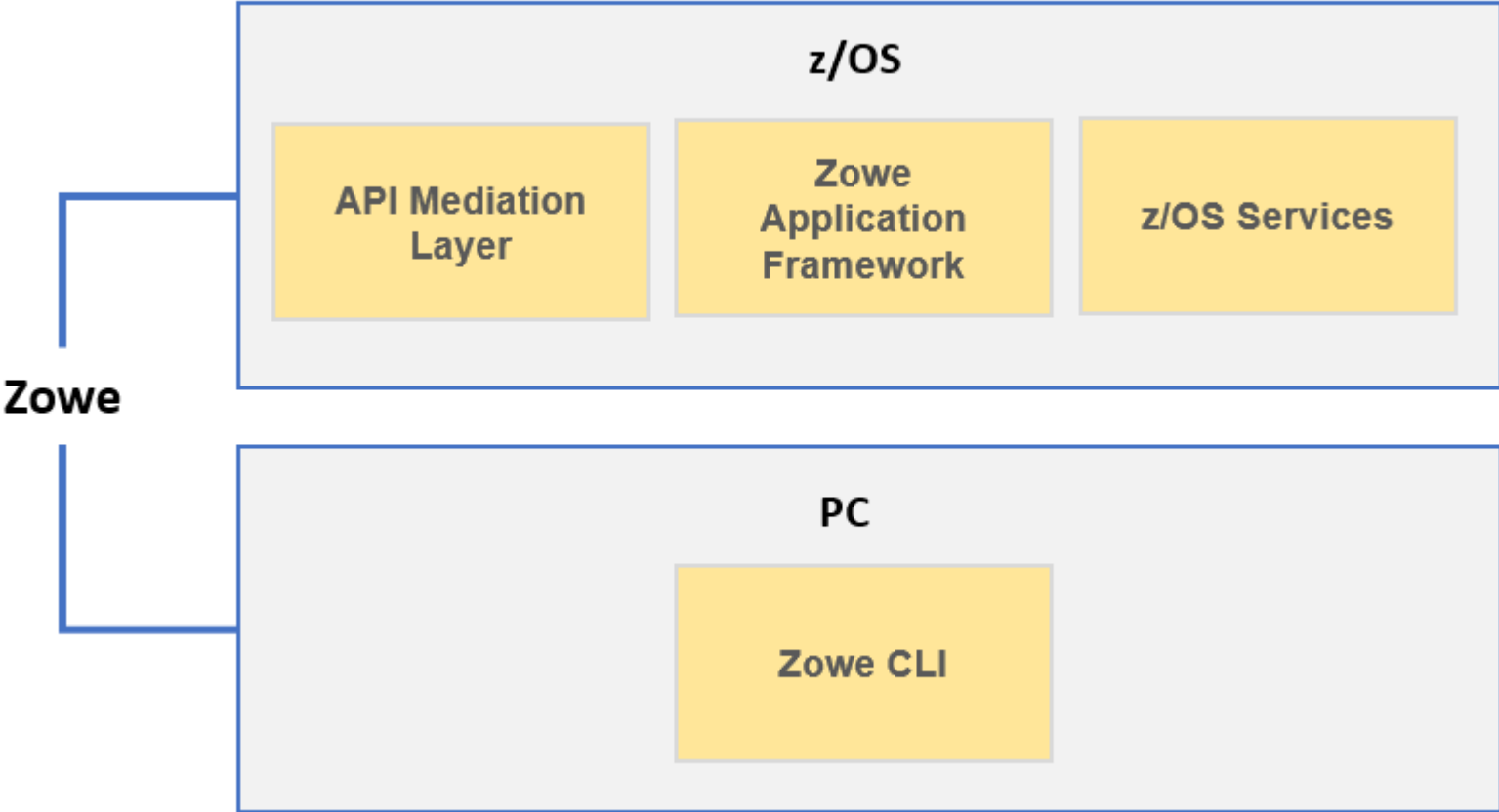
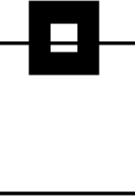
Allowing to interact with the mainframe to efficiently build z/OS applications

4. API Mediation Layer

Central point for all mainframe service REST APIs of the ecosystem

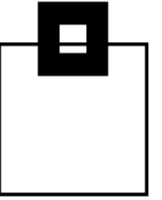


# Zowe ecosystem overview



# Zowe differentiation to prior GUIs

---



## Zowe is

- the very first open source project on z/OS
- an extensible, common framework for existing and new applications
- designed to make the mainframe an agile, integrated platform
- *a* **THE** common UI for senior mainframe staff and new workforce
- a unified framework that merges proven and latest technology



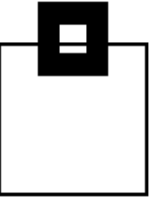
## ...to

- demystify the mainframe and attract new people
- reduce the learning curve and improve productivity
- enhance integration and consumability
- simplify the architecture and reduce operational costs
- improve co-existence with a modern, platform-neutral interface



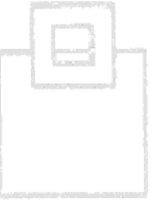
# Zowe differentiation to prior GUIs

---



Zowe is vendor independent:

- Open source project under the Open Mainframe Project
- Free to be used under the Eclipse Public License 2.0
- Open, extensible interfaces of the code
- IBM, Rocket and BROADCOM (fka. CA) are founding members



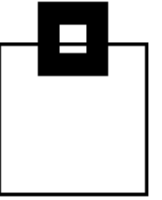
→ Use, change and contribute





# Zowe differentiation to prior GUIs

---

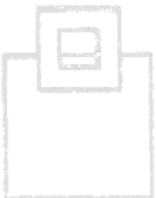
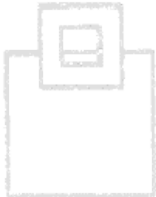
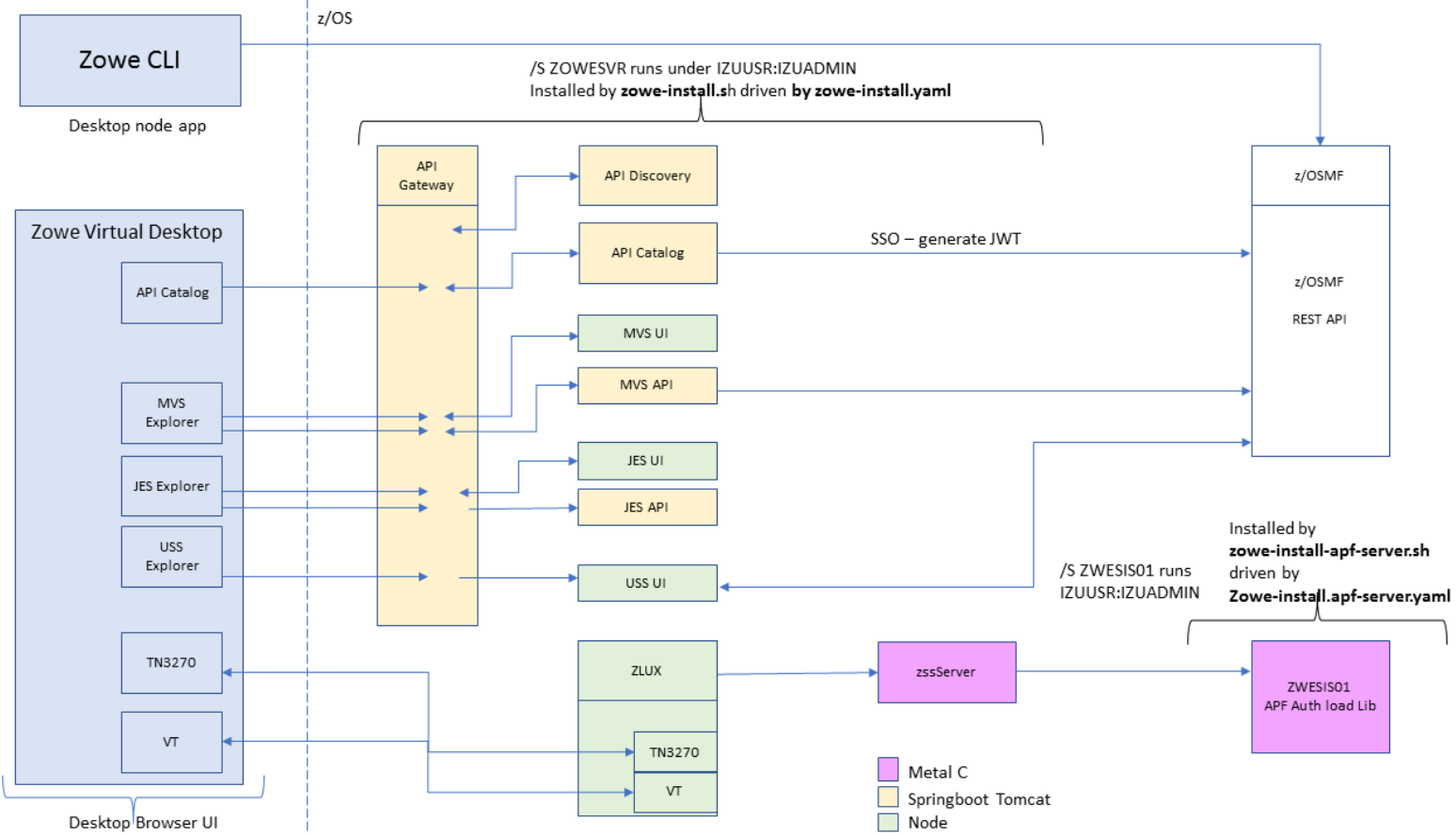
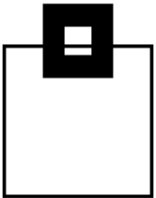


Zowe integrates nicely into an existing environment:

- Security management: SAF – System Authorization Facility
  - Controlling access by RACF, or other security products, like ACF2
- Resource management: RLF – Resource Limit Facility
  - Control processor usage of Db2 queries
- z/OS and USS support:
  - Explore JES, MVS, USS
  - Access and interact with subsystems like Db2, CICS
  - Browse and edit data sets
  - Execute JCL, Shell and z/OS commands, bash and z/OS scripts
- Platform independent browser technology:
  - HTML5, CSS, JS, TS, ...
- Platform independent CLI
  - Node.js, npm, IDEs, Jenkins, TravisCI, ...

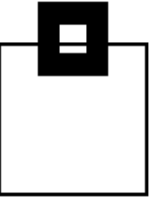


# Zowe components



# Zowe components

---

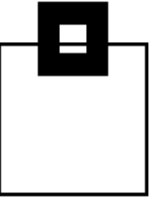


- Zowe Application framework is four major components:
  1. Desktop  
Browser based web desktop
  2. Application Server  
Web services framework plus proxy applications that communicates with z/OS services and components
  3. ZSS Server  
REST services to support the Application Server
  4. Application plug-ins  
Included and addable applications to access the mainframe and to perform various tasks, e.g.
    - Dataset editor and browser (z/OS and USS)
    - Workflows
    - z/OS subsystem browser (JES, CICS, Db2, IMS, ...)
    - ...



# Zowe components

---

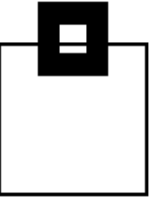


- Zowe z/OS services contain the following core components:
  1. z/OS dataset services  
list, browse, edit, create, delete, ... datasets and members
  2. z/OS job services  
list, browse, submit jobs
- A full list of capabilities of the RESTful API can be listed via the API catalog
  - The Open API Specification describes the APIs and allows to use any standard-based REST API developer tool, or API management process
  - APIs can be used by any application
  - z/OS services are running as microservices with a Spring Boot embedded Tomcat stack

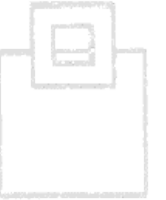


# Zowe components

---

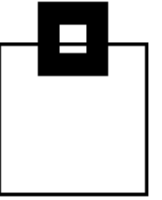


- Zowe CLI comes with the following capabilities:
  - Interact with files:
    - Create, edit, download, and upload data sets
  - Submit jobs:
    - Submit JCL from data sets or local storage, monitor the status, and view/download the output
  - Execute commands:
    - Issue TSO, or z/OS console commands
  - Integrated scripts:
    - Define scripts that do both mainframe and local tasks
  - Return JSON documents:
    - Return the data in JSON format to be used in other programming languages



# Zowe components

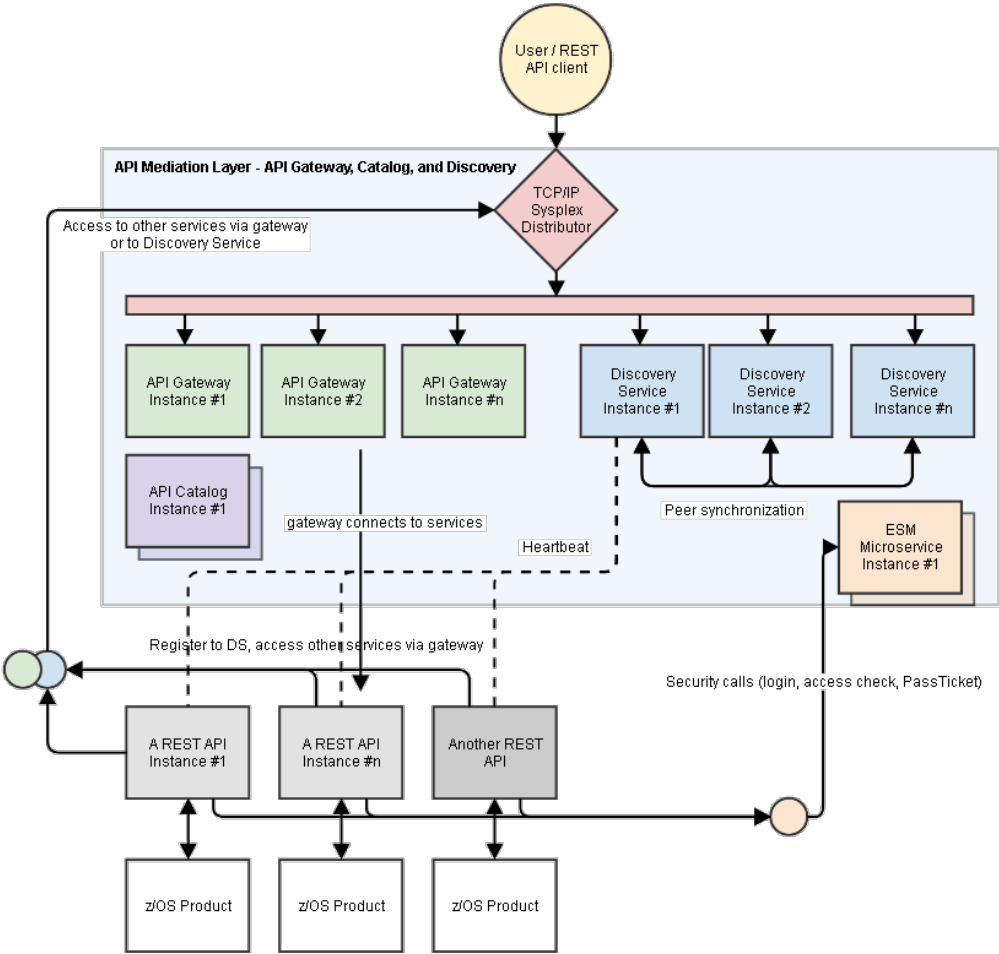
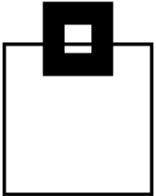
---



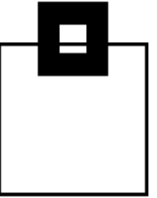
- Zowe API mediation layer consists of the following components:
  - API gateway:
    - Clients interact with microservices behind a reverse proxy forwarding requests to the appropriate service
    - The gateway is built on Netflix Zuul and Spring Boot technology
  - Discovery services:
    - Accepts the REST service announcements and serves active ones
    - The service is built on Netflix Eureka and Spring Boot technology
  - API catalog:
    - UI catalog of published APIs along with their documentation (Swagger) and status
    - Services can be implemented by multiple instances for high-availability or scalability
  - ESM microservices:
    - Authenticates and authorizes users with mainframe credentials



# Zowe components



# Zowe components @ github.com

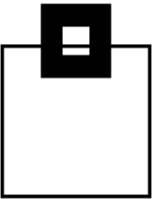
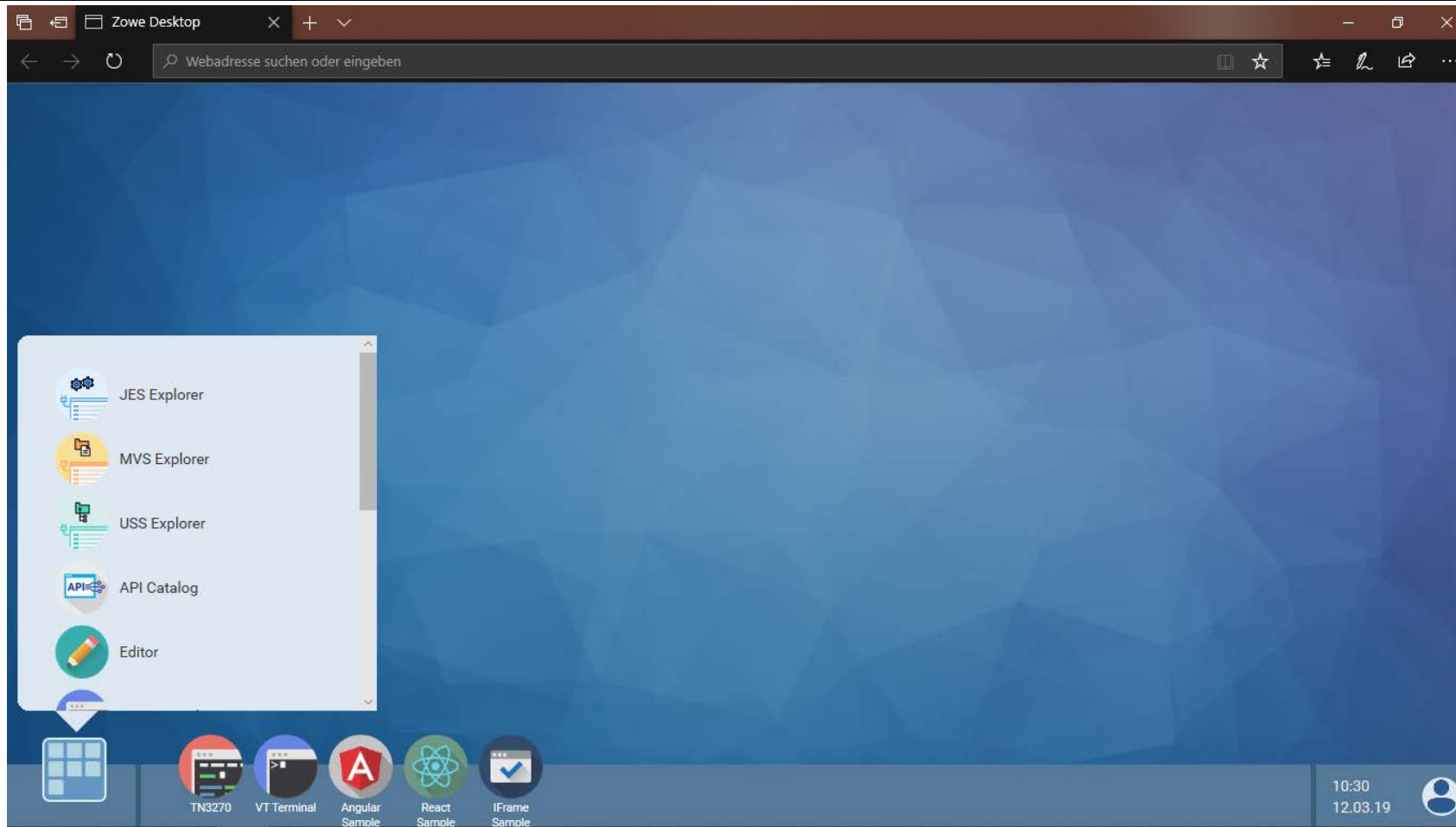


- zowe-cli - Zowe CLI
- ztrial-scenarios - This repo tracks the zTrial scenarios for Zowe.
- zowe-common-c - C Libraries for various OS & Networking needs
- zlux-app-server - A collection of build, deploy, and run scripts & configuration files for running a simple zLUX server.
- zlux - The top-level superproject for zLUX. zLUX includes the Zowe Desktop framework in addition to several built-in apps and an example server implementation.
- docs-site - Documentation for the Zowe project
- community - Community Engagement - Contribution Guidelines, Meeting Minutes, and more
- zowe-cli-db2-plugin - DB2 Plugin for the Zowe CLI
- zowe-cli-cics-plugin - CICS Plugin for the Zowe CLI
- zowe-cli-sample-plugin - Plugin Tutorial for Zowe CLI
- perf-timing - Performance tests
- api-layer - Zowe API Mediation Layer
- sample-trial-app
- zowe-install-packaging - Packaging repository for the Zowe install scripts and files
- imperative - imperative CLI Framework
- vscode-extension-for-zowe - Visual Studio Code Plug-in for Zowe, which lets users interact with z/OS data sets on a remote mainframe instance. Powered by Zowe CLI
- cpu\_usage\_sample - An example of a Spring Boot application
- zowe-install-test - Perform Zowe installation and smoke test
- zlux-server-framework - Contains essential zLUX proxy server components including SSO and service catalogs
- ztrial-sample-cli-plugin
- zlux-build - Repository for common build scripts among various superprojects
- explorer-jes-rt - Functional tests for JES explorer
- explorer-jes
- explorer-mvs
- explorer-uss
- explorer-ui-server - Simple HTTPS web server, used by explorer UI plugins
- data-sets - Repo for the springboot based data set APIs
- jobs - Repo for the jobs api controller and code
- explorer-api-common - common repo for explorer api projects
- zlux-app-manager - zLUX Framework components for management of zLUX Apps. Used for window managers or web layouts.
- zlc - Zowe Leadership Committee collaboration
- vt-ng2 - A simple USS/Unix/VT terminal emulator written in Angular and Javascript
- trs2-ng2 - A TRS29 emulator written in Angular and Javascript
- zss - Zowe Secure Services Server for enabling low-level microservices
- zlux-ng2 - Angular Hosting Environment for the zLUX Framework's web components
- zss-auth - Auth handler for App server to connect to ZSS through standard ZSS login
- db-browser - A database viewer and editor for working with a variety of databases within the Zowe Desktop
- db-browser-db2 - db2 module for db-browser App for Zowe
- jupyter-app - A Zowe App for displaying Jupyter
- zos-subsystems - An example app showing z/OS infrastructure
- workshop-starter-app - An App to provide at the start of a workshop session to showcase Zowe App development & App-to-App communication
- file-transfer-app - An App for transferring files to and from a mainframe
- zosmf-auth - Auth handler for App server to connect to z/OSMF through standard z/OSMF login
- zlux-workflow
- zlux-shared - zLUX framework components that are utilized both by the server and in the web browser
- zlux-platform
- zlux-editor - A simple editor in a browser
- sample-react-app - Sample to showcase a react app that natively can be presented into the Zowe desktop
- sample-iframe-app
- sample-angular-app
- spring-boot-jzos-sample - An example of a Spring Boot sample to be statically linked into the API Gateway
- zowe-promote-publish - Zowe Pipeline to Promote and Publish a PAX Candidate
- release-management - Material and activities related to release management
- zowe-cli-standalone-packages - Jenkins pipeline which generates a Zowe CLI ZIP containing the base CLI and Zowe plugins.
- sample-node-api - A sample node.js api for finding cars and accounts for a dealership
- sample-trial-react-app - Sample React App
- zowe-cli-version-controller - Main controller and maintainer of the versioning scheme
- zlux-grid
- jenkins-slave-images
- zlux-file-explorer
- orion-editor-component
- zlux-widgets
- zlux-file-properties
- explorer-server-tests
- explorer-server - Explorer Server component contribution
- workshop-user-browser-app - Starter files & a tutorial README to get started on building a simple Zowe App
- explorer-server-auth
- taskmanager - Shows running services / processes on the z/OS Sysplex Served by Zowe
- zowe.github.io - Testing Github Pages for Community Website as an Alternative to Wordpress
- zowe-cli-sample-scripts - Demo scripts for the Zowe CLI
- Onboarding-scripts - Template scripts for extenders to onboard their products with
- explorer-utilities - Explorer shared utilities project
- zowe-cli-profile-migration - Zowe CLI Profile Migration Tool
- docs-site-temp
- explorer-injector
- webui-scenarios - Several sample projects that create WebUI's that integrate into Zowe
- explorer-model - The Explorer server model project

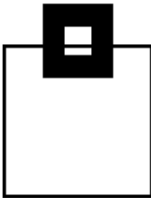




# Zowe examples – the Zowe desktop



# Zowe examples – the tn3270 app 😊



```

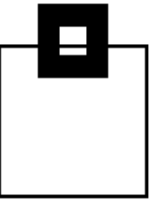
Zowe Desktop
Webadresse suchen oder eingeben
TN3270
Menu Utilities Compilers options Status Help
ISPF Primary Option Menu
Option ==>
0 Settings Terminal and user parameters XSE
1 View Display source data or listings SDSF
2 Edit Create or change source data Multi stack
3 Utilities Perform utility functions =3.4
4 Foreground Interactive language processing =3.12
5 Batch Submit job for language processing =3.14
6 Command Enter TSO or workstation commands =6
7 Dialog Test Perform dialog testing
9 IBM Products IBM program development products
10 SCLM SW Configuration Library Manager
11 workplace ISPF Object/Action workplace
M More Additional IBM Products
S SE Software Engineering Menu

Enter X to Terminate using log/list defaults

MA A 4 /14
```



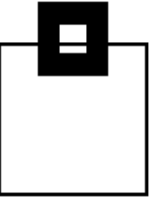
# Zowe examples – z/OS Subsystems



Type	Count	Explore
CICS	0	<a href="#">Explore</a>
DB2	41	<a href="#">Explore</a>
IMS	0	<a href="#">Explore</a>
MQ	0	<a href="#">Explore</a>



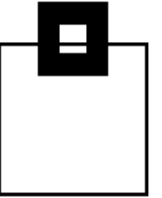
# Zowe examples – z/OS Subsystems



Name	Configuration
DBPR	1
VX2	1
VX1	1
VC9	1
VC8	1
VC7	1
QC8	1
QC7	1
CCA	1
BC8	1
BC7	1



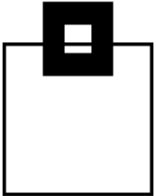
# Zowe examples – z/OS Subsystems



Is Active	Version	Release	Mod Level	Master Jobname	Master ASID	Domain Name	Location	Listener Port	Secure Port
true	12	1		AC7MSTR	163	s0w1.fritz.box	Z100AC7	5129	0

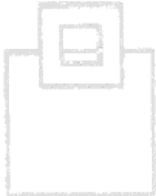


# Zowe examples – the JES Explorer

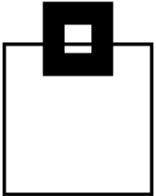


The screenshot shows the JES Explorer interface. On the left, there is a sidebar with 'Job Filters' and a list of jobs: 'ZOWESVR:STC06153 [ACTIVE]', 'JESMSGLG', 'JESJCL', 'JESYSMSG', 'STDOUT', and 'STDERR'. The main area displays the job log for 'ZOWESVR - STC06153 - JESJCL'. The log contains the following text:

```
1 //ZOWESVR JOB MSGLEVEL=1 STC06153
2 //STARTING EXEC ZOWESVR
3
4 XX* This program and the accompanying materials are made available *
5 XX* under the terms of the Eclipse Public License v2.0 which *
6 XX* accompanies this distribution, and is available at *
7 XX* https://www.eclipse.org/legal/epl-v2.0.html *
8 XX* *
9 XX* SPDX-License-Identifier: EPL-2.0 *
10 XX* *
11 XX* Copyright IBM Corporation 2018 *
12 XX* *
13 XX* *
14 XX* ZOWE SERVER PROCEDURE *
15 XX* *
16 XX* This is a procedure to start the Zowe web server and Node server.*
17 XX* This procedure requires a WebSphere Liberty Angel procedure *
18 XX* to be running, such as z/OSMF procedure "IZUANG*". *
19 XX* *
20 XX* Invoke this procedure, specifying the path where the ZOWE server *
21 XX* is installed on your system. *
22 XX* *
23 XX* S ZOWESVR,SRVRPATH='/var/zowe/0.9.5/explorer-server' *
24 XX* *
25 XX* *
26 XX* *
27 3 XXZOWESVR PROC SRVRPATH='/var/zowe/0.9.5/explorer-server'
28 XX*-----
29 XX* SRVRPATH - The path to the HFS directory where the Atlas server
30 XX* was installed.
31 XX*-----
32 4 XXEXPORT EXPORT SYMLIST=*
33 XX*-----
34 XX* Start the node server
35 XX* Start the Zowe Atlas server
36 XX*-----
37 5 XXZOWESTEP EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
38 XX PARM='PGM /bin/sh &SRVRPATH/./scripts/internal/run-zowe.sh'
39 IEF653I SUBSTITUTION JCL - PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,PARM='PGM /bin/sh
40 /var/zowe/0.9.5/explorer-server/./scripts/internal/run-zowe.sh'
41 6 XXSTDOUT DD SYSOUT=*
42
```



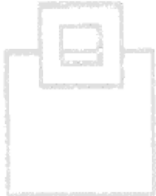
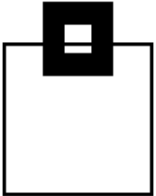
# Zowe examples – the MVS Explorer



The screenshot shows the Zowe Desktop application with a browser window at the top displaying 'Zowe Desktop' and a search bar. Below the browser is the 'MVS Explorer' window. On the left is a file tree for user 'HEINRIC' containing various files like 'HEINRIC.HEINRIC.SPUFI.IN', 'EMPDIR', 'RECIPES', and 'HEINRIC.JOB.CNTL'. The main area shows the JCL code for 'HEINRIC.JOB.CNTL(#AUDIT)'. The code includes comments for creating a daily audit report and a SQL SELECT statement with joins and filters.

```
1 //AUDITRPT JOB (),SE,CLASS=A,MSGCLASS=X
2 /**
3 /** CREATE DAILY AUDIT REPORT
4 /**
5 //S1 EXEC PGM=IKJEFT01,DYNAMNBR=20
6 //STEPLIB DD DISP=SHR,DSN=DSNA12.SDSNEXIT.QA1B
7 // DD DISP=SHR,DSN=DSNA12.SDSNLOAD
8 //SYSTSPT DD SYSOUT=*
9 //SYSPRINT DD SYSOUT=*
10 //SYSTSIN DD *
11 DSN SYSTEM(QA1B)
12 RUN PROGRAM(DSNTIAD) PLAN(DSNTIAD) +
13 LIB('DSNA12.RUNLIB.LOAD')
14 END
15 //SYSIN DD *
16 SELECT
17 T1.TRANSACTION , T1.END_USERID , T1.WORKSTATION , T1.PRIM_AUTHOR
18 AS
19 "PRIMARY_AUTHORIZATION_ID" , T1.PROGRAM , T1.PACKAGE_COLLID ,
20 T1.STMT_TIMESTAMP , T1.STMT_STATS_UPD,
21 T1.STMT_GROUP_SSID , T2.SQL_TEXT
22 FROM
23 IQA0610.WLXT001 T1
24 ,IQA0610.WLXT005 T2
25 WHERE ((
26 T1.WLX_TIMESTAMP >= '2014-12-01-12.30.08.135560')
27 AND (T2.WLX_TIMESTAMP = T1.WLX_TIMESTAMP)
28 AND (T2.STMT_GROUP_SSID = T1.STMT_GROUP_SSID)
29 AND (T2.STMT_ID = T1.STMT_ID)
30 AND (T2.STMT_ORIGIN = T1.STMT_ORIGIN)
31 AND (T2.STMT_TIMESTAMP = T1.STMT_TIMESTAMP)
32 AND (T2.STMT_TYPE = T1.STMT_TYPE)
33 );
34 COMMIT;
35
```

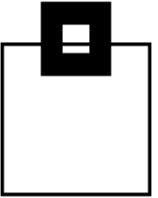
# Zowe examples – the USS Explorer



```
1 This program and the accompanying materials are
2 made available under the terms of the Eclipse Public License v2.0 which accompanies
3 this distribution, and is available at https://www.eclipse.org/legal/epl-v20.html
4
5 SPDX-License-Identifier: EPL-2.0
6
7 Copyright Contributors to the Zowe Project.
8 # zLUX
9
10 This is the project home of the zLUX portion of the Zowe Project. Within, there are several repositories and superprojects which help to cat
11
12 ##### This project is in active development and documentation may be brief at this time. Feel free to contribute to the Wiki and we'll keep t
13
14 If you want to get started with a Zowe for the first time, it is best to start by cloning this "zlux" repository. Contained within is a flat
15
16 New users should take a look at [zlux-example-server](https://github.com/zowe/zlux-example-server) to see how a simple zLUX server is assem
17
18 ##### Note: When using zlux-example-server or any Zowe server which needs to make use of the Rocket ZSS (Z Secure Services) Server , please r
19
20 This program and the accompanying materials are
21 made available under the terms of the Eclipse Public License v2.0 which accompanies
22 this distribution, and is available at https://www.eclipse.org/legal/epl-v20.html
23
24 SPDX-License-Identifier: EPL-2.0
25
26 Copyright Contributors to the Zowe Project.
27
```



# Zowe examples – the API Catalog

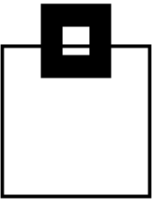


The screenshot shows a web browser window displaying the Zowe API Catalog. The browser's address bar contains the text "Webadresse suchen oder eingeben". The page title is "API Catalog". The interface features a blue header with the "API Catalog" logo and a power button icon. Below the header is a search bar with the placeholder text "Search for APIs". The main content area is titled "Available API services" and displays four service cards:

- API Mediation Layer API**: The API Mediation Layer for z/OS internal API services. The API Mediation Layer provides a single point of access to mainframe REST APIs and offers enterprise cloud-like feature...  
● All services are running
- z/OS Datasets services**: IBM z/OS Datasets REST services  
● All services are running
- z/OS Jobs services**: IBM z/OS Jobs REST services  
● All services are running
- z/OS Miscellaneous services**: IBM z/OS Miscellaneous REST services  
● All services are running



# Zowe examples – the API Catalog

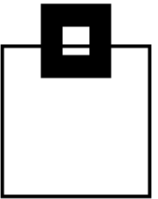


The screenshot displays a web browser window with the following content:

- Browser Tab:** REST API Documentation
- Address Bar:** Webadresse suchen oder eingeben
- Page Title:** API Catalog
- Header:** API Catalog (with a power button icon)
- Navigation:** < Back
- Main Section:** z/OS Jobs services  
IBM z/OS Jobs REST services
- Sub-section:** jobs
- Section Header:** IBM z/OS Jobs  
IBM z/OS Jobs REST API service
- Details:**
  - IBM z/OS Jobs**
  - API Version: 0.9.3
  - [ Base URL: 192.168.222.10:7554/api/v1/jobs ]
  - IBM z/OS Jobs REST API service
  - [External documentation](#)



# Zowe examples – the API Catalog



## Liberty REST APIs

Discover REST APIs available within Liberty

### API Discovery : APIs available from the API Discovery feature

Show/Hide | List Operations | Expand Operations

### Zowe : Dataset APIs

Show/Hide | List Operations | Expand Operations

### Zowe : JES Jobs APIs

Show/Hide | List Operations | Expand Operations

### Zowe : System APIs

Show/Hide | List Operations | Expand Operations

### Zowe : USS Files APIs

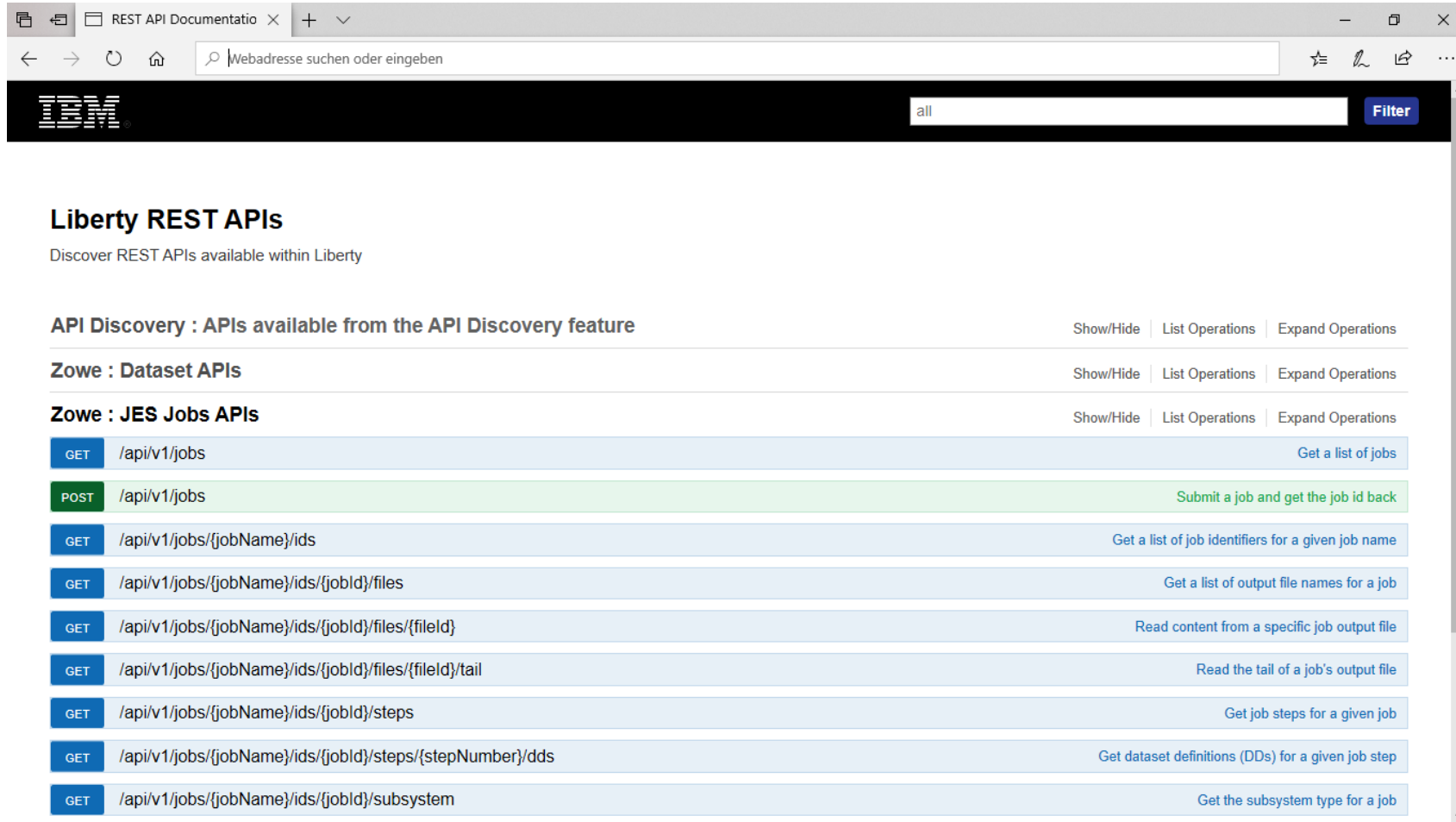
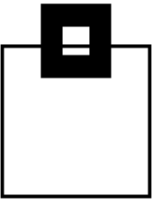
Show/Hide | List Operations | Expand Operations

### Zowe : zOS System APIs

Show/Hide | List Operations | Expand Operations



# Zowe examples – the API Catalog

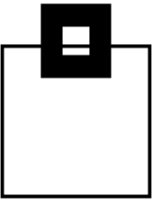


The screenshot shows a web browser window displaying the Zowe API Catalog. The browser's address bar contains the text "Webadresse suchen oder eingeben". The page header features the IBM logo on the left and a search bar with the text "all" and a "Filter" button on the right. The main content area is titled "Liberty REST APIs" and includes a subtitle "Discover REST APIs available within Liberty". Below this, there are three sections: "API Discovery : APIs available from the API Discovery feature", "Zowe : Dataset APIs", and "Zowe : JES Jobs APIs". Each section has a "Show/Hide", "List Operations", and "Expand Operations" link. The "Zowe : JES Jobs APIs" section is expanded, showing a list of REST API endpoints with their methods and descriptions:

Method	Endpoint	Description
GET	/api/v1/jobs	Get a list of jobs
POST	/api/v1/jobs	Submit a job and get the job id back
GET	/api/v1/jobs/{jobName}/ids	Get a list of job identifiers for a given job name
GET	/api/v1/jobs/{jobName}/ids/{jobId}/files	Get a list of output file names for a job
GET	/api/v1/jobs/{jobName}/ids/{jobId}/files/{fileId}	Read content from a specific job output file
GET	/api/v1/jobs/{jobName}/ids/{jobId}/files/{fileId}/tail	Read the tail of a job's output file
GET	/api/v1/jobs/{jobName}/ids/{jobId}/steps	Get job steps for a given job
GET	/api/v1/jobs/{jobName}/ids/{jobId}/steps/{stepNumber}/dds	Get dataset definitions (DDs) for a given job step
GET	/api/v1/jobs/{jobName}/ids/{jobId}/subsystem	Get the subsystem type for a job



# Zowe examples – the API Catalog



The screenshot shows the Zowe API Catalog interface for the endpoint `GET /api/v1/jobs`. The page title is "Zowe : JES Jobs APIs". The interface includes a search bar, navigation buttons, and a "Get a list of jobs" button. The main content area displays "Implementation Notes" stating that the API returns a list of jobs for a given prefix and owner. Below this is the "Response Class (Status 200)" section, which shows an "Ok" status and a "Model" tab with an "Example Value" of a JSON array. The JSON array contains a single job object with fields: `name` (string), `jobInstances` (array), `jobId` (string), `jobName` (string), `owner` (string), `type` (string), and `status` (string, value: "ACTIVE"). The "Response Content Type" is set to `application/json`. At the bottom, there is a "Parameters" table with three rows: `prefix` (string, query), `owner` (string, query), and `status` (string, query). A "Try it out!" button is located at the bottom left of the interface.

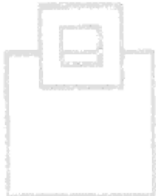
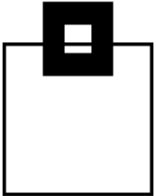
```
[
  {
    "name": "string",
    "jobInstances": [
      {
        "jobId": "string",
        "jobName": "string",
        "owner": "string",
        "type": "string",
        "status": "ACTIVE",

```

Parameter	Value	Description	Parameter Type	Data Type
prefix	<input type="text" value="*"/>	Job name prefix. If omitted, defaults to "*".	query	string
owner	<input type="text"/>	Job owner. Defaults to requester's userid.	query	string
status	<input type="text" value=""/>	Job status to filter on, defaults to ALL.	query	string



# Zowe examples – the API Catalog



**Zowe : JES Jobs APIs** Show/Hide List Operations Expand Operations

**GET** /api/v1/jobs Get a list of jobs

**POST** /api/v1/jobs Submit a job and get the job's back

Implementation Notes  
This API submits a partitioned data set member or Unix file. For fully qualified dataset members use 'MYJOB.S.TEST.CNTL(TESTJOBX)' For non fully qualified use TEST.CNTL(TESTJOBX) For Unix files use /u/myjobs/job1

Response Class (Status 201)  
Job successfully created

Model Example Value

```
{
  "jobId": "string",
  "jobName": "string",
  "owner": "string",
  "type": "string",
  "status": "ACTIVE",
  "returnCode": "string",
  "subsystem": "string",
  "executionClass": "string",
  "jobname": "string"
}
```

Response Content Type: application/json

Parameters:

Parameter	Value	Description	Parameter Type	Data Type
body	<input type="text" value="(*file):HEINRIC_JOB.CNTL(1E1BR14)"/>	USS file path or Data set name in the form: (*file):AF.LAS.1ES1JCL(1S1J0001) (*file):1ES1JCL(1S1J0001) (*file):/u/myjobs/job1	body	string

Parameter content type: application/json

**Try it out** [View Response](#)

**Curl**

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{"file": "HEINRIC_JOB.CNTL(1E1BR14)"}' 'https://xib1.fr.itz.ibm.com/zowe/api/v1/jobs'
```

**Request URL**

```
https://xib1.fr.itz.ibm.com/zowe/api/v1/jobs
```

**Response Body**

```
{
  "jobId": "10000000",
  "jobName": "HEINRICCS",
  "owner": "HEINRIC",
  "type": "JOB",
  "status": "SUBMIT",
  "subsystem": "JES3",
  "executionClass": "A",
  "jobname": "Job is queued for conversion"
}
```

**Response Code**

```
201
```

**Response Headers**

```
{
  "content-language": "en-US",
  "content-length": "122",
  "content-type": "application/json",
  "date": "Tue, 13 Mar 2018 09:18:06 GMT",
  "location": "https://xib1.fr.itz.ibm.com/zowe/api/v1/jobs/HEINRICCS/10000000",
  "x-powered-by": "Servlet/3.1"
}
```

**GET** /api/v1/jobs/{jobName}/ids Get a list of job identifiers for a given job name

# Zowe examples – the API Catalog

The screenshot displays the Zowe API Catalog interface for the endpoint `/api/v1/jobs/{jobName}/steps/{jobId}`. The interface includes a list of endpoints, implementation notes, response class information, a model example, a parameters table, curl command, request URL, response body, response code, and response headers.

**Implementation Notes:** This API returns the details of a job for a given job name and identifier.

**Response Class (Status 200):** OK

**Model Example Value:**

```
{
  "jobId": "string",
  "jobName": "string",
  "owner": "string",
  "type": "string",
  "status": "ACTING",
  "returnValue": "string",
  "subsystem": "string",
  "executionClass": "string",
  "phaseName": "string"
}
```

**Parameters Table:**

Parameter	Value	Description	Parameter Type	Data Type
jobName	HEINRICH	Job name.	path	string
jobId	JOB00009	Job identifier.	path	string

**Try it out:** [View Response](#)

**Curl:**

```
curl -X GET --header 'Accept: application/json' 'https://swg1.fr1.us-east-1.amazonaws.com/api/v1/jobs/HEINRICH/JOB00009'
```

**Request URL:** `https://swg1.fr1.us-east-1.amazonaws.com/api/v1/jobs/HEINRICH/JOB00009`

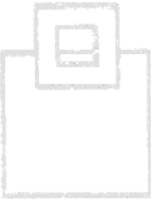
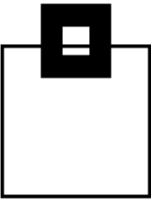
**Response Body:**

```
{
  "jobId": "JOB00009",
  "jobName": "HEINRICH",
  "owner": "HEINRICH",
  "type": "JOB",
  "status": "OUTPUT",
  "returnValue": "CC 0000",
  "subsystem": "JES3",
  "executionClass": "A",
  "phaseName": "Job is on the hard copy queue"
}
```

**Response Code:** 200

**Response Headers:**

```
{
  "content-language": "en-US",
  "content-length": "287",
  "content-type": "application/json",
  "date": "Tue, 10 Mar 2020 16:12:04 GMT",
  "x-powered-by": "Servlet/3.1"
}
```



# Zowe examples – the API Catalog

**Zowe : Dataset APIs** Show Hide List Operations Expanded Operations

- DELETE** /api/v1/datasets/{dsn} Delete a data set or member
- POST** /api/v1/datasets/{dsn} Create (and possibly) a data set
- GET** /api/v1/datasets/{dsn}/attributes Retrieve attributes of a data set(s)
- GET** /api/v1/datasets/{dsn}/content Read content from a data set or member

**Implementation Notes**  
This API reads content from a sequential data set or member of a partitioned data set.

**Response Class (Status 200)**  
OK

**Model: Example Value**

```
{
  "records": "This is some text content.",
  "checksum": "string"
}
```

**Response Content Type** application/json

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
dsn	heinfo.job.cmt(IEFBR14)	Data set name, e.g. HLQ.PS or HLQ.PD(WHERE)	path	string
convert	True (default)	Indicator to codepage convert content	query	boolean
checksum	True	Indicator to return a checksum (if planning subsequent write)	query	boolean
start		Starting relative record number to read. Defaults to record 0.	query	string
end		Ending relative record number to read. If not specified, all records are read.	query	string

**Try it out** [Info Response](#)

**Curl**

```
curl -X GET --header 'Accept: application/json' 'https://w03.frizx.box:7443/api/v1/datasets/heinfo.job.cmt(IEFBR14)/content?convert=true&checksum=true'
```

**Request URL**

```
https://w03.frizx.box:7443/api/v1/datasets/heinfo.job.cmt(IEFBR14)/content?convert=true&checksum=true
```

**Response Body**

```
{
  "records": "/HEINFOS JOB CLASS=A,MODELAS=H,MSGLVL=1,1,1,TPRDN=SCAN,IN//          MDTIFH8VYSU0D=H/PRCALLOC EXEC PGM=IEFBR14H//SYSOUT  DD SYSOUT=*//",
  "checksum": "FF6E189700F960C843D078B37E16A2E"
}
```

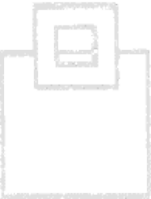
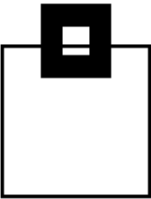
**Response Code**

```
200
```

**Response Headers**

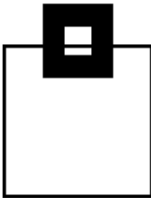
```
{
  "content-language": "en-US",
  "content-length": "280",
  "content-type": "application/json",
  "date": "Tue, 19 Mar 2019 16:55:52 GMT",
  "x-powered-by": "Express/3.1"
}
```

- PUT** /api/v1/datasets/{dsn}/content Write content to a data set or member
- GET** /api/v1/datasets/{dsn}/members Get a list of members for a partitioned data set
- GET** /api/v1/datasets/{filter} Get a list of data sets by filter





# Zowe examples – the API Catalog



**PUT** /api/v1/datasets/{dsn}/content Write content to a data set or member

**Implementation Notes**  
This API writes content to a sequential data set or partitioned data set member.

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
dsn	heinicr.job.cnt(iefbr14)	Dataset name	path	string
body	{ "records": "//HEINRICHS JOB CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),\n// NOTIFY=&SYSUID\n//PREALLOC EXEC PGM=IEFBR14\n//SYSOUT DD SYSOUT=*" }	Request content (content-type:application/json) in the form: ("records":"data Content","checksum":"checksum_value") if checksum is passed and it does not match the checksum returned by a previous read, it is deemed a concurrent update has occurred, and the write fails.	body	string

Parameter content type: application/json

**Response Messages**

HTTP Status Code	Reason	Response Model	Headers
200	Ok		

[Try it out!](#) [Hide Response](#)

**Curl**

```
curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
  "records": "//HEINRICHS JOB CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),\n// NOTIFY=&26SYSUID\n//PREALLOC EXEC PGM=IEFBR14\n//SYSOUT DD SYSOUT=*" \
}' 'https://s0w1.fritz.box:7443/api/v1/datasets/heinicr.job.cnt1(iefbr14)/content'
```

**Request URL**

https://s0w1.fritz.box:7443/api/v1/datasets/heinicr.job.cnt1(iefbr14)/content

**Response Body**

no content

**Response Code**

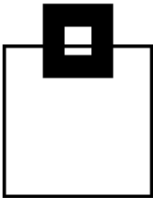
200

**Response Headers**

```
{
  "content-language": "en-US",
  "content-length": "0",
  "date": "Tue, 12 Mar 2019 14:19:01 GMT",
  "x-powered-by": "Servlet/3.1"
}
```



# Zowe examples – the API Catalog



**Zowe : JES Jobs APIs** Show/Hide Last Operations Expand Operations

**GET /api/v1/jobs** Get a list of jobs

**POST /api/v1/jobs** Submit a job and get the job list back

Implementation Notes  
This API submits a partitioned data set member or Unix file. For fully qualified dataset members use 'MYJOBS.TEST.CNTL(TESTJOBX)' For non fully qualified use TEST.CNTL(TESTJOBX) For Unix files use /u/myjobs/job1

Response Class (Status 201)  
Job successfully created

Model | Example Value

```
{
  "jobId": "string",
  "jobName": "string",
  "owner": "string",
  "type": "string",
  "status": "ACTJOB",
  "reasonCode": "string",
  "subsystem": "string",
  "executionClass": "string",
  "phaseName": "string"
}
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
body	<code>{ "file": "HEIMLIC_JOB.CNTL(12FR814)" }</code>	USS file path or Data set name in the form: ('file': 'ALIAS.JES1.JCL(1S1J0001)', ('file': 'JES1.JCL(1S1J0001)', ('file': '/u/myjobs/job1')	body	string

Parameter content type: application/json

Try it out: [View Response](#)

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ "file": "HEIMLIC_JOB.CNTL(12FR814)" }' "https://s0w1.fritz.box:7443/api/v1/jobs"
```

Request URL

```
https://s0w1.fritz.box:7443/api/v1/jobs
```

Response Body

```
{
  "jobId": "3088819",
  "jobName": "HEIMLIC",
  "owner": "HEIMLIC",
  "type": "JOB",
  "status": "ACTIVE",
  "subsystem": "JES2",
  "executionClass": "A",
  "phaseName": "JOB is actively executing"
}
```

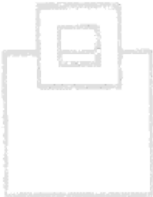
Response Code

```
201
```

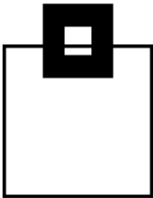
Response Headers

```
{
  "content-language": "en-US",
  "content-length": "179",
  "content-type": "application/json",
  "date": "Tue, 10 Mar 2020 16:58:31 GMT",
  "location": "https://s0w1.fritz.box:7443/api/v1/jobs/HEIMLIC/3088819",
  "powered-by": "Zowe/v1.11"
}
```

<https://s0w1.fritz.box:7443/ibm/api/explorer/> Get a list of job instances for a given job name



# Zowe examples – the API Catalog



**GET** /api/v1/jobs/{jobName}/{jobId}

Get the details of a job for a given job name and identifier.

**Implementation Notes**  
This API returns the details of a job for a given job name and identifier.

**Response Class (Status 200)**  
OK

**Model Example Value**

```
{
  "jobId": "HEINRIC",
  "jobName": "HEINRIC",
  "owner": "HEINRIC",
  "type": "JOB",
  "status": "OUTPUT",
  "returnCode": "CC 0000",
  "subsystem": "JES2",
  "executionClass": "A",
  "phaseName": "JOB is on the hard copy queue"
}
```

**Response Content Type** application/json

Parameter	Value	Description	Parameter Type	Data Type
jobName	HEINRIC	job name	path	string
jobId	JOB0019	job identifier	path	string

[Try it out](#) [View Response](#)

**Curl**

```
curl -X GET --header 'Accept: application/json' 'https://nlw1.fr1tx.baw:7443/api/v1/jobs/HEINRIC24/JOB0019'
```

**Request URL**

```
https://nlw1.fr1tx.baw:7443/api/v1/jobs/HEINRIC24/JOB0019
```

**Response Body**

```
{
  "jobId": "JOB0019",
  "jobName": "HEINRIC",
  "owner": "HEINRIC",
  "type": "JOB",
  "status": "OUTPUT",
  "returnCode": "CC 0000",
  "subsystem": "JES2",
  "executionClass": "A",
  "phaseName": "JOB is on the hard copy queue"
}
```

**Response Code**

```
200
```

**Response Headers**

```
{
  "content-language": "en-US",
  "content-length": "139",
  "content-type": "application/json",
  "date": "Tue, 10 Mar 2020 16:59:41 GMT",
  "x-powered-by": "Servlet/3.1"
}
```

Zowe : System APIs

Zowe : USS Files APIs

Zowe : zOS System APIs

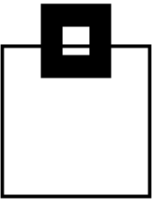
Show/Hide | List Operations | Expand Operations

Show/Hide | List Operations | Expand Operations

Show/Hide | List Operations | Expand Operations



# Zowe examples – the API Catalog



api/v1/jobs/{jobName}/ids/{jobId}/files Get a list of output file names for a job.

Implementation Notes  
This API returns the output file names for a given job.

Response Class (Status 200)  
OK

Model | Example Value

```
{
  "idname": "string",
  "recta": "string",
  "irec1": 0,
  "byteCount": 0,
  "recordCount": 0,
  "id": 0
}
```

Response Content Type: application/json

Parameter	Value	Description	Parameter Type	Data Type
jobName	helios	Job name.	path	string
jobId	JOB08318	Job identifier.	path	string

Try it out: [View Response](#)

Curl

```
curl -X GET --header 'Accept: application/json' 'https://ibm1.fritz.box:7443/api/v1/jobs/helios324/ids/2088815/files'
```

Request URL

```
https://ibm1.fritz.box:7443/api/v1/jobs/helios324/ids/2088815/files
```

Response Body

```
{
  "idname": "D65N9L0",
  "recta": "0A",
  "irec1": 111,
  "byteCount": 883,
  "recordCount": 18,
  "id": 2
},
{
  "idname": "D6330L1",
  "recta": "V",
  "irec1": 116,
  "byteCount": 112,
  "recordCount": 5,
  "id": 3
},
{
  "idname": "D61Y050",
  "recta": "not"
}
```

Response Code

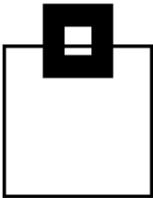
```
200
```

Response Headers

```
{
  "content-language": "en-US",
  "content-length": "258",
  "content-type": "application/json",
  "date": "Tue, 12 Mar 2019 15:22:39 GMT",
  "powered-by": "Service/3.1"
}
```



# Zowe examples – the API Catalog



GET /api/v1/jobs/{jobName}/ids/{jobId}/files

Implementation Notes  
This API returns the output file names for a given job.

Response Class (Status 200)  
OK

Model: Example Value

```
[
  {
    "idname": "string",
    "race": "string",
    "ireel": 0,
    "byteCount": #,
    "recordCount": 0,
    "id": 0
  }
]
```

Response Content Type: application/json

Parameter	Value	Description	Parameter Type	Data Type
jobName	hehnic0	Job name.	path	string
jobId	JOB08910	Job identifier.	path	string

Try it out! [View Response](#)

Curl

```
curl -X GET --header 'Accept: application/json' 'https://wbu1.fritz.box:7443/api/v1/jobs/hehnic0/ids/JOB08910/files'
```

Request URL

```
https://wbu1.fritz.box:7443/api/v1/jobs/hehnic0/ids/JOB08910/files
```

Response Body

```
[
  {
    "idname": "3E3RSGLD",
    "race": "UA",
    "ireel": 111,
    "byteCount": 803,
    "recordCount": 38,
    "id": 2
  },
  {
    "idname": "3E3DCL",
    "race": "V",
    "ireel": 118,
    "byteCount": 312,
    "recordCount": 5,
    "id": 3
  },
  {
    "idname": "3E3VRSQ",
    "race": "M",
    "id": 4
  }
]
```

Response Code

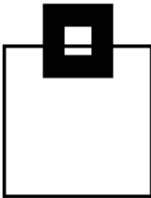
```
200
```

Response Headers

```
{
  "content-language": "en-US",
  "content-length": "218",
  "content-type": "application/json",
  "date": "Tue, 12 Mar 2019 15:22:39 GMT",
  "x-powered-by": "Servlet/3.1"
}
```



# Zowe examples – the API Catalog



GET /api/v1/jobs/{jobName}/ids/{jobId}/files/{fileId} Read content from a specific job output file

**Implementation Notes**  
This API reads content from a specific job output file. The API can read all output, or a relative record range.

**Response Class (Status 200)**  
OK

**Model / Example Value**

```
{
  "content": "string"
}
```

**Response Content Type** application/json

Parameter	Value	Description	Parameter Type	Data Type
jobName	heinfo	Job name.	path	string
jobId	JOB08818	Job identifier.	path	string
fileId	4	Job file id number.	path	string
start		Optional starting relative record number to read.	query	string
end		Optional ending relative record number to read. If omitted, all records are returned.	query	string

[Try it out!](#) [Hide Response](#)

**Curl**

```
curl -X GET --header 'Accept: application/json' 'https://xbl1.fr1tz.box:7443/api/v1/jobs/heinfo/ids/JOB08818/files/4'
```

**Request URL**

```
https://xbl1.fr1tz.box:7443/api/v1/jobs/heinfo/ids/JOB08818/files/4
```

**Response Body**

```
{
  "content": " 2D078002 HDWRCV LAST ACCESS AT 13:38:46 ON TUESDAY, MARCH 12, 2019\n\n IEFAP111 HEIMICS IS USING THE FOLLOWING JOB RELATED SETTINGS:\n\n          SNA=ABOVE, FID1 SIZE=32K, DSENGSR=DISALLOW, QDORAS=JOB\n\n IEF2302 ALLOC. FOR HEIMICS PREALLOC\n\n IEF2372 2652 ALLOCATED TO SYSOUT\n\n IEF1421 HEIMICS PREALLOC - STEP WAS EXECUTED - COND CODE 0000\n\n IEF22
```

**Response Code**  
200

**Response Headers**

```
{
  "content-language": "en-US",
  "content-length": "944",
  "content-type": "application/json",
  "date": "Tue, 12 Mar 2019 13:37:08 GMT",
  "x-powered-by": "Servlet/3.1"
}
```

GET /api/v1/jobs/{jobName}/ids/{jobId}/files/{fileId}/tail Read the tail of a job's output file

GET /api/v1/jobs/{jobName}/ids/{jobId}/steps Get job steps for a given job

GET /api/v1/jobs/{jobName}/ids/{jobId}/steps/{stepNumber}/jobs Get dataset definitions (DDNs) for a given job step

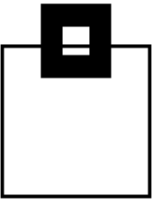
GET /api/v1/jobs/{jobName}/ids/{jobId}/subsystem Get the subsystem type for a job

DELETE /api/v1/jobs/{jobName}/{jobId} Cancel a Job and purge it's associated files

GET /api/v1/jobs/{jobName}/{jobId} Get the details of a job for a given job name and identifier

**Implementation Notes**  
This API returns the details of a job for a given job name and identifier.

# Zowe examples – User Tasks/Workflows



Browser window: Zowe Desktop

Page title: User Tasks/Workflows

Navigation: My tasks | Workflows | Warnings | **Configuration**

Section: Active z/OSMF server

Host:  Port:  ● Online Test

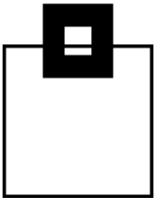
Buttons: + | Reload | Save

Host:  Port:  ● Online Test | Set as active server | ✕

Bottom buttons: Test All | Cancel | OK



# Zowe examples – User Tasks/Workflows



**Create Workflow**

Advanced Mode

Name: ContinuousDelivery DeploymentCheck for Db2 z/OS

Variables

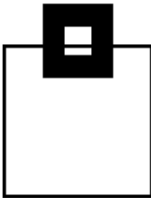
Name	Value	
sourceDb	ACG	X
targetDb	BCG	X
testMode	No	X
localSystemDSN	SE.PRODUCT.EASY0200.CDDC.EXEC	X
remoteSystemDSN	SE.PRODUCT.EASY0200.CDDC.EXEC	X
scenario	CDCAFDA3	X
backupDSN	SE.BACKUP.ACG.DUMP	X
workloadSet	2019-03-18-11.56.25.524960	X

Buttons: Cancel, OK

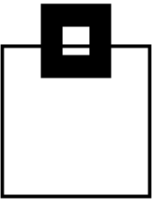




# Zowe examples – the Editor



# Hands on usage based on a cloning example



- Goal: Run a batch job based Db2 system cloning process out of Zowe



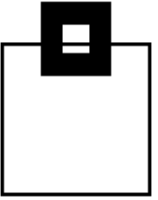
```
Continuous Delivery Deployment Check ----- Scenario Control Menu -----
Command ==>
MENU=ON SCENARIO=CDCAFDA3 SOURCE=N/A TARGET=N/A OFFLINE COPY - TEST MODE
INTEGRATED FRENAME - NEW TARGET - INTERSYSTEM SCENARIO (SOURCE BASED)

Execute options 1 through 45 in sequence by pressing ENTER.
Enter M to switch between menu display ON or OFF.

Press ENTER to proceed with Select DB2

==>  1. Select DB2           - Select source
      2. Select DB2           - Select target
      3. Prepare              - Define datasets
      4. Build samples        - Generate sample input for new DB2s
      5. Set environment      - General cloning options and sources
      6. Validate variables   - Check customer variables
      7. Validate datasets    - Check installation specific datasets
      8. Gather information    - Get all needed information
      9. WLX Variables        - Source: Get WLX Variables
     10. WLX STC              - Source: Start WLX STC
```

# Hands on usage based on a cloning example



- Goal: Run a batch job based Db2 system cloning process out of Zowe

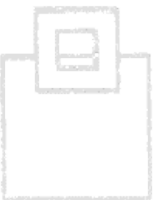
```
Zowe Desktop
Webadresse suchen oder eingeben
TN3270
----- Scenario Control Menu -----
Command ==>
MENU=ON SCENARIO=CDCAFDA3 SOURCE=N/A TARGET=N/A OFFLINE COPY - TEST MODE
INTEGRATED FRENAME - NEW TARGET - INTERSYSTEM SCENARIO (SOURCE BASED)

Execute options 1 through 45 in sequence by pressing ENTER.
Enter M to switch between menu display ON or OFF.

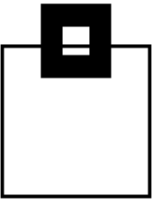
Press ENTER to proceed with select DB2

==> 1. Select DB2          - select source
      2. Select DB2       - select target
      3. Prepare          - Define datasets
      4. Build samples    - Generate sample input for new DB2s
      5. Set environment  - General cloning options and sources
      6. validate variables - Check customer variables
      7. Validate datasets - Check installation specific datasets
      8. Gather information - Get all needed information
      9. WLX Variables    - Source: Get WLX Variables
     10. WLX STC          - Source: Start WLX STC

MA      A
2 /15
```



# Hands on usage based on a cloning example

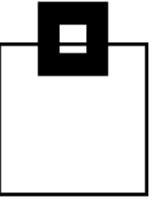


- The flow of batch jobs is driven by a XML scenario:

```
--
46 <menuitem>                                <!-- Menu item 01 -->
47 <name>Select DB2</name>
48 <description>Select source</description>
49 </menuitem>
50 <menuitem>                                <!-- Menu item 02 -->
51 <name>Select DB2</name>
52 <description>Select target</description>
53 </menuitem>
54 <menuitem>                                <!-- Menu item 03 -->
55 <name>Prepare</name>
56 <description>Define datasets</description>
57 </menuitem>
58 <menuitem>                                <!-- Menu item 04 -->
59 <name>Build samples</name>
60 <description>Generate sample input for new DB2s</description>
61 </menuitem>
62 <menuitem>                                <!-- Menu item 05 -->
63 <name>Set environment</name>
64 <description>General cloning options and sources</description>
65 </menuitem>
66 <menuitem>                                <!-- Menu item 06 -->
67 <name>Validate variables</name>
68 <description>Check customer variables</description>
69 </menuitem>
70 <menuitem>                                <!-- Menu item 07 -->
71 <name>Validate datasets</name>
72 <description>Check installation specific datasets</description>
73 </menuitem>
74 <menuitem>                                <!-- Menu item 08 -->
75 <name>Gather information</name>
76 <description>Get all needed information</description>
77 </menuitem>
78 <menuitem>                                <!-- Menu item 09 -->
79 <name>WLX Variables</name>
80 <description>Source: Get WLX Variables</description>
81 </menuitem>
82 <menuitem>                                <!-- Menu item 10 -->
83 <name>WLX STC</name>
84 <description>Source: Start WLX STC</description>
85 </menuitem>
86 <menuitem>                                <!-- Menu item 11 -->
87 <name>WLX TIA PFC</name>
```



# Hands on usage based on a cloning example

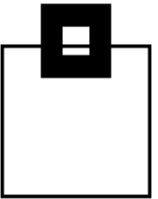


- The flow of batch jobs is migrated to a Workflow:

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <workflow>
3   <autoTakeOwnership>false</autoTakeOwnership>
4   <workflowInfo>
5     <workflowID scope="none">MyWF</workflowID>
6     <workflowDescription>ContinuousDelivery DeploymentCheck for Db2 z/OS</workflowDescription>
7     <workflowVersion>1.0</workflowVersion>
8     <vendor>SOFTWARE ENGINEERING</vendor>
9     <General/>
10  </workflowInfo>
11  <step name="Prepare" optional="false">
12    <title>Prepare Systems for Cloning</title>
13    <description/>
14    <step name="CDCAFDA3" optional="false">
15      <title>Define datasets</title>
16      <description>PREPARE CLONING (ISC SUPPORT INCLUDED)
17      PROVIDE SCENARIO SPECIFIC DATA SETS
18      PROVIDE CUSTOMIZATION MEMBER HSC#VARS </description>
19      <instructions substitution="false">Generated instruction text for step: PrepareSource
20      Update this field with your own text</instructions>
21      <weight>5</weight>
22      <autoEnable>false</autoEnable>
23      <canMarkAsFailed>false</canMarkAsFailed>
24    </step>
25    <httpMethod>POST</httpMethod>
26    <uriPath substitution="false">https://s0w1.fritz.box:7443/api/v1/jobs curl -X POST </uriPath>
27    <requestBody substitution="false">{"file":"HEINRIC.JOB.CNTL(IEFBR14)"}</requestBody>
28    <expectedStatusCode>201</expectedStatusCode>
29  </step>
30 </workflow>
31
```

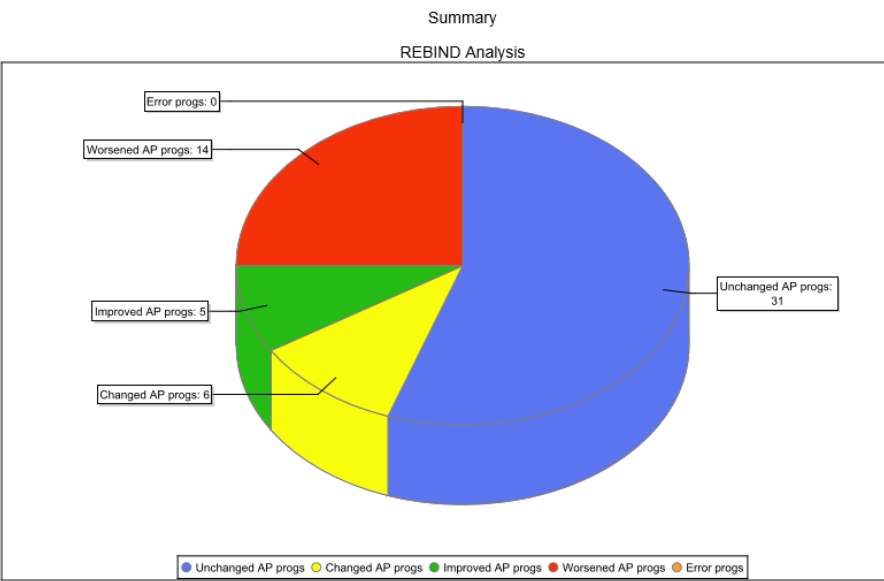


# Hands on usage based on a cloning example



## Access Path Check – Static & Dynamic SQL Access Path Pre- and/ or Post-Check:

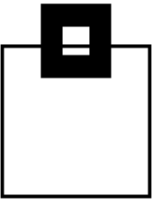
STMT No.	Section Number	DSC STMT ID	Im
2547	13	2547	DE
796	2	796	DE
804	3	804	DE
812	4	812	DE
820	5	820	DE
671	2	671	DE
679	3	679	DE
687	4	687	DEGRADED
695	5	695	DEGRADED
3	3	3	DEGRADED
4	4	4	DEGRADED
5	5	5	DEGRADED
6	6	6	DEGRADED
968	7	968	IMPROVED
1167	9	1167	IMPROVED
235	1	235	IMPROVED
1194	2	1194	IMPROVED
2409	7	2409	IMPROVED
1194	2	1194	IMPROVED



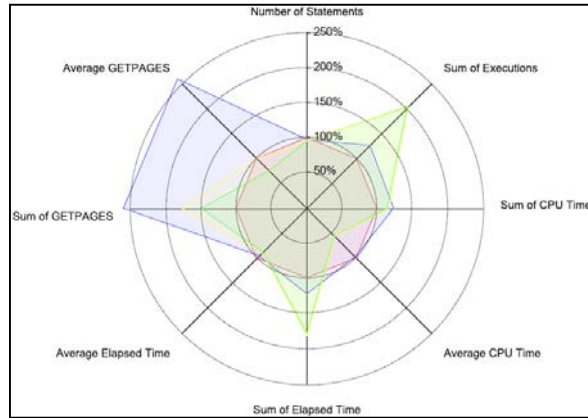
CDDC results summary

WLX	BIX	All	Invalid	Inoperative
Packages	Analyzed	197	0	0
	Not analyzed	683	22	0
	Improved	16	0	0
	Worsened	19	0	0
	Changed	22	0	0
	Unchanged	140	0	0
Statements static	Analyzed	1730	0	0
	Not analyzed	2810	0	0
	Improved	214	0	0
	Worsened	54	0	0
	Changed	72	0	0
	Unchanged	1390	0	0
Statements dynamic	Analyzed	296	0	0
	Not analyzed	52	0	0
	Improved	3	0	0
	Worsened	34	0	0
	Changed	15	0	0
	Unchanged	244	0	0

# Hands on usage based on a cloning example



- Drill down to look into details, when anomalies are detected



Function level	Catalog level
V12R1M500	V12R1M503
V12R1M503	V12R1M503

of Number of Statements	Sum of GETPAGES	Sum of Synchronous Buffer Reads	Sum of Rows examined	Sum of Rows processed	Sum of Sorts performed	Sum of Index scans
861	86540	2926	832096	1935	136	165754
874	86713	2909	832408	1951	136	165758

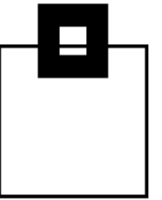
Sum of WF and Tablespace Scans	Sum of Parallel Groups	Sum of Synchronous Buffer Writes	Sum of Elapsed Time	Sum of Wait Latch Request	Sum of Wait Page Latch	Sum of Wait Drain Lock	Sum of Wait Drain Claims	Sum of Wait Log Writer
282	0	79	267413647	221714	150707	9309254	0	1681516
279	0	79	265360495	362544	19190	7138740	0	1636997

Sum of Wait Synchronous IO	Sum of Wait Lock requests	Sum of Wait Synchronous Execution	Sum of Wait Global Locks	Sum of Wait other Thread Read	Sum of Wait other Thread Write	Sum of No RID Limits	Sum of No RID Storage	Sum of no RID WF Storage	Sum of no RID WF Limits
25248113	4249320	0	7867344	6456613	883679	0	0	0	0
21286897	3750693	0	5892333	6395228	586230	0	0	0	0



# Hands on usage based on a cloning example

---



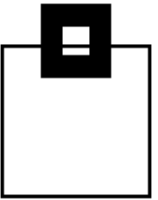
- Due to the nature of Zowe anything can be combined with everything, e.g.
  - Console, Shell, Db2 COMMANDS
  - JOBS
  - REXXs
  - Instructions
  - ...
- and any information can be accessed:
  - Any type of MVS/USS dat sets
  - Job output
  - ...



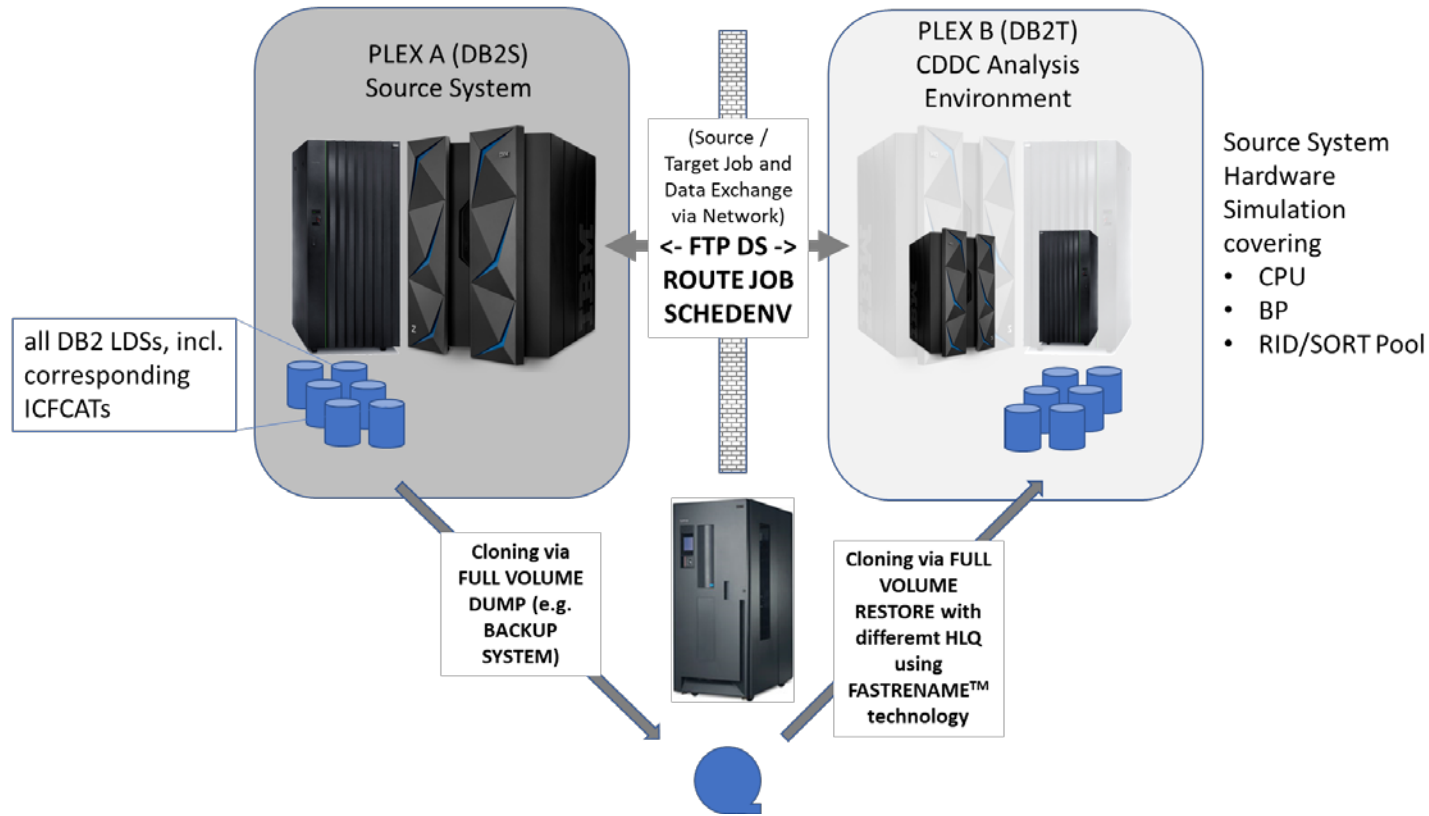
→ This makes the Zowe desktop your single point of control



# Hands on usage based on a cloning example

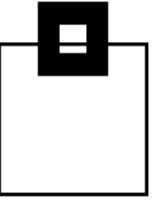


## Instant Cloning - Clone based code level checks:



# Hands on usage based on a cloning example

---



- Zowe is perfect for ContinuousDelivery DeploymentCheck for Db2 z/OS:
    - We automatically clone a source Db2 into a target Db2
    - We can apply changes into the target Db2
    - We can replay workload, captured from source
    - We can do before and after comparisons within our clone
    - We can spot differences due to
      - BIF/ICI
      - Application changes
      - Access path changes
    - And we can display the results nicely in a HTML5 GUI
- The entire process can be fully automated, but customized as needed

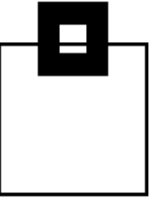


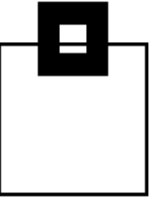
# Summary of experience

---

- Starting with Zowe can be challenging, depending on your accessible resources/knowledge:
  - MVS
  - Unix
  - Security
    - Authorization
    - Certificates
  - Tomcat
  - zOSMF
  - ...

, but ...





### ■ It's worth it!!!

- We started with quite early (< 1.0) versions, but 1.0.1 was released on the 7th of March
  - It starts to become solid and certainly ready to look at it
- Use any of your z/OS capabilities as a cloud service
- Make your z/OS system accessible for non ISPFers
- Modernize z/OS applications
- Attract the youngsters to exploit the strength of the z platform
- SEGUS is committed to exploit Zowe with their existing and upcoming tools and to contribute to the new ecosystem.

