ZOWE – The zGUI (r)evolution

First hands-on experience and best practices

Roy Boxwell

SEGUS & SOFTWARE ENGINEERING

Session code: V1

06/04/2019 - 10:40 am

Db2 z/OS

IDUG

Leading the Db2 User Community since 1988

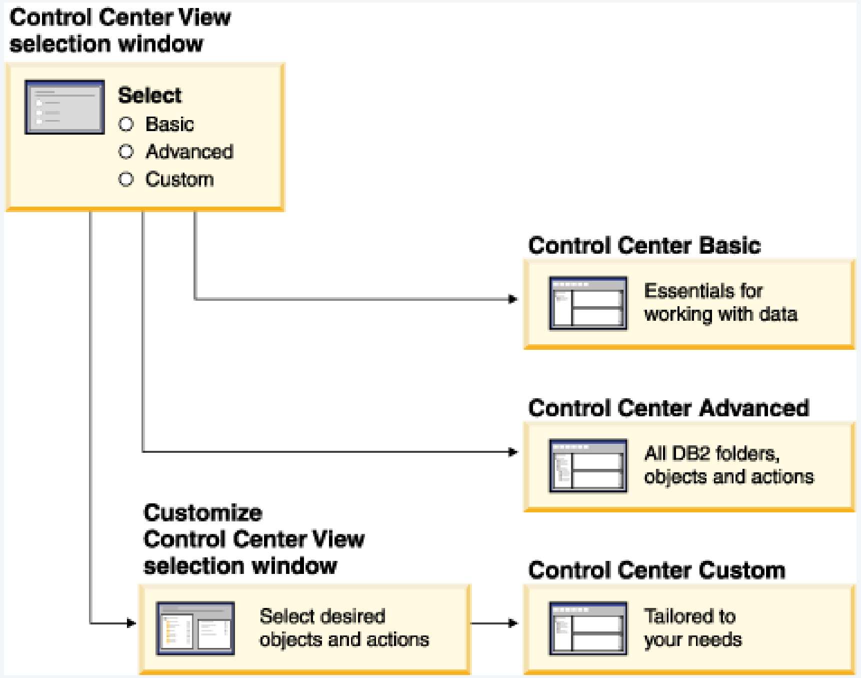
Agenda

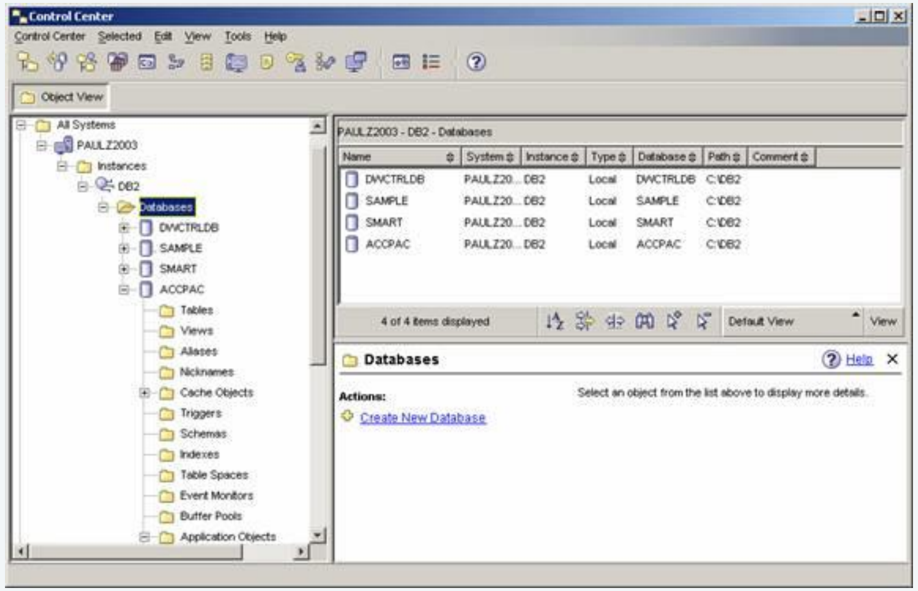
- GUIs in the past
- Zowe ecosystem overview
- Zowe differentiation to prior GUIs
- Zowe components
- Zowe examples
- Hands-on usage based on a cloning example
- Summary of experience

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
Select
Basic
Advanced
Custom
Control Center Basic
Essentials for working with data
Control Center Advanced
All DB2 folders, objects and actions
Customize Control Center View selection window
Select desired objects and actions
Control Center Custom
Tailored to your needs



PAULZ2003 - DB2 - Databases

Name	System	Instance	Type	Database	Path	Comment
DWCTRLDB	PAULZ20...	DB2	Local	DWCTRLDB	C:\DB2	
SAMPLE	PAULZ20...	DB2	Local	SAMPLE	C:\DB2	
SMART	PAULZ20...	DB2	Local	SMART	C:\DB2	
ACCPAC	PAULZ20...	DB2	Local	ACCPAC	C:\DB2	

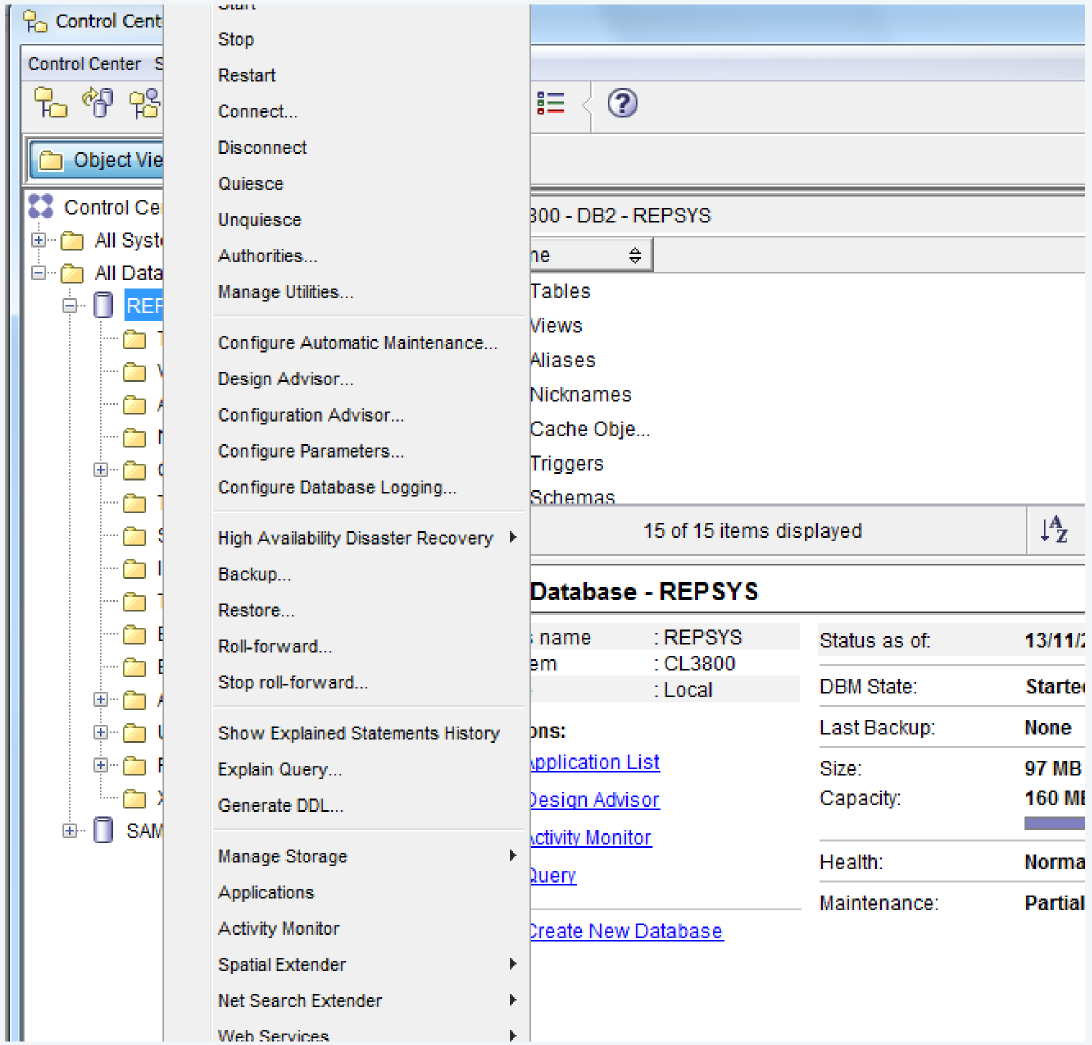
4 of 4 items displayed
Default View
View

Databases
Help
Actions:
Create New Database
Select an object from the list above to display more details.

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



name	: REPSYS
em	: CL3800
	: Local

Status as of:	13/11/
DBM State:	Starte
Last Backup:	None
Size:	97 MB
Capacity:	160 MI
Health:	Norma
Maintenance:	Partial

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

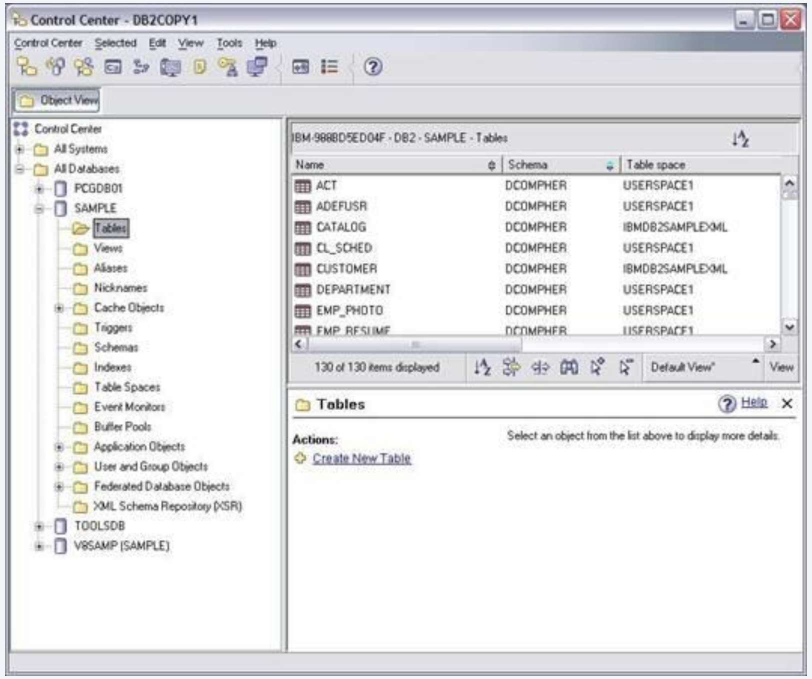
Db2 Control Center (Db2cc)

- …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



Control Center - DB2COPY1

Control Center Selected Edit View Tools Help

Object View

- Control Center
  - All Systems
  - All Databases
    - PCGDB01
    - SAMPLE
      - Tables
      - Views
      - Aliases
      - Nicknames
      - Cache Objects
      - Triggers
      - Schemas
      - Indexes
      - Table Spaces
      - Event Monitors
      - Buffer Pools
      - Application Objects
      - User and Group Objects
      - Federated Database Objects
      - XML Schema Repository (XSR)
    - TOOLSDB
    - V8SAMP (SAMPLE)

IBM-988BD5ED04F - DB2 - SAMPLE - Tables

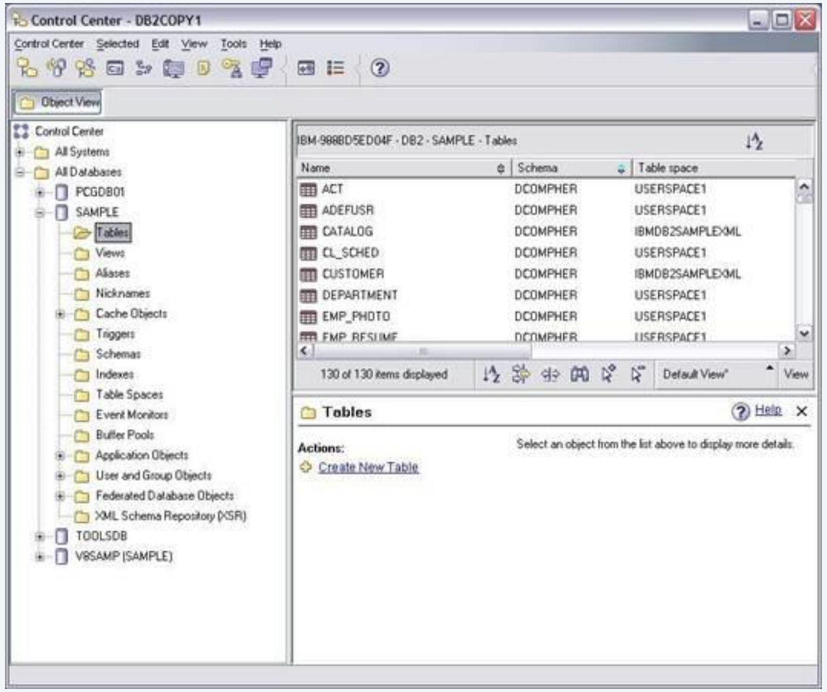
Name	Schema	Table space
ACT	DCOMPHER	USERSPACE1
ADEFUSR	DCOMPHER	USERSPACE1
CATALOG	DCOMPHER	IBMDB2SAMPLEXML
CL_SCHED	DCOMPHER	USERSPACE1
CUSTOMER	DCOMPHER	IBMDB2SAMPLEXML
DEPARTMENT	DCOMPHER	USERSPACE1
EMP_PHOTO	DCOMPHER	USERSPACE1
EMP_RESUME	DCOMPHER	USERSPACE1

130 of 130 items displayed Default View View

Tables Help

Actions:
Create New Table

Select an object from the list above to display more details.



Control Center - DB2COPY1

Control Center Selected Edit View Tools Help

Object View

IBM-988BD5ED04F - DB2 - SAMPLE - Tables

130 of 130 items displayed Default View View

Tables Help

Actions:
Create New Table

Select an object from the list above to display more details.

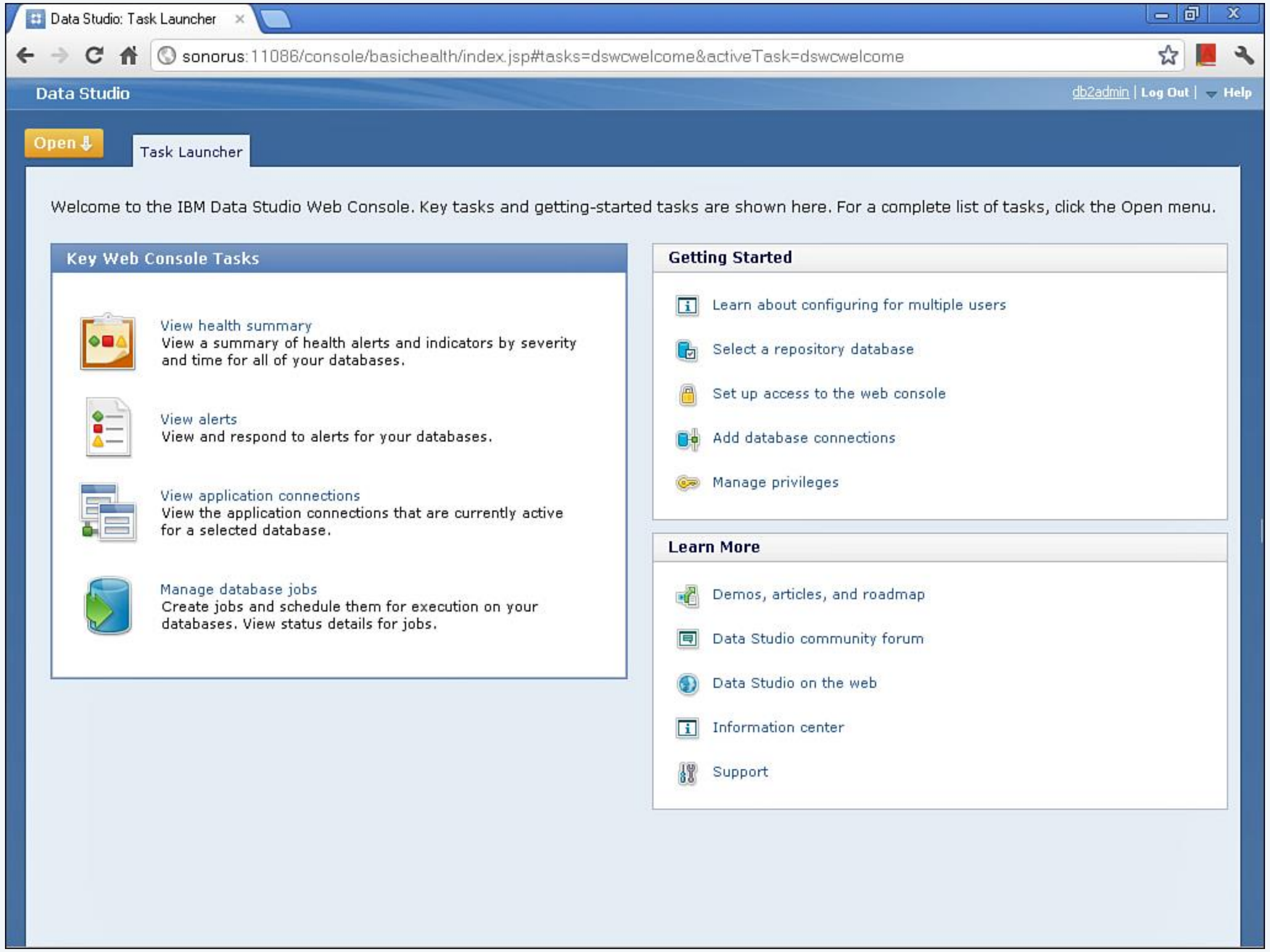
GUIs in the past

- 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


Data Studio: Task Launcher
sonorus:11086/console/basichealth/index.jsp#tasks=dswcwelcome&activeTask=dswcwelcome
Data Studio
db2admin | Log Out | Help
Open
Task Launcher

Welcome to the IBM Data Studio Web Console. Key tasks and getting-started tasks are shown here. For a complete list of tasks, click the Open menu.

Key Web Console Tasks

View health summary
View a summary of health alerts and indicators by severity and time for all of your databases.

View alerts
View and respond to alerts for your databases.

View application connections
View the application connections that are currently active for a selected database.

Manage database jobs
Create jobs and schedule them for execution on your databases. View status details for jobs.

Getting Started

- Learn about configuring for multiple users
- Select a repository database
- Set up access to the web console
- Add database connections
- Manage privileges

Learn More

- Demos, articles, and roadmap
- Data Studio community forum
- Data Studio on the web
- Information center
- Support

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 Technolgies (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

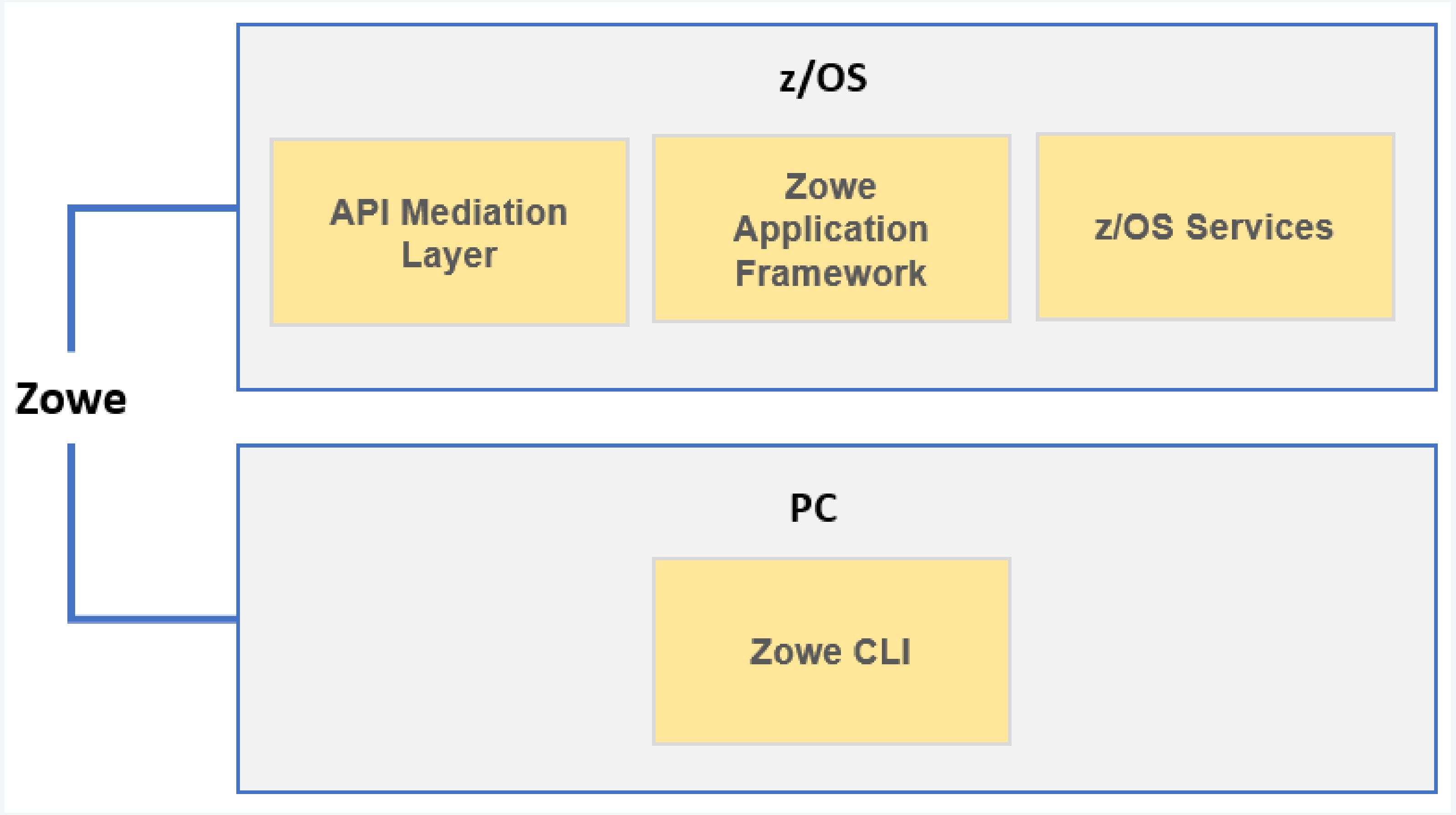
- Application Developers
- System Programmers
- DBAs
- DevOps Architects

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-existance 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

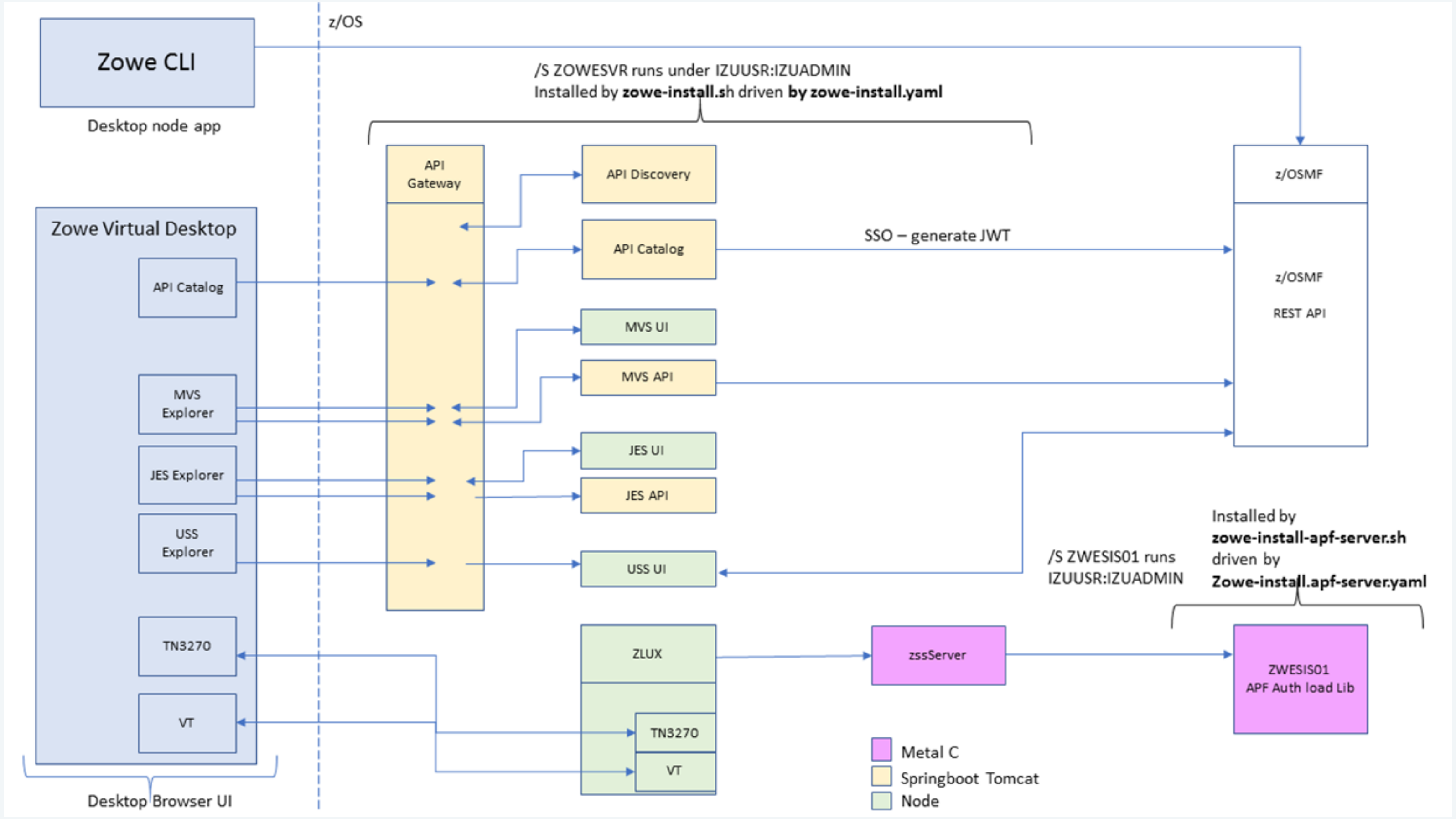
Zowe differentiation to prior GUIs

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

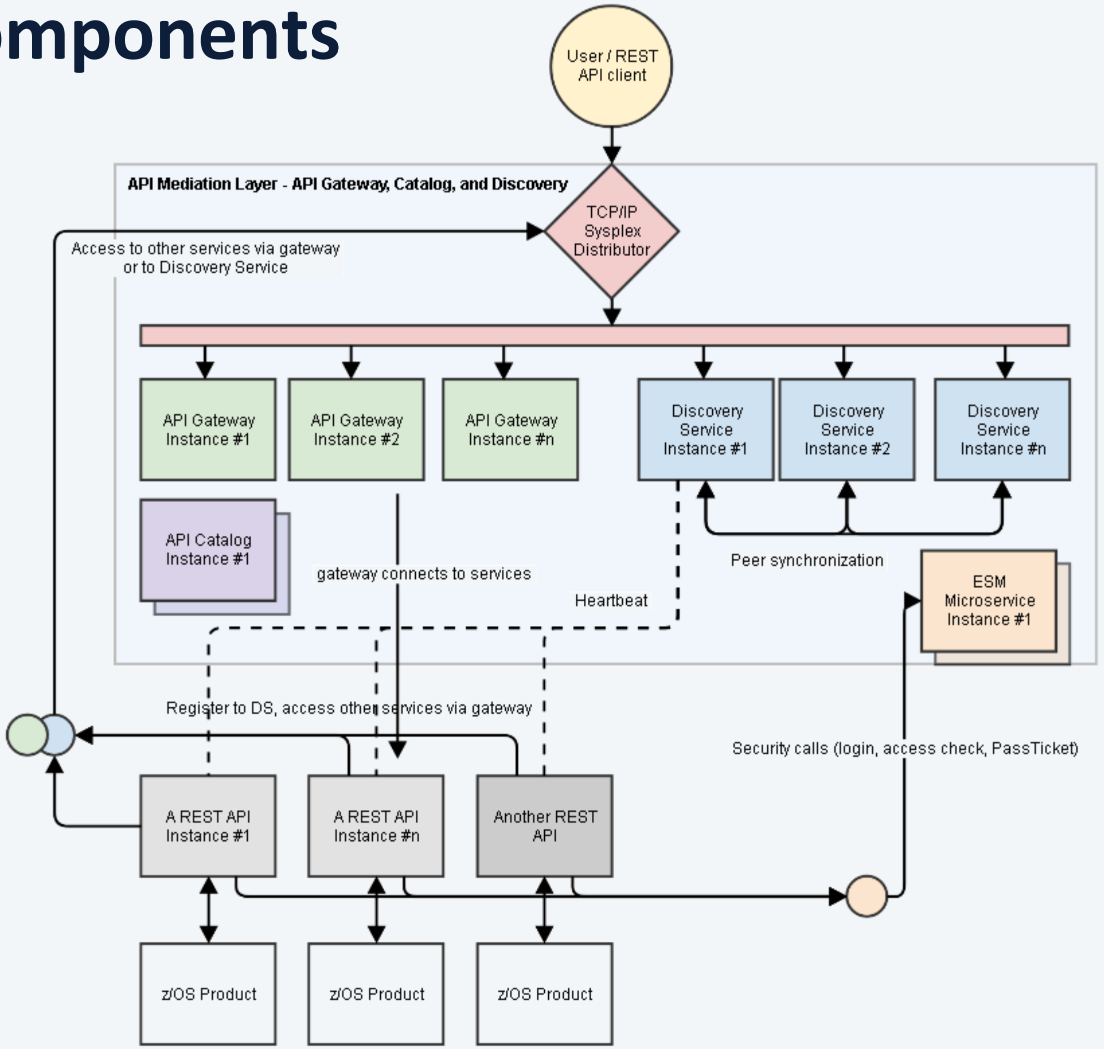
- **Interact with files:**
  Create, edit, download, and upload data sets
- **Submit jobs:**
  Submit JCL from data sets or local storage, monitor the status, and view/download the output
- **Execute commands:**
  Issue TSO, or z/OS console commands
- **Integrated scripts:**
  Define scripts that do 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

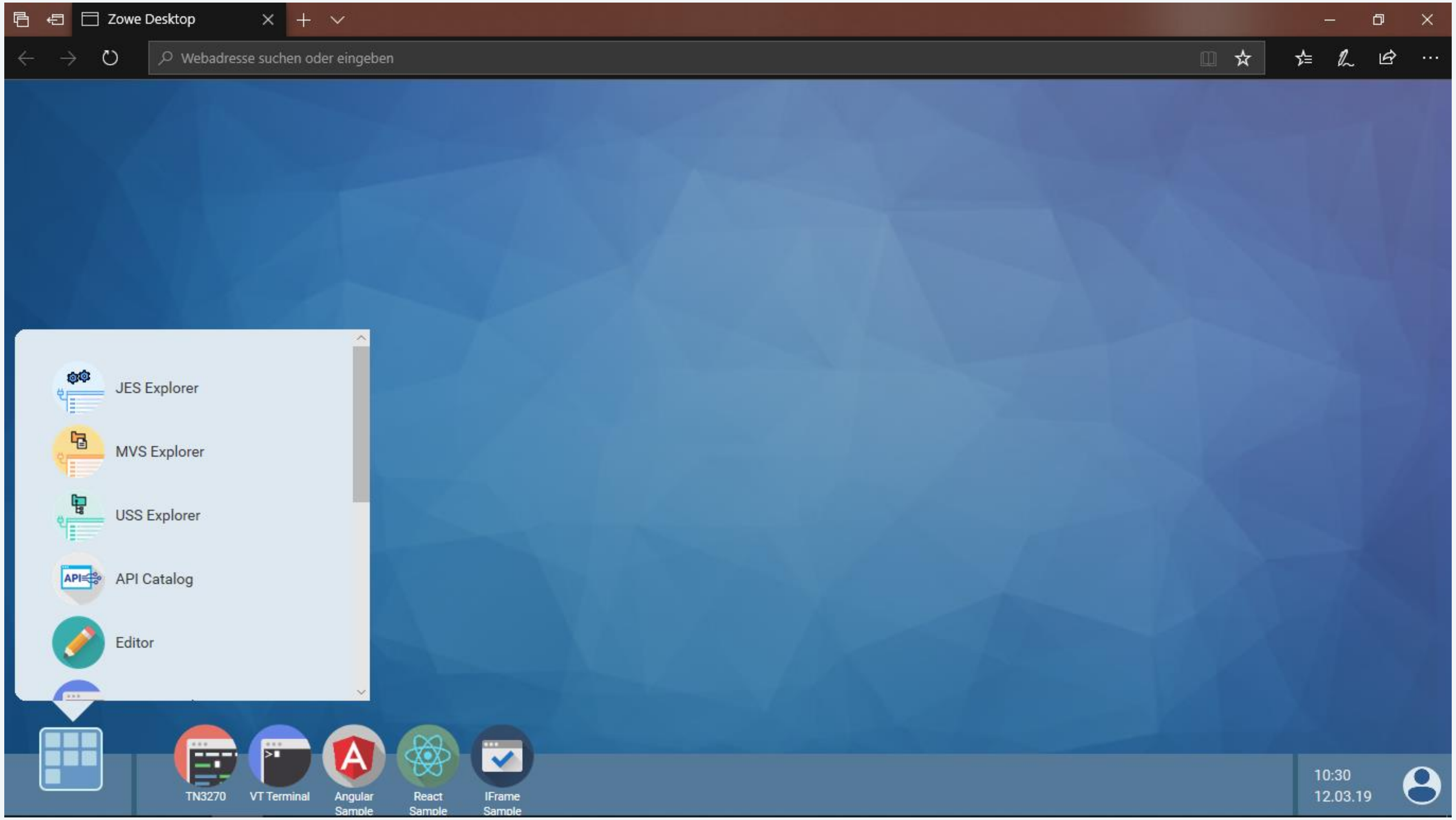
Zowe components


User / REST API client
API Mediation Layer - API Gateway, Catalog, and Discovery
TCP/IP Sysplex Distributor
Access to other services via gateway or to Discovery Service
API Gateway Instance #1
API Gateway Instance #2
API Gateway Instance #n
Discovery Service Instance #1
Discovery Service Instance #2
Discovery Service Instance #n
API Catalog Instance #1
gateway connects to services
Heartbeat
Peer synchronization
ESM Microservice Instance #1
Register to DS, access other services via gateway
Security calls (login, access check, PassTicket)
A REST API Instance #1
A REST API Instance #n
Another REST API
z/OS Product
z/OS Product
z/OS Product

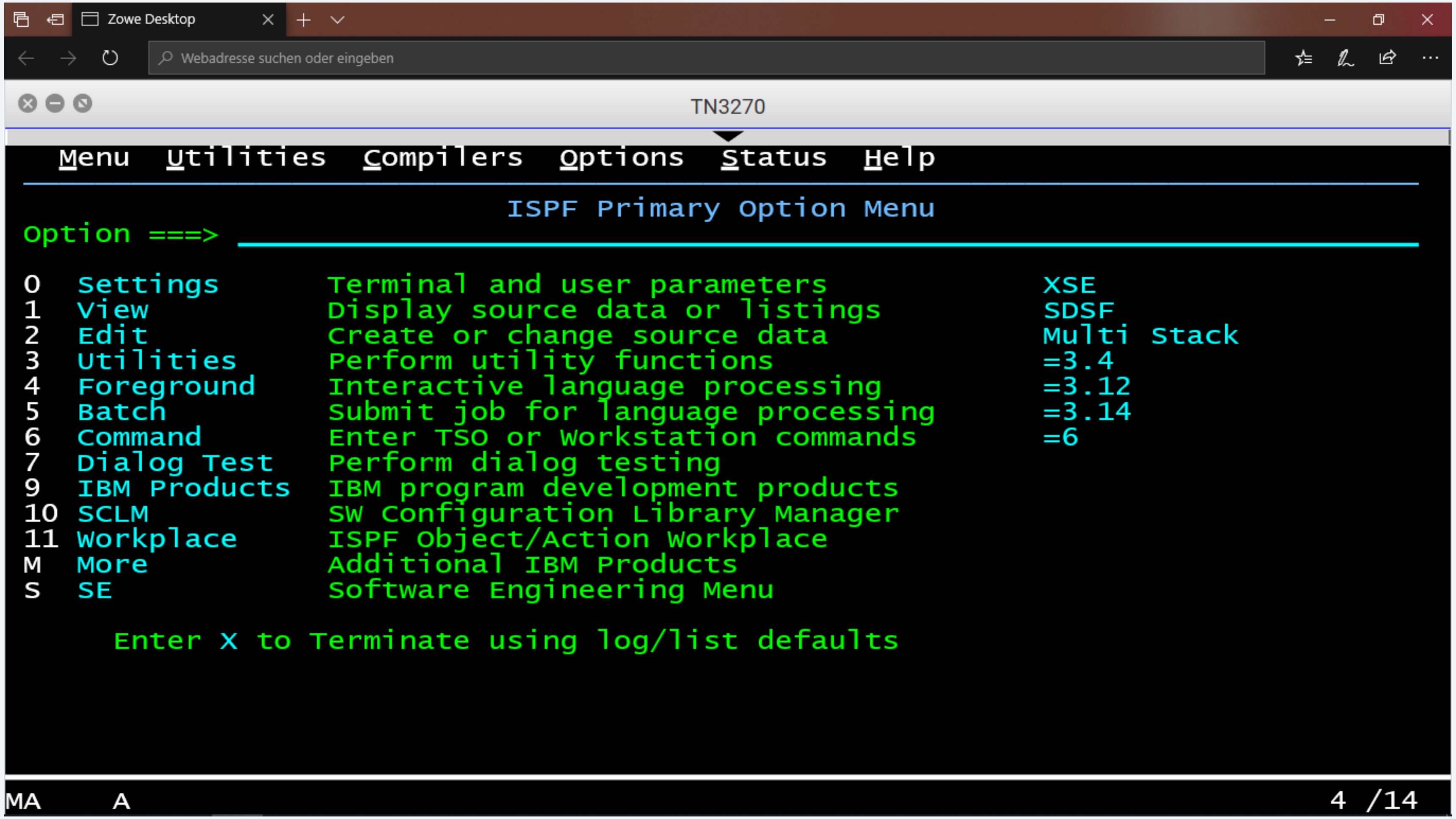
Zowe components @ github.com

- 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-fvt - 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
- tn3270-ng2 - A TN3270 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 transfering 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-package - 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

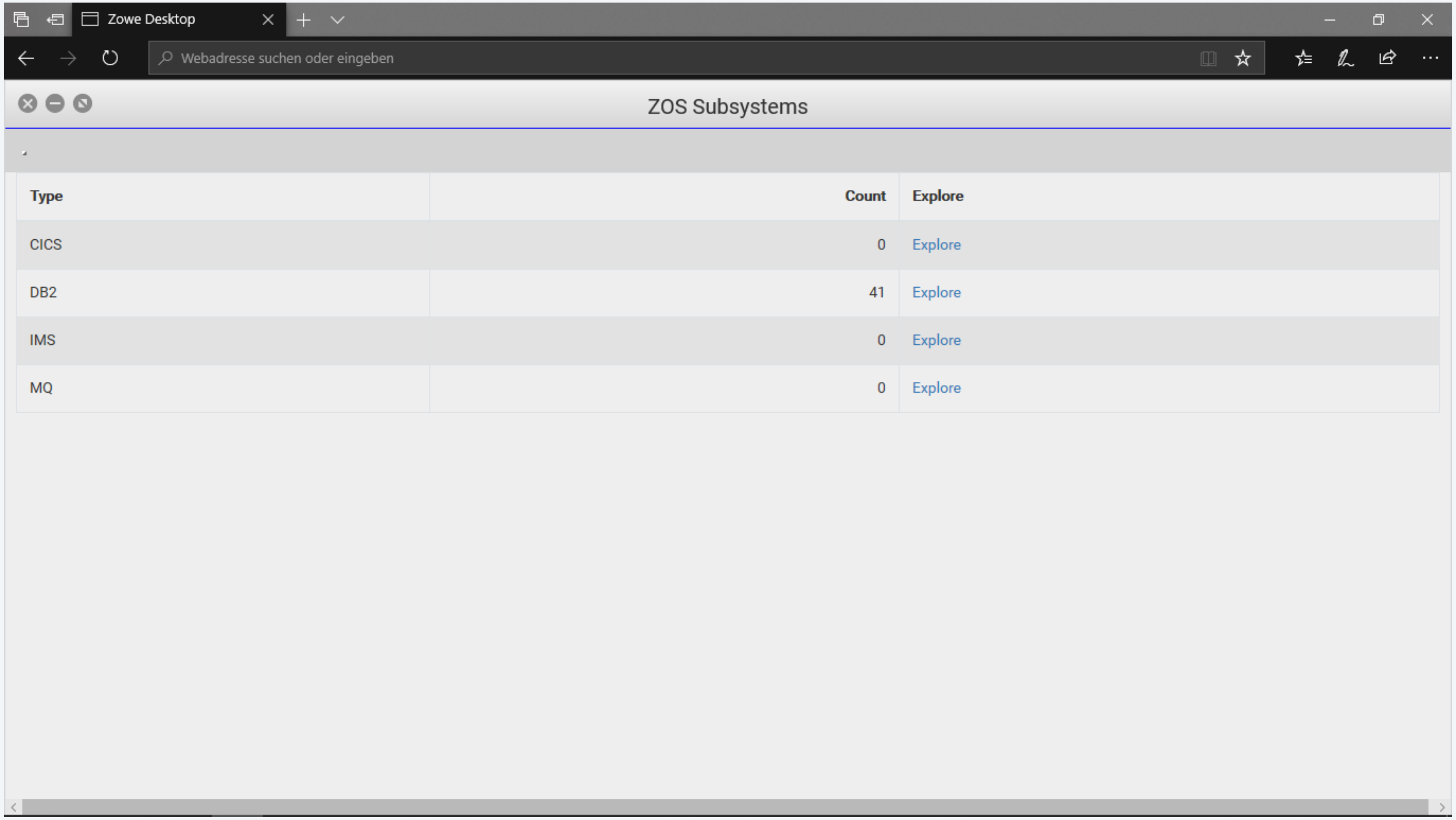


Zowe examples – the tn3270 app ☺



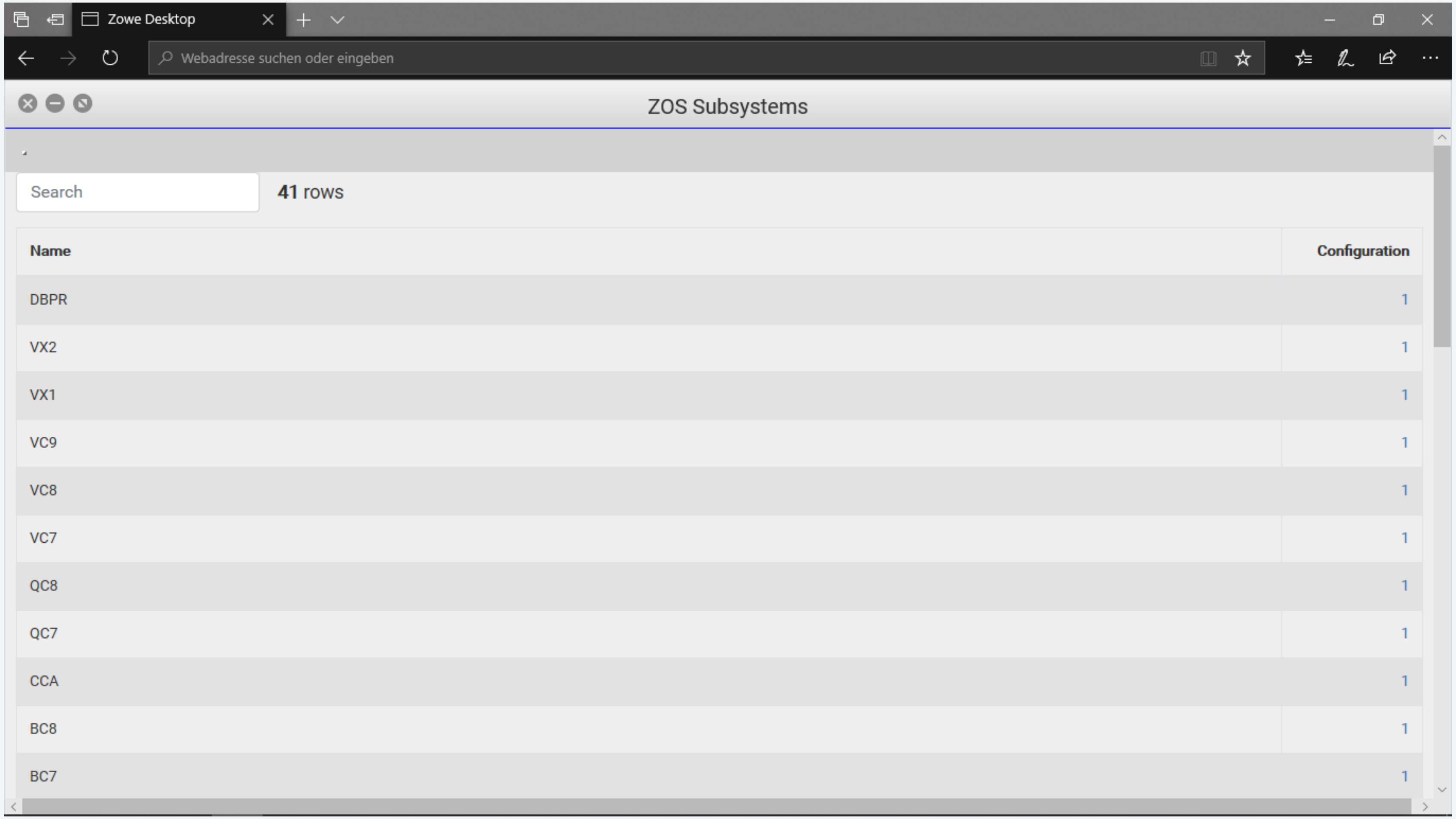
```
   Menu  Utilities  Compilers  Options  Status  Help
                          ISPF Primary Option Menu
Option ===> 

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



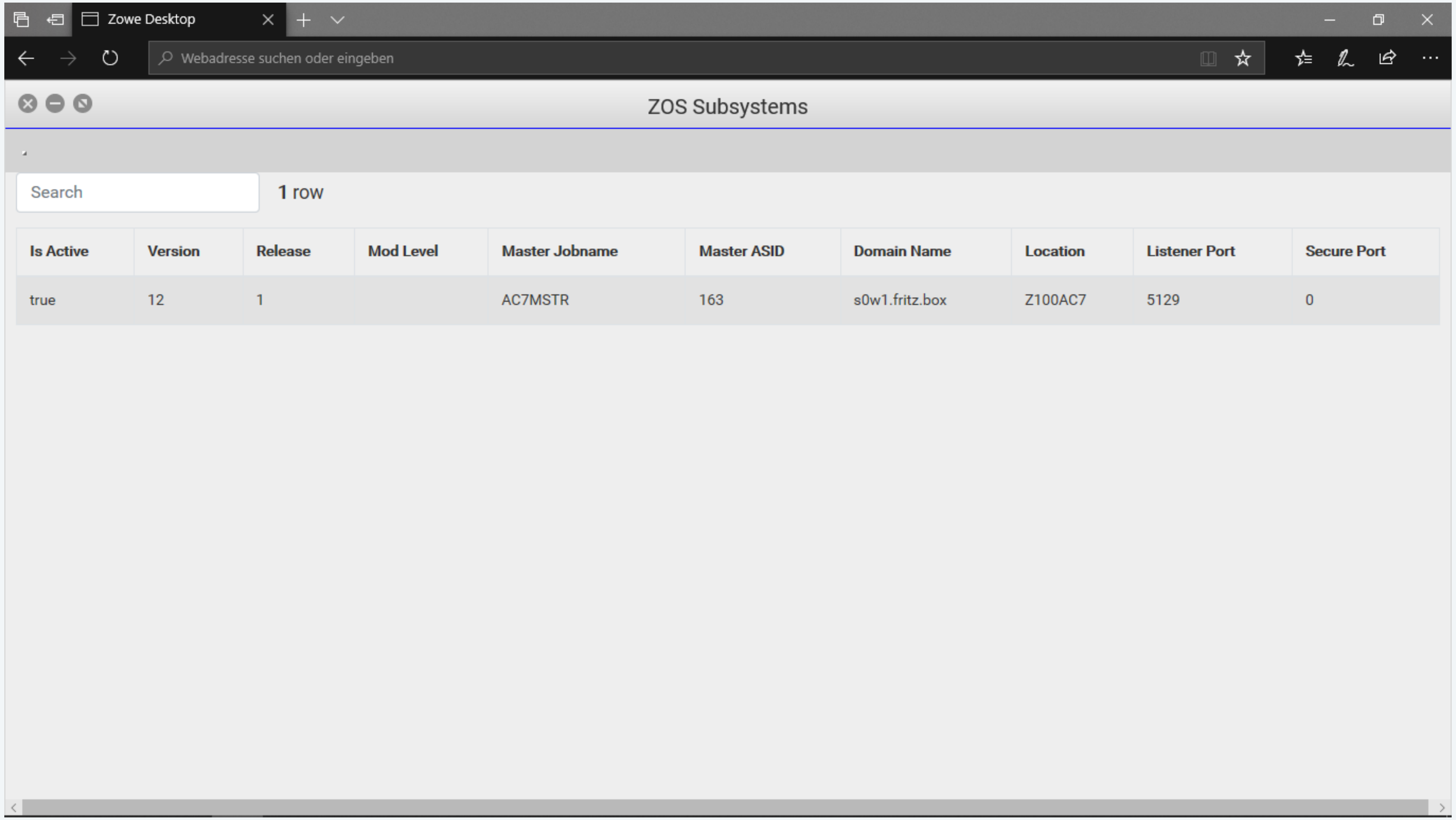
Type	Count	Explore
CICS	0	Explore
DB2	41	Explore
IMS	0	Explore
MQ	0	Explore

Zowe examples – z/OS Subsystems



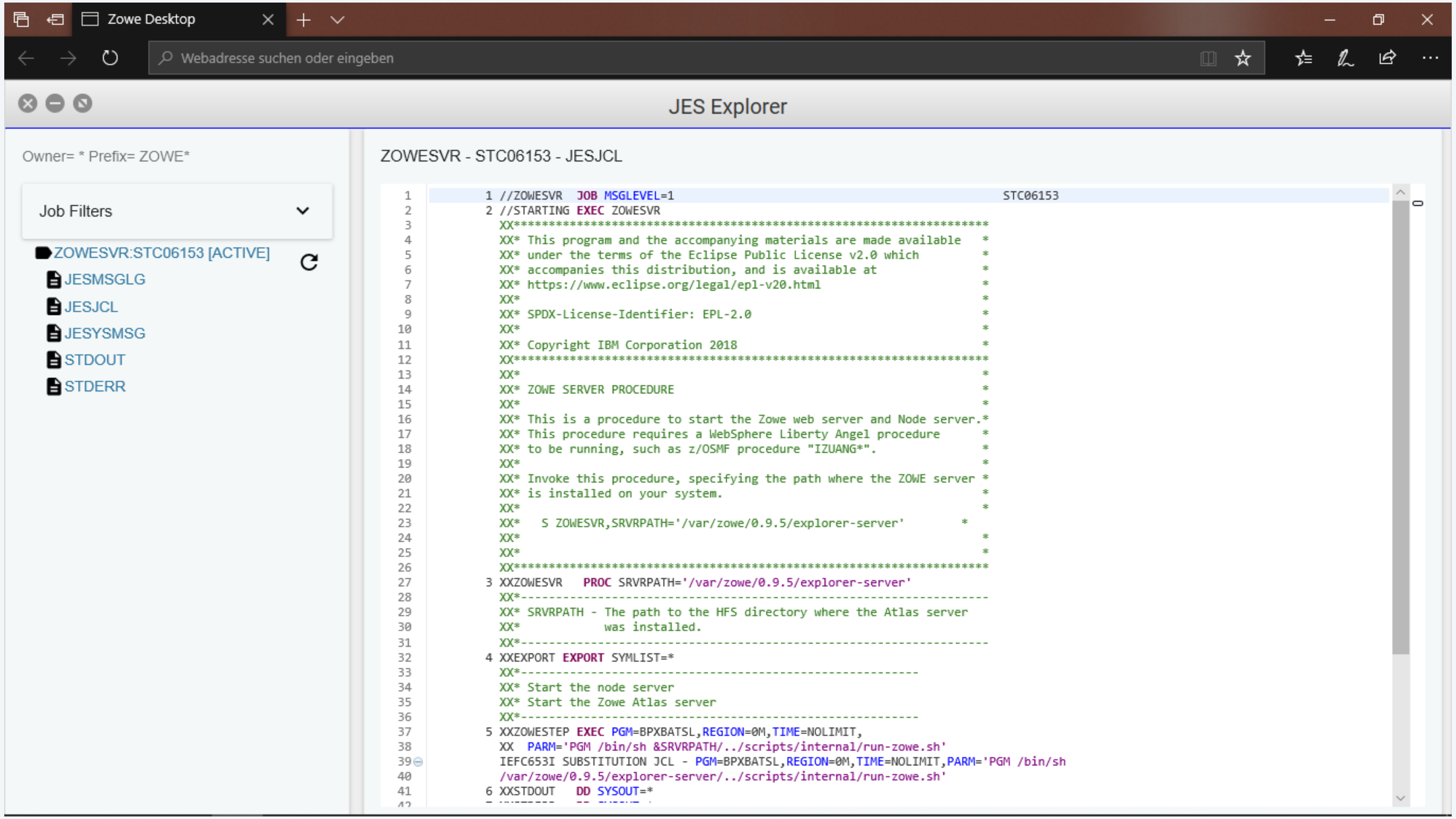
Name	Configuration
DBPR	1
VX2	1
VX1	1
VC9	1
VC8	1
VC7	1
QC8	1
QC7	1
CCA	1
BC8	1
BC7	1

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

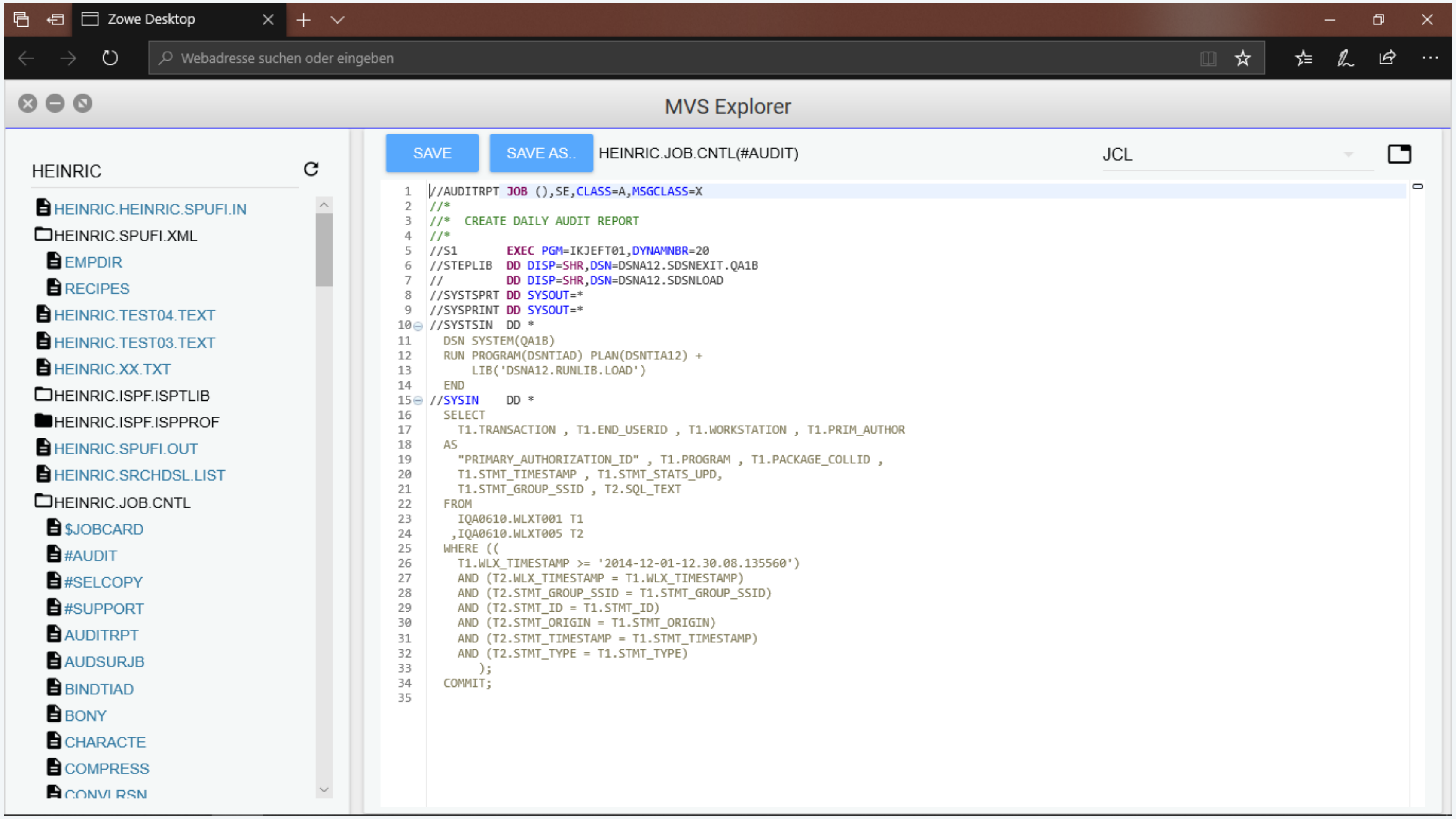
Zowe examples – the JES Explorer



ZOWESVR - STC06153 - JESJCL

```
        1 //ZOWESVR  JOB MSGLEVEL=1                                          STC06153
        2 //STARTING EXEC ZOWESVR
          XX********************************************************************
          XX* This program and the accompanying materials are made available   *
          XX* under the terms of the Eclipse Public License v2.0 which         *
          XX* accompanies this distribution, and is available at              *
          XX* https://www.eclipse.org/legal/epl-v20.html                       *
          XX*                                                                  *
          XX* SPDX-License-Identifier: EPL-2.0                                 *
          XX*                                                                  *
          XX* Copyright IBM Corporation 2018                                   *
          XX********************************************************************
          XX*                                                                  *
          XX* ZOWE SERVER PROCEDURE                                            *
          XX*                                                                  *
          XX* This is a procedure to start the Zowe web server and Node server.*
          XX* This procedure requires a WebSphere Liberty Angel procedure      *
          XX* to be running, such as z/OSMF procedure "IZUANG*".               *
          XX*                                                                  *
          XX* Invoke this procedure, specifying the path where the ZOWE server *
          XX* is installed on your system.                                     *
          XX*                                                                  *
          XX*   S ZOWESVR,SRVRPATH='/var/zowe/0.9.5/explorer-server'         *
          XX*                                                                  *
          XX*                                                                  *
          XX********************************************************************
        3 XXZOWESVR   PROC SRVRPATH='/var/zowe/0.9.5/explorer-server'
          XX*-------------------------------------------------------------------
          XX* SRVRPATH - The path to the HFS directory where the Atlas server
          XX*            was installed.
          XX*-------------------------------------------------------------------
        4 XXEXPORT EXPORT SYMLIST=*
          XX*---------------------------------------------------------
          XX* Start the node server
          XX* Start the Zowe Atlas server
          XX*---------------------------------------------------------
        5 XXZOWESTEP EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
          XX  PARM='PGM /bin/sh &SRVRPATH/../scripts/internal/run-zowe.sh'
          IEFC653I SUBSTITUTION JCL - PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,PARM='PGM /bin/sh
          /var/zowe/0.9.5/explorer-server/../scripts/internal/run-zowe.sh'
        6 XXSTDOUT   DD SYSOUT=*
```

Zowe examples – the MVS Explorer

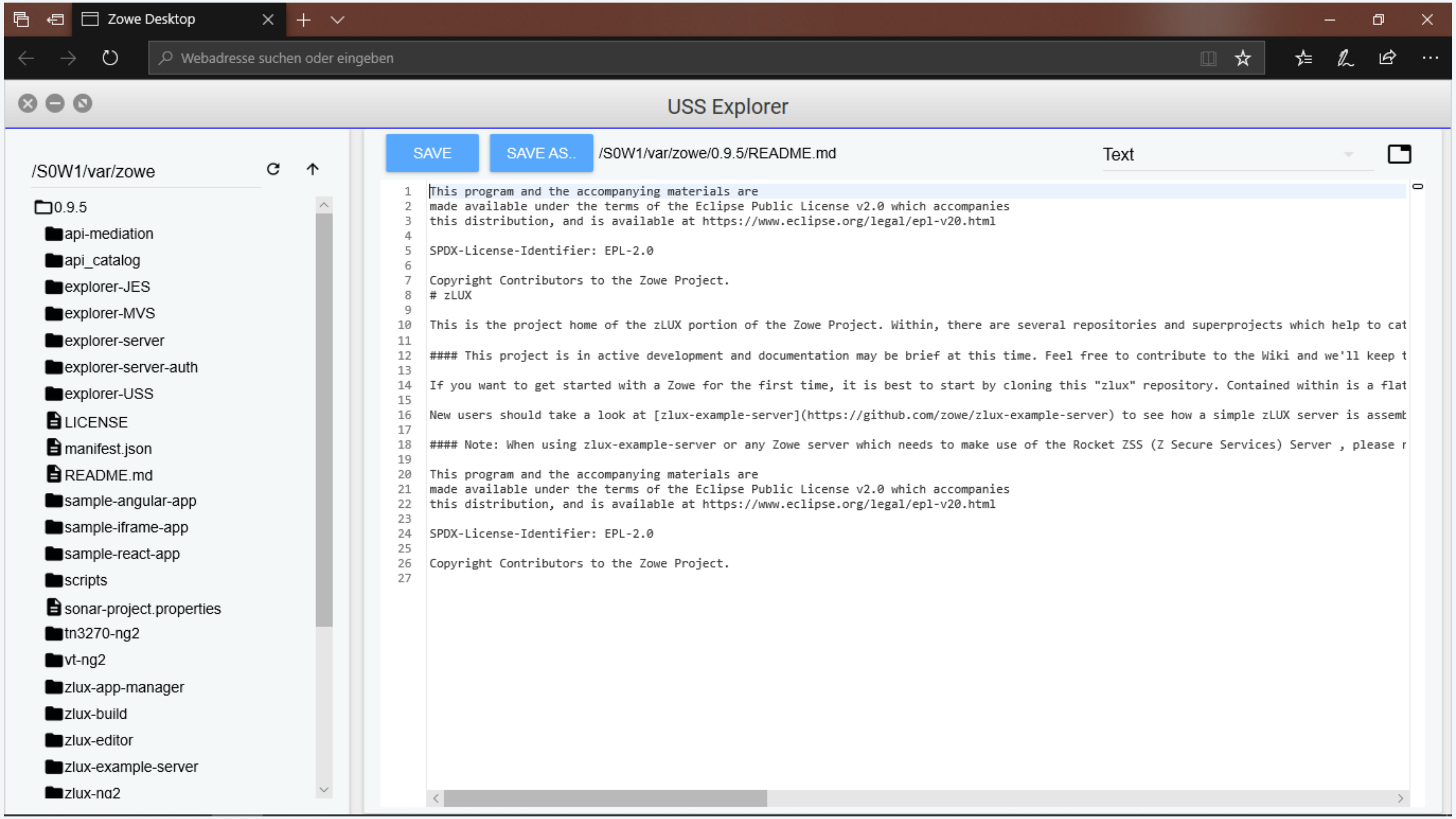


HEINRIC.JOB.CNTL(#AUDIT)

- HEINRIC.HEINRIC.SPUFI.IN
- HEINRIC.SPUFI.XML
  - EMPDIR
  - RECIPES
- HEINRIC.TEST04.TEXT
- HEINRIC.TEST03.TEXT
- HEINRIC.XX.TXT
- HEINRIC.ISPF.ISPTLIB
- HEINRIC.ISPF.ISPPROF
- HEINRIC.SPUFI.OUT
- HEINRIC.SRCHDSL.LIST
- HEINRIC.JOB.CNTL
  - $JOBCARD
  - #AUDIT
  - #SELCOPY
  - #SUPPORT
  - AUDITRPT
  - AUDSURJB
  - BINDTIAD
  - BONY
  - CHARACTE
  - COMPRESS
  - CONVLRSN

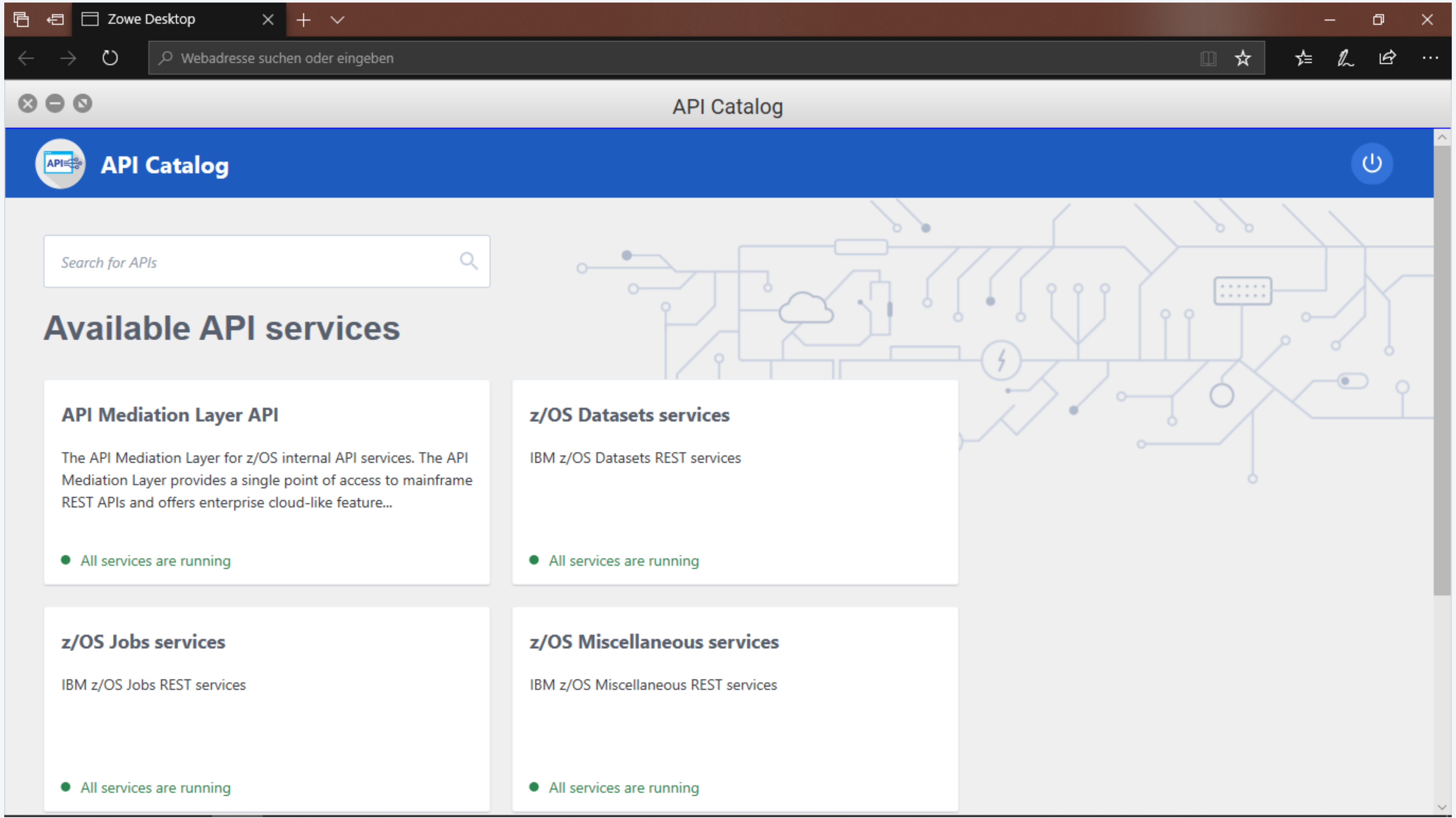
```
//AUDITRPT JOB (),SE,CLASS=A,MSGCLASS=X
//*
//*  CREATE DAILY AUDIT REPORT
//*
//S1       EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD DISP=SHR,DSN=DSNA12.SDSNEXIT.QA1B
//         DD DISP=SHR,DSN=DSNA12.SDSNLOAD
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSIN  DD *
  DSN SYSTEM(QA1B)
  RUN PROGRAM(DSNTIAD) PLAN(DSNTIA12) +
      LIB('DSNA12.RUNLIB.LOAD')
  END
//SYSIN    DD *
  SELECT
    T1.TRANSACTION , T1.END_USERID , T1.WORKSTATION , T1.PRIM_AUTHOR
  AS
    "PRIMARY_AUTHORIZATION_ID" , T1.PROGRAM , T1.PACKAGE_COLLID ,
    T1.STMT_TIMESTAMP , T1.STMT_STATS_UPD,
    T1.STMT_GROUP_SSID , T2.SQL_TEXT
  FROM
    IQA0610.WLXT001 T1
   ,IQA0610.WLXT005 T2
  WHERE ((
    T1.WLX_TIMESTAMP >= '2014-12-01-12.30.08.135560')
    AND (T2.WLX_TIMESTAMP = T1.WLX_TIMESTAMP)
    AND (T2.STMT_GROUP_SSID = T1.STMT_GROUP_SSID)
    AND (T2.STMT_ID = T1.STMT_ID)
    AND (T2.STMT_ORIGIN = T1.STMT_ORIGIN)
    AND (T2.STMT_TIMESTAMP = T1.STMT_TIMESTAMP)
    AND (T2.STMT_TYPE = T1.STMT_TYPE)
       );
  COMMIT;
```

Zowe examples – the USS Explorer



```
This program and the accompanying materials are
made available under the terms of the Eclipse Public License v2.0 which accompanies
this distribution, and is available at https://www.eclipse.org/legal/epl-v20.html

SPDX-License-Identifier: EPL-2.0

Copyright Contributors to the Zowe Project.
# zLUX

This is the project home of the zLUX portion of the Zowe Project. Within, there are several repositories and superprojects which help to cat

#### 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

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

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 assemb

#### 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

This program and the accompanying materials are
made available under the terms of the Eclipse Public License v2.0 which accompanies
this distribution, and is available at https://www.eclipse.org/legal/epl-v20.html

SPDX-License-Identifier: EPL-2.0

Copyright Contributors to the Zowe Project.

```

Zowe examples – the API Catalog


Zowe Desktop
Webadresse suchen oder eingeben
API Catalog
API Catalog
Search for APIs

Available API services

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

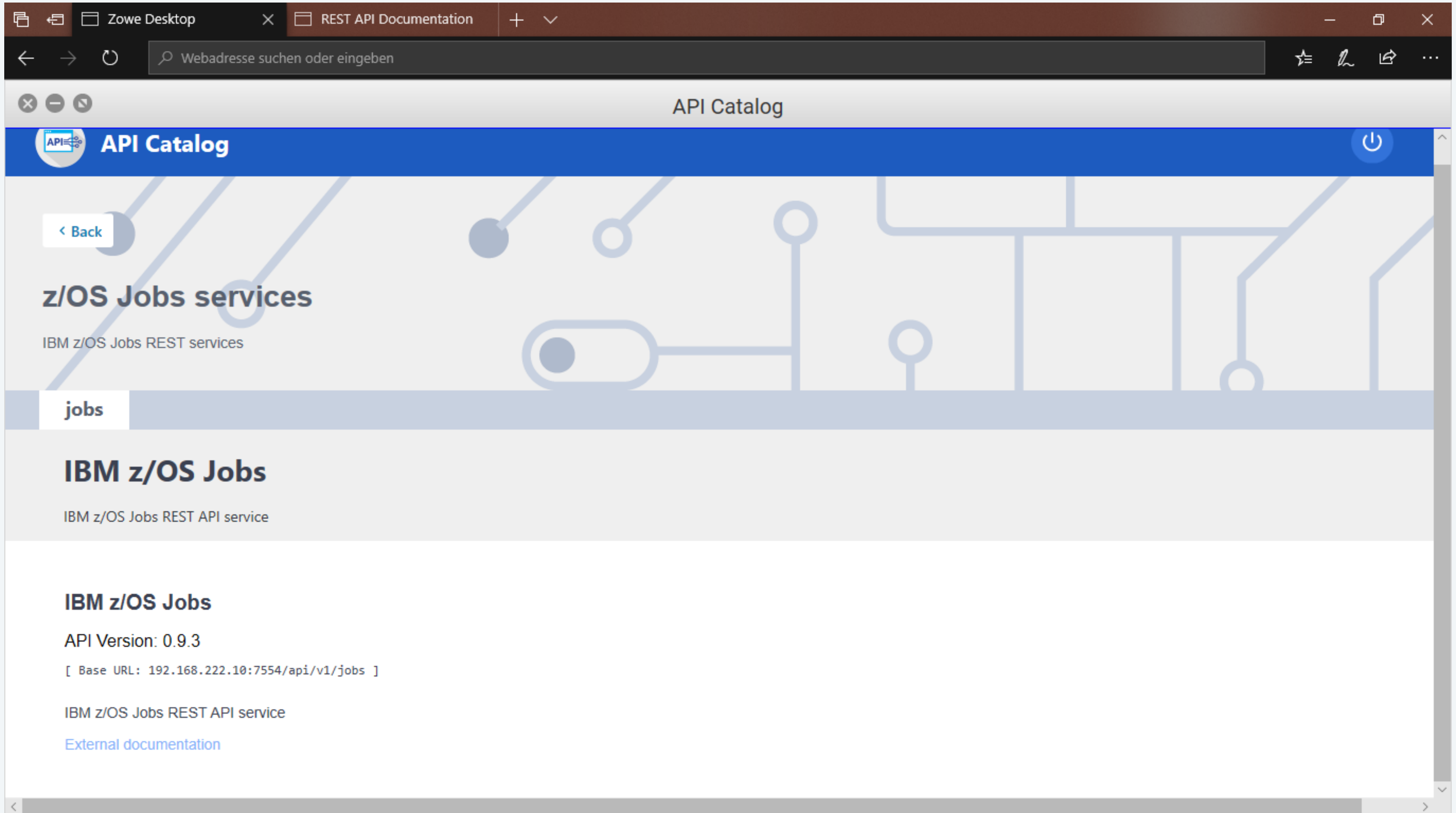
● All services are running

z/OS Miscellaneous services

IBM z/OS Miscellaneous REST services

● All services are running

Zowe examples – the API Catalog



z/OS Jobs services

IBM z/OS Jobs REST services

IBM z/OS Jobs

IBM z/OS Jobs REST API service

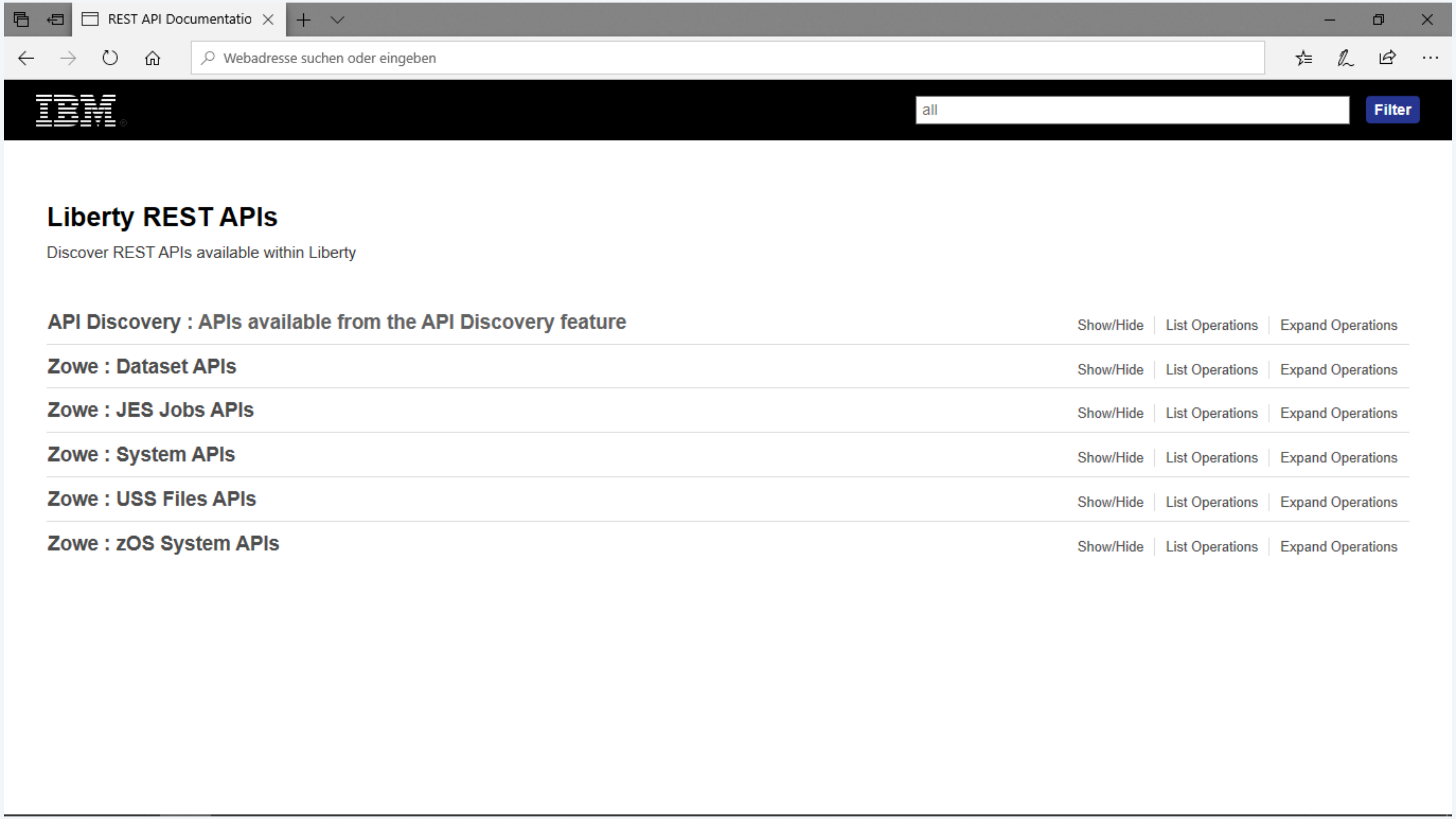
IBM z/OS Jobs

API Version: 0.9.3

```
[ Base URL: 192.168.222.10:7554/api/v1/jobs ]
```

IBM z/OS Jobs REST API service

Zowe examples – the API Catalog

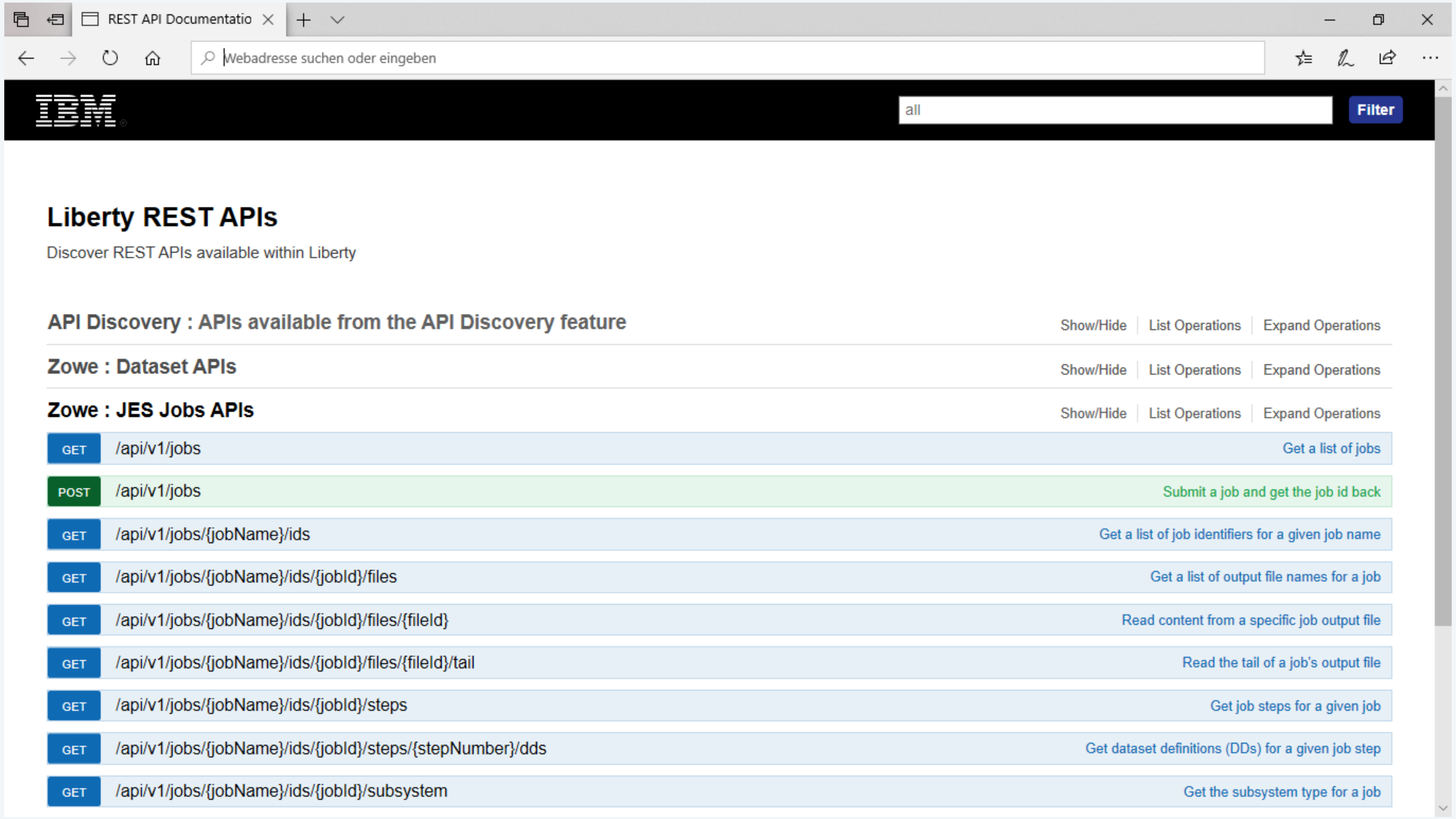


Liberty REST APIs

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


REST API Documentatio

Webadresse suchen oder eingeben

IBM

all Filter

Liberty REST APIs

Discover REST APIs available within Liberty

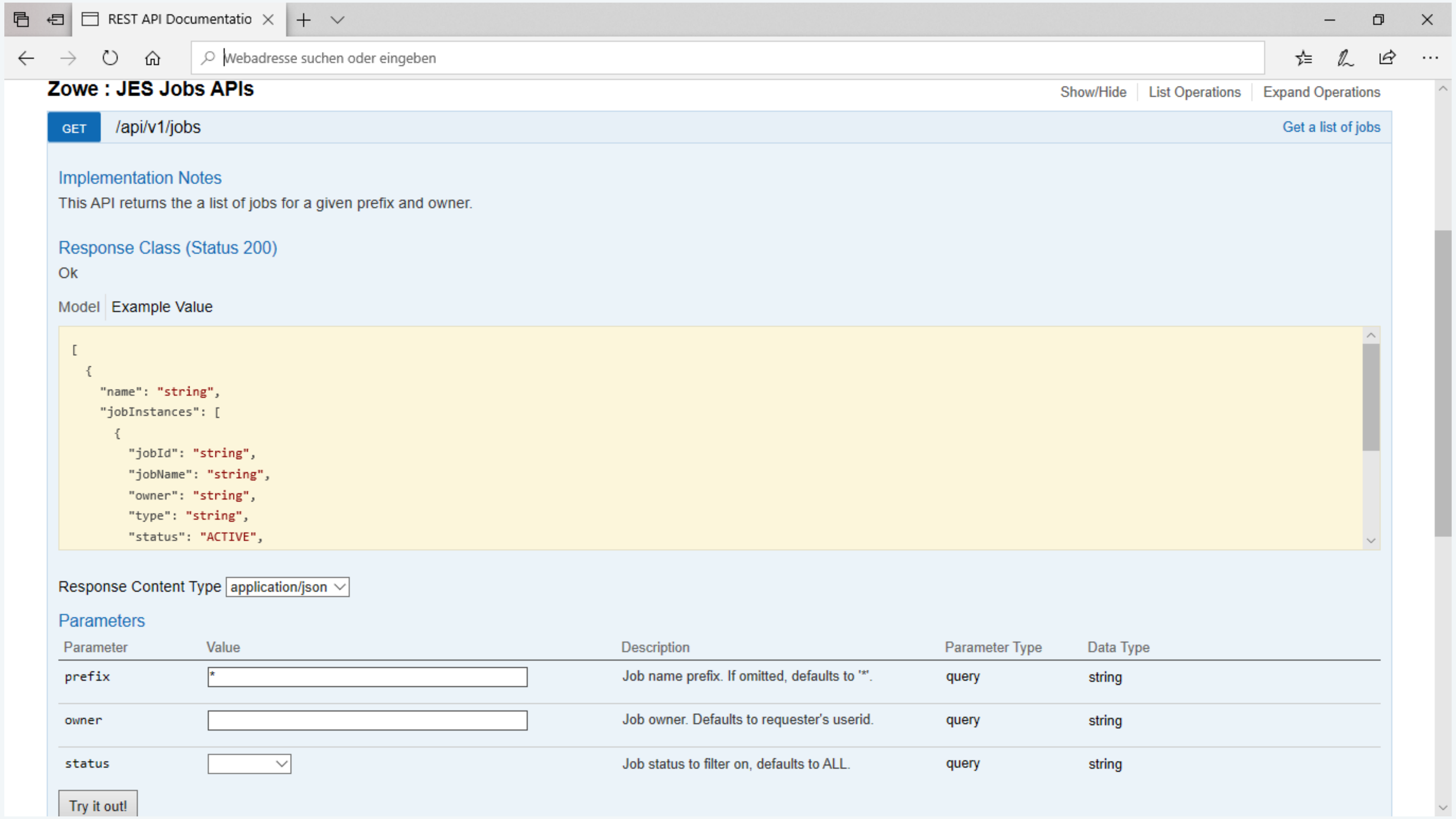
API Discovery : APIs available from the API Discovery feature Show/Hide | List Operations | Expand Operations

Zowe : Dataset APIs Show/Hide | List Operations | Expand Operations

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

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



REST API Documentatio

Webadresse suchen oder eingeben

Zowe : JES Jobs APIs

Show/Hide | List Operations | Expand Operations

GET /api/v1/jobs — Get a list of jobs

Implementation Notes

This API returns the a list of jobs for a given prefix and owner.

Response Class (Status 200)

Ok

Model | Example Value

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

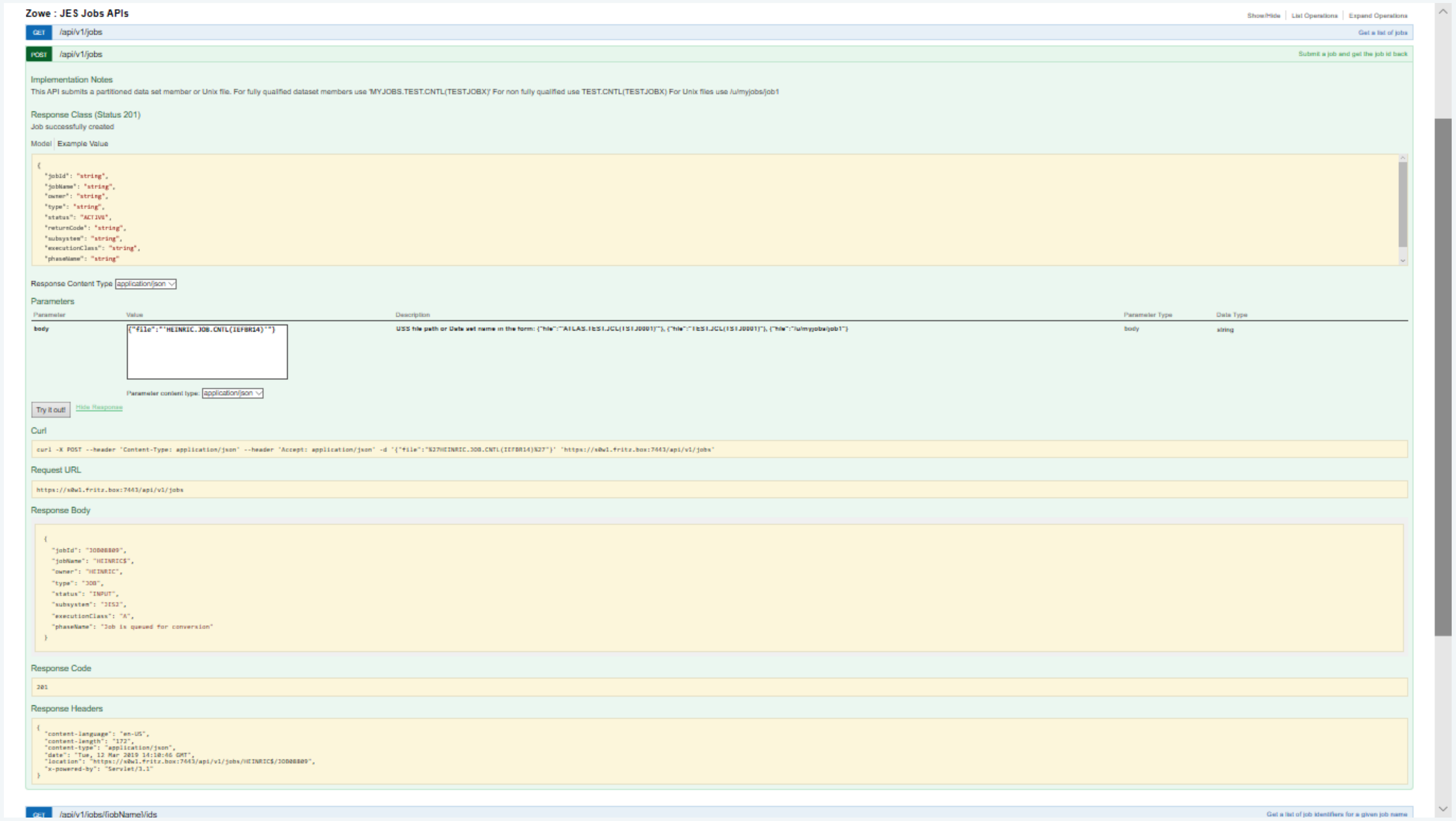
Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
prefix	*	Job name prefix. If omitted, defaults to '*'.	query	string
owner		Job owner. Defaults to requester's userid.	query	string
status		Job status to filter on, defaults to ALL.	query	string

Try it out!

Zowe examples – the API Catalog



Zowe : JES Jobs APIs

GET /api/v1/jobs — Get a list of jobs

POST /api/v1/jobs — Submit a job and get the job id 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": "ACTIVE",
  "returnCode": "string",
  "subsystem": "string",
  "executionClass": "string",
  "phaseName": "string"
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
body	{"file":"'HEINRIC.JOB.CNTL(IEFBR14)'"}	USS file path or Data set name in the form: {"file":"'ATLAS.TEST.JCL(TSTJ0001)'"}, {"file":"TEST.JCL(TSTJ0001)"}, {"file":"/u/myjobs/job1"}	body	string

Parameter content type: application/json

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{"file":"%27HEINRIC.JOB.CNTL(IEFBR14)%27"}' 'https://s0w1.fritz.box:7443/api/v1/jobs'
```

Request URL

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

Response Body

```
{
  "jobId": "JOB08809",
  "jobName": "HEINRIC$",
  "owner": "HEINRIC",
  "type": "JOB",
  "status": "INPUT",
  "subsystem": "JES2",
  "executionClass": "A",
  "phaseName": "Job is queued for conversion"
}
```

Response Code

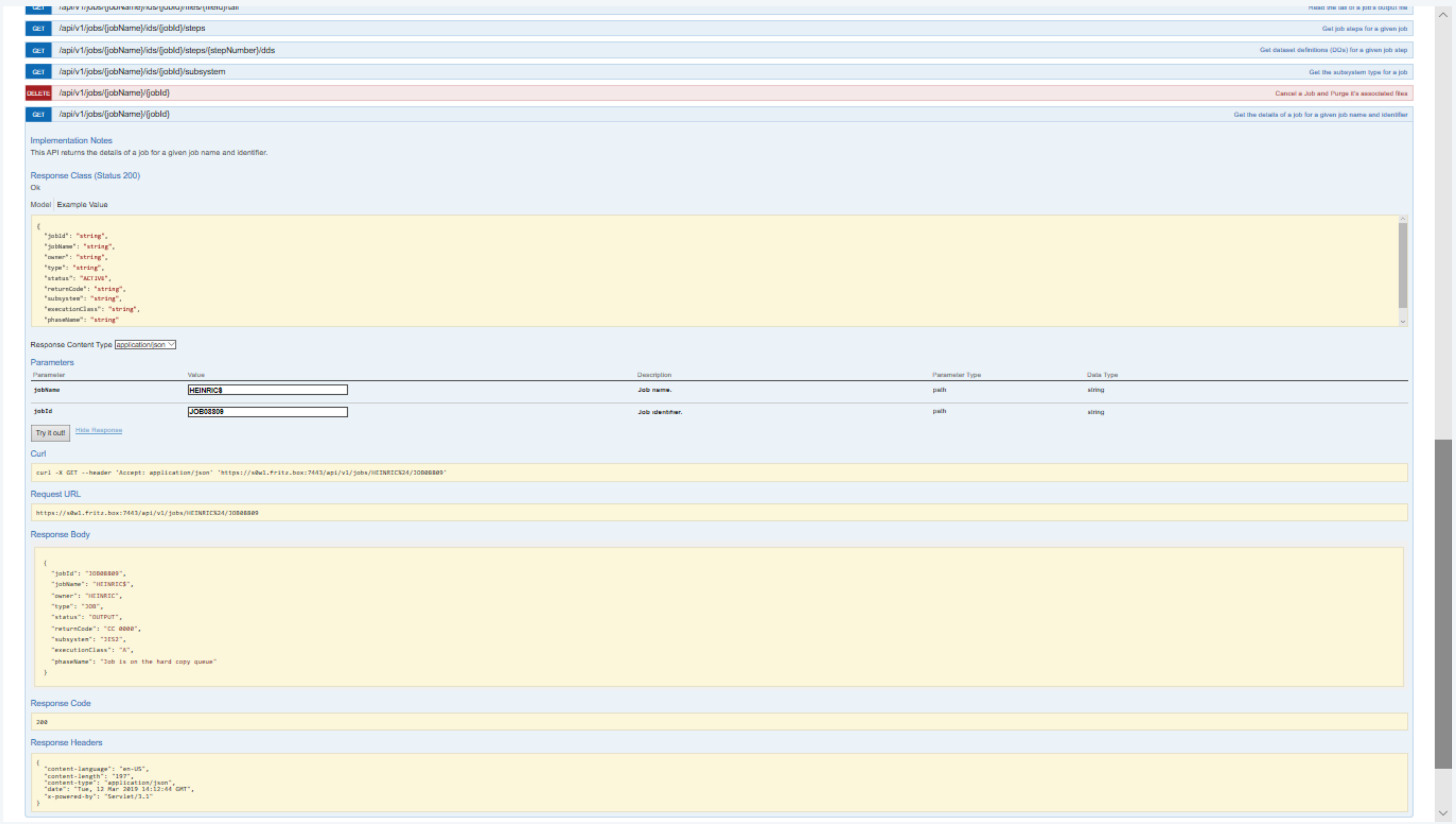
```
201
```

Response Headers

```
{
  "content-language": "en-US",
  "content-length": "172",
  "content-type": "application/json",
  "date": "Tue, 12 Mar 2019 14:10:46 GMT",
  "location": "https://s0w1.fritz.box:7443/api/v1/jobs/HEINRIC$/JOB08809",
  "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



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
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.

Response Class (Status 200)

Ok

Model | Example Value

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

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
jobName	HEINRIC$	Job name.	path	string
jobId	JOB08809	Job identifier.	path	string

Try it out! Hide Response

Curl

```
curl -X GET --header 'Accept: application/json' 'https://s0w1.fritz.box:7443/api/v1/jobs/HEINRIC%24/JOB08809'
```

Request URL

```
https://s0w1.fritz.box:7443/api/v1/jobs/HEINRIC%24/JOB08809
```

Response Body

```
{
  "jobId": "JOB08809",
  "jobName": "HEINRIC$",
  "owner": "HEINRIC",
  "type": "JOB",
  "status": "OUTPUT",
  "returnCode": "CC 0000",
  "subsystem": "JES2",
  "executionClass": "A",
  "phaseName": "Job is on the hard copy queue"
}
```

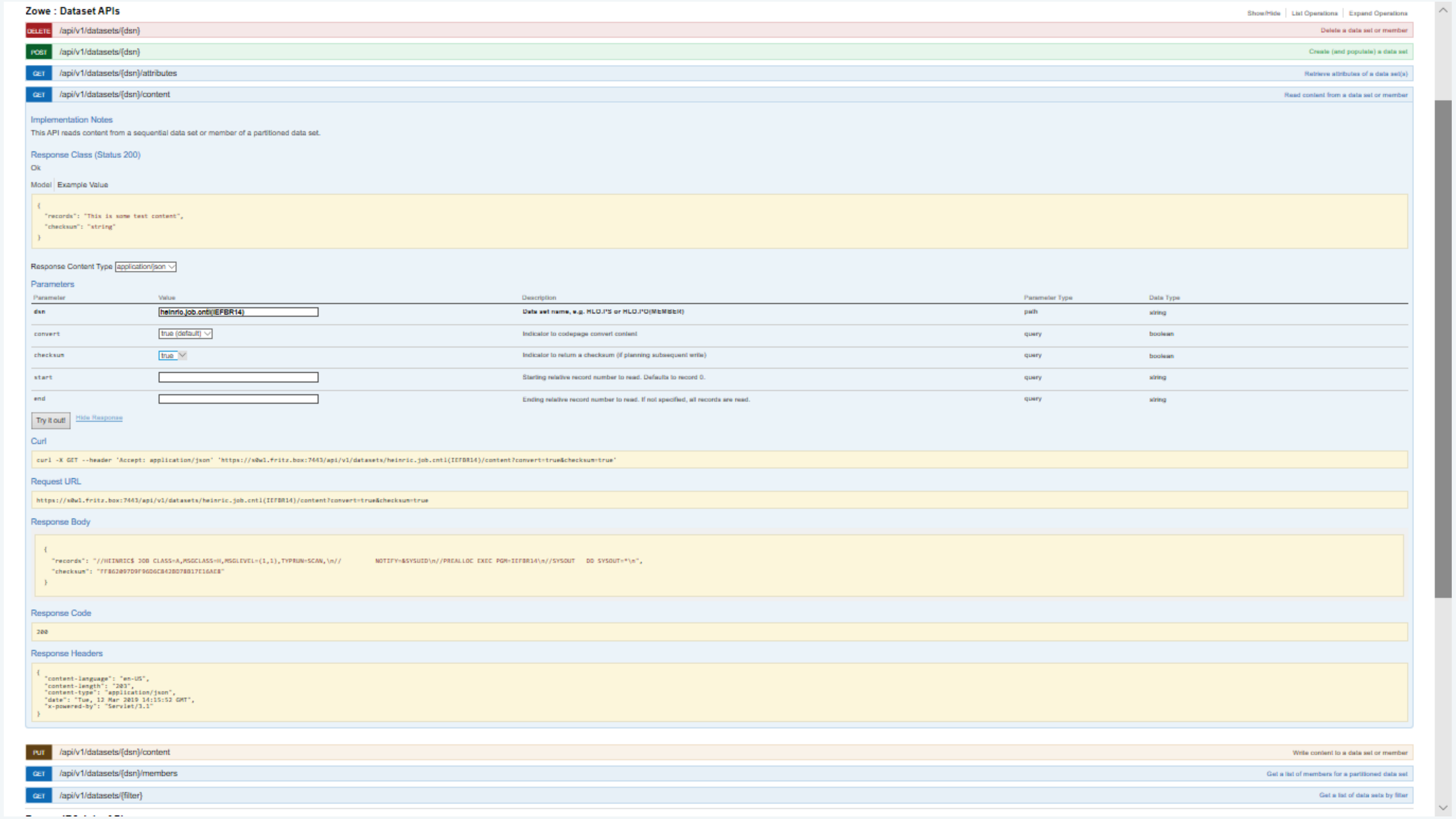
Response Code

```
200
```

Response Headers

```
{
  "content-language": "en-US",
  "content-length": "197",
  "content-type": "application/json",
  "date": "Tue, 12 Mar 2019 14:12:44 GMT",
  "x-powered-by": "Servlet/3.1"
}
```

Zowe examples – the API Catalog



Zowe : Dataset APIs

DELETE	/api/v1/datasets/{dsn}	Delete a data set or member
POST	/api/v1/datasets/{dsn}	Create (and populate) 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 test content",
  "checksum": "string"
}
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
dsn	heinric.job.cntl(IEFBR14)	Data set name, e.g. HLQ.PS or HLQ.PO(MEMBER)	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

Curl

```
curl -X GET --header 'Accept: application/json' 'https://s0w1.fritz.box:7443/api/v1/datasets/heinric.job.cntl(IEFBR14)/content?convert=true&checksum=true'
```

Request URL

```
https://s0w1.fritz.box:7443/api/v1/datasets/heinric.job.cntl(IEFBR14)/content?convert=true&checksum=true
```

Response Body

```
{
  "records": "//HEINRIC$ JOB CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),TYPRUN=SCAN,\n//        NOTIFY=&SYSUID\n//PREALLOC EXEC PGM=IEFBR14\n//SYSOUT   DD SYSOUT=*\n",
  "checksum": "FF862097D9F96D6C8428D78817E16AE8"
}
```

Response Code

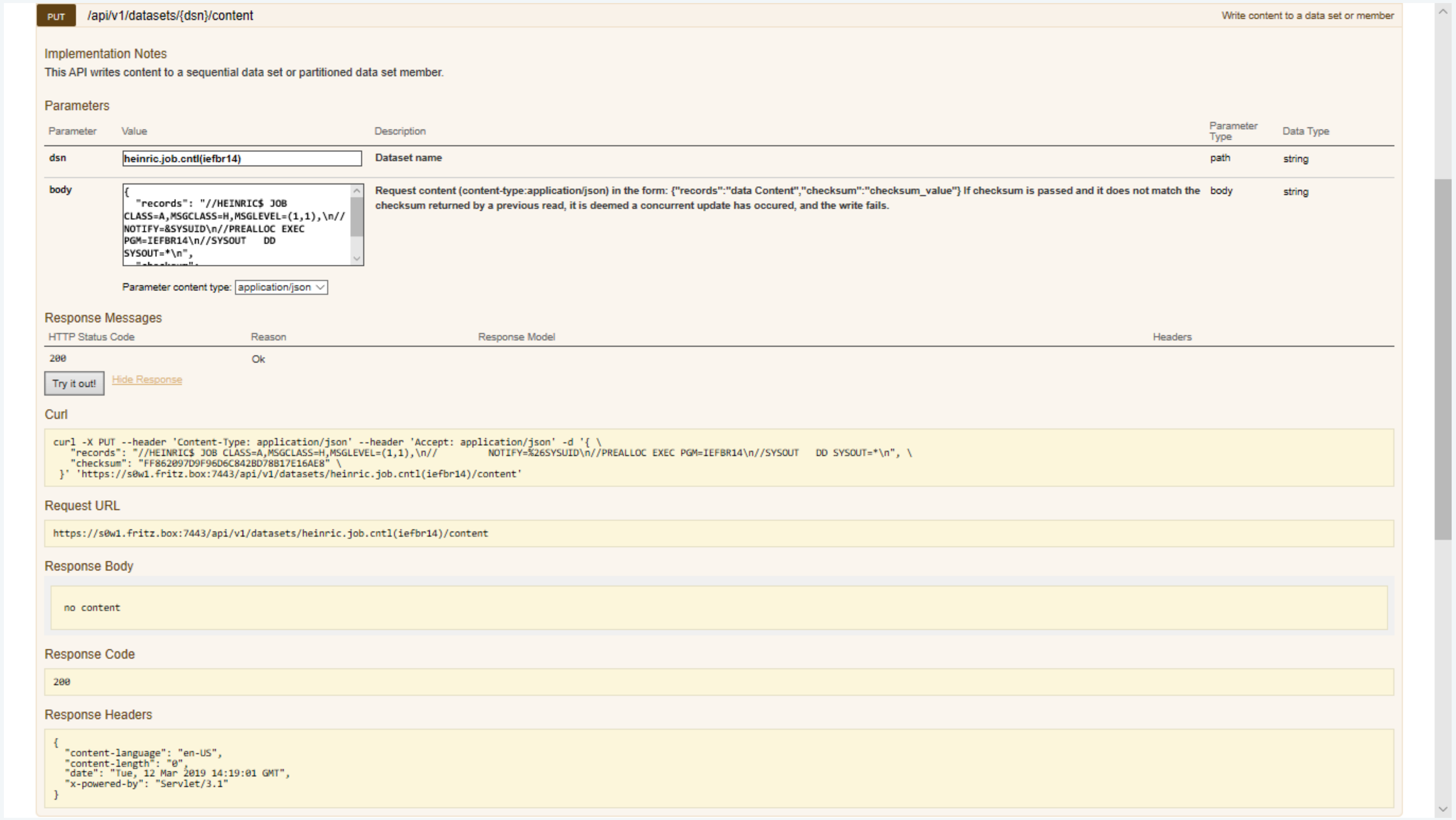
```
200
```

Response Headers

```
{
  "content-language": "en-US",
  "content-length": "203",
  "content-type": "application/json",
  "date": "Tue, 12 Mar 2019 14:15:52 GMT",
  "x-powered-by": "Servlet/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	heinric.job.cntl(iefbr14)	Dataset name	path	string
body	{ "records": "//HEINRIC$ JOB CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),\n// NOTIFY=&SYSUID\n//PREALLOC EXEC PGM=IEFBR14\n//SYSOUT DD SYSOUT=*\n",	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 occured, 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": "//HEINRIC$ JOB CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),\n//         NOTIFY=%26SYSUID\n//PREALLOC EXEC PGM=IEFBR14\n//SYSOUT   DD SYSOUT=*\n", \
   "checksum": "FF862097D9F96D6C842BD78B17E16AE8" \
 }' 'https://s0w1.fritz.box:7443/api/v1/datasets/heinric.job.cntl(iefbr14)/content'
```

Request URL

```
https://s0w1.fritz.box:7443/api/v1/datasets/heinric.job.cntl(iefbr14)/content
```

Response Body

```
no content
```

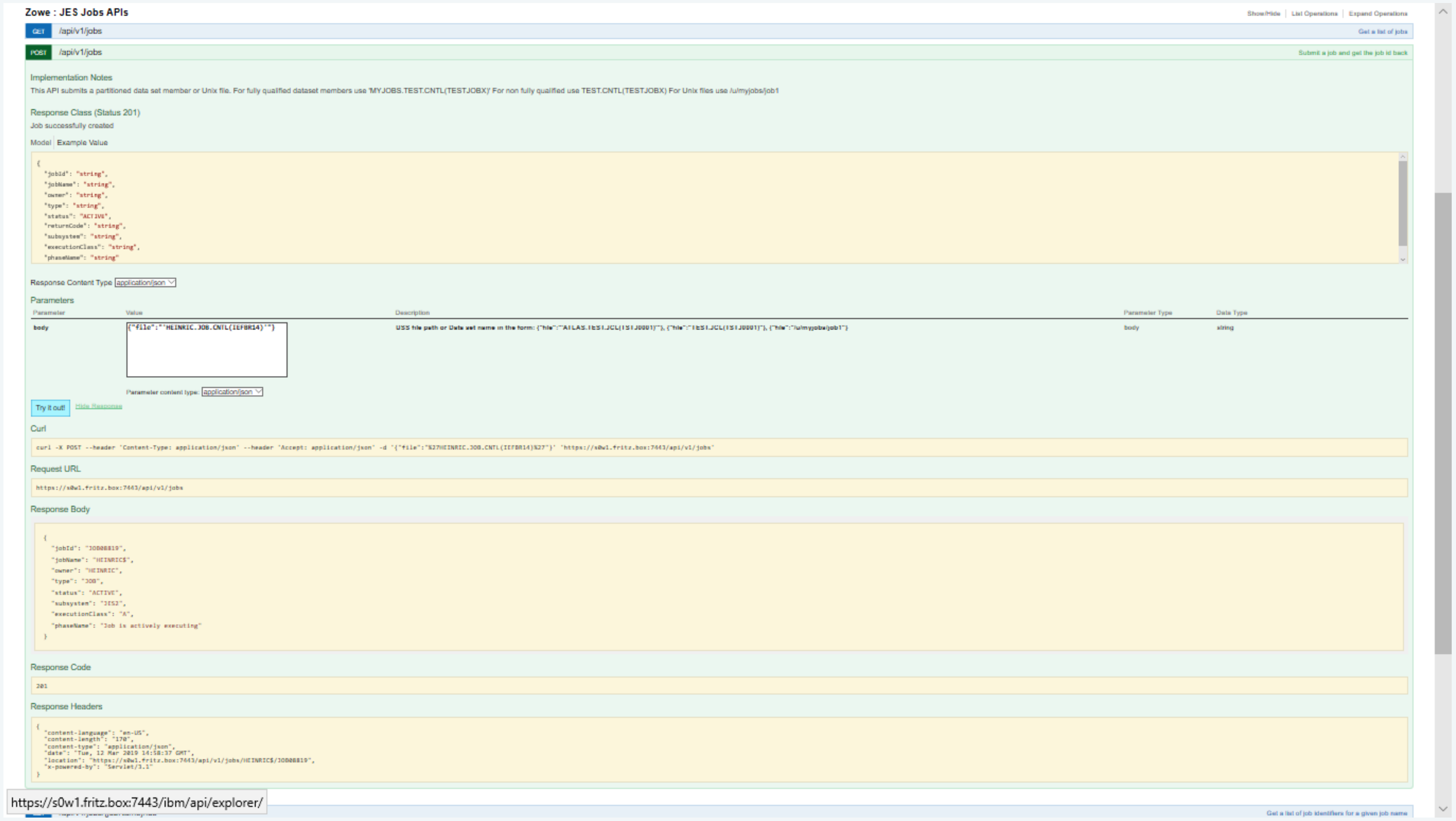
Response Code

```
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 | List Operations | Expand Operations

GET /api/v1/jobs — Get a list of jobs

POST /api/v1/jobs — Submit a job and get the job id 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": "ACTIVE",
  "returnCode": "string",
  "subsystem": "string",
  "executionClass": "string",
  "phaseName": "string"
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
body	{"file":"'HEINRIC.JOB.CNTL(IEFBR14)'"}	USS file path or Data set name in the form: {"file":"'ATLAS.TEST.JCL(TSTJ0001)'"}, {"file":"TEST.JCL(TSTJ0001)"}, {"file":"/u/myjobs/job1"}	body	string

Parameter content type: application/json

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{"file":"%27HEINRIC.JOB.CNTL(IEFBR14)%27"}' 'https://s0w1.fritz.box:7443/api/v1/jobs'
```

Request URL

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

Response Body

```
{
  "jobId": "JOB08819",
  "jobName": "HEINRIC$",
  "owner": "HEINRIC",
  "type": "JOB",
  "status": "ACTIVE",
  "subsystem": "JES2",
  "executionClass": "A",
  "phaseName": "Job is actively executing"
}
```

Response Code

```
201
```

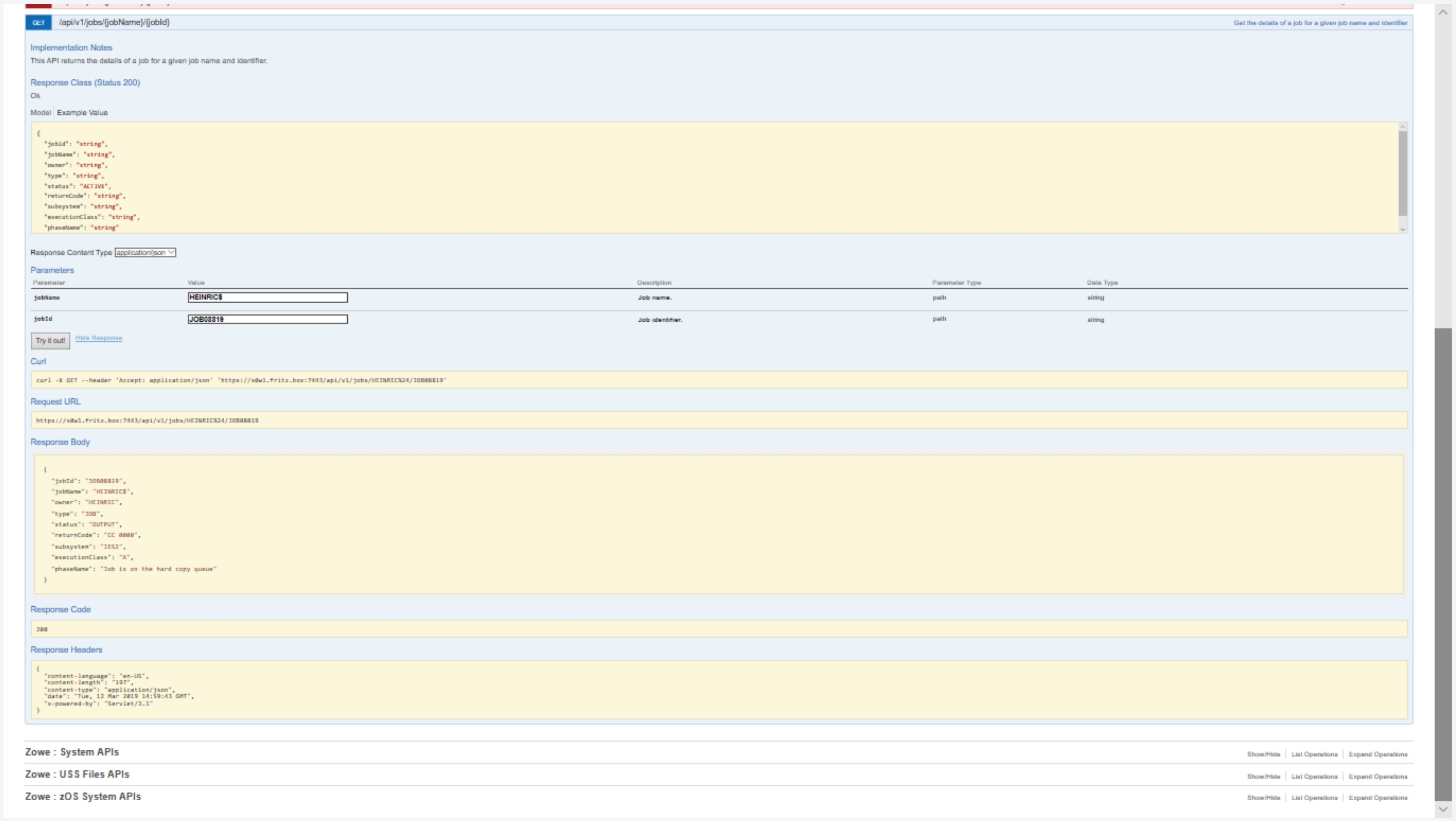
Response Headers

```
{
  "content-language": "en-US",
  "content-length": "170",
  "content-type": "application/json",
  "date": "Tue, 12 Mar 2019 14:58:37 GMT",
  "location": "https://s0w1.fritz.box:7443/api/v1/jobs/HEINRIC$/JOB08819",
  "x-powered-by": "Servlet/3.1"
}
```

https://s0w1.fritz.box:7443/ibm/api/explorer/

Get a list of job identifiers 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": "string",
  "jobName": "string",
  "owner": "string",
  "type": "string",
  "status": "ACTIVE",
  "returnCode": "string",
  "subsystem": "string",
  "executionClass": "string",
  "phaseName": "string"
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
jobName	HEINRIC$	Job name.	path	string
jobId	JOB08819	Job identifier.	path	string

Try it out! Hide Response

Curl

```
curl -X GET --header 'Accept: application/json' 'https://s0w1.fritz.box:7443/api/v1/jobs/HEINRIC%24/JOB08819'
```

Request URL

```
https://s0w1.fritz.box:7443/api/v1/jobs/HEINRIC%24/JOB08819
```

Response Body

```
{
  "jobId": "JOB08819",
  "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

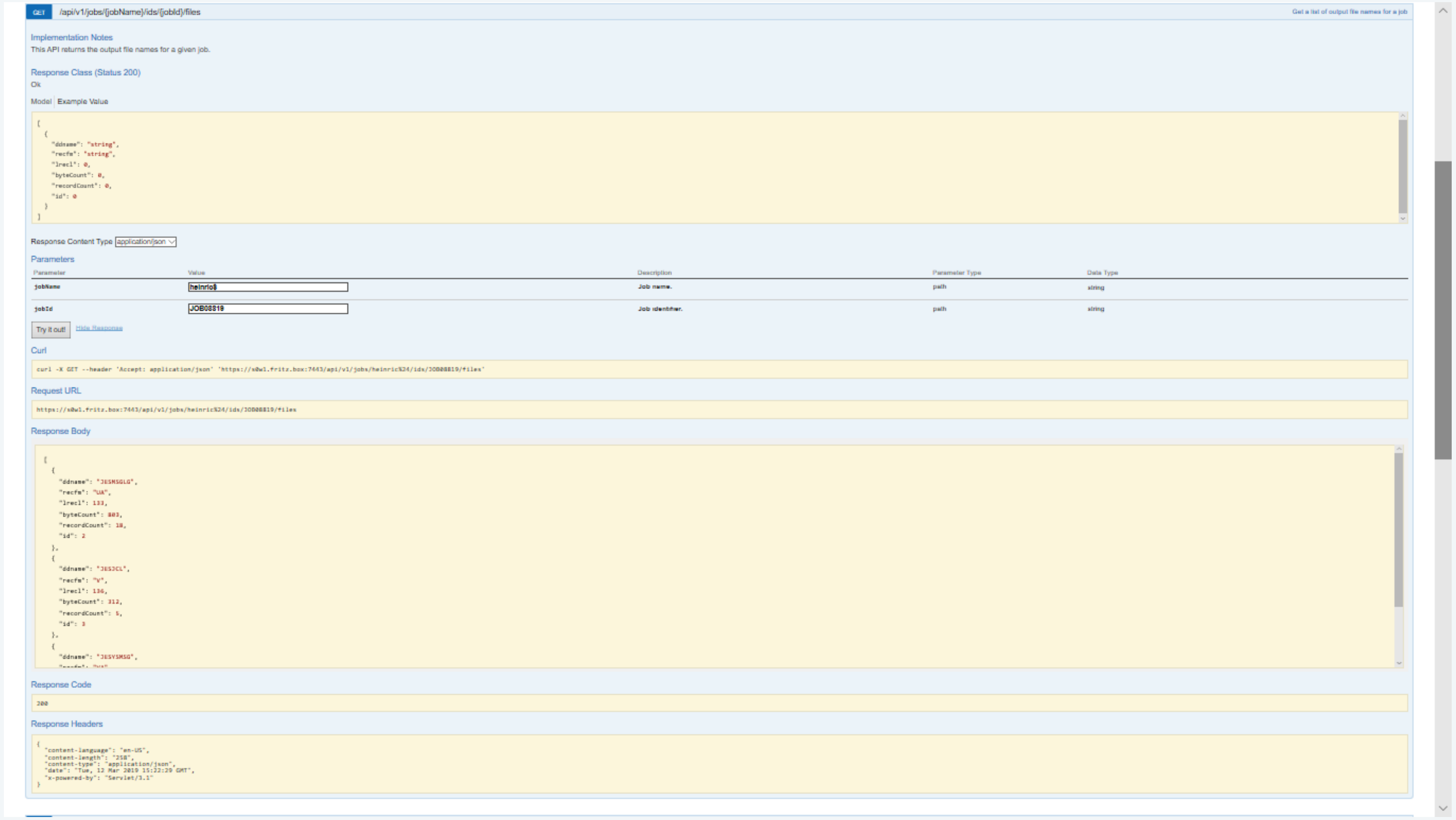
```
{
  "content-language": "en-US",
  "content-length": "197",
  "content-type": "application/json",
  "date": "Tue, 12 Mar 2019 14:59:43 GMT",
  "x-powered-by": "Servlet/3.1"
}
```

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



GET /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

```
[
  {
    "ddname": "string",
    "recfm": "string",
    "lrecl": 0,
    "byteCount": 0,
    "recordCount": 0,
    "id": 0
  }
]
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
jobName	heinric$	Job name.	path	string
jobId	JOB08819	Job identifier.	path	string

Try it out! Hide Response

Curl

```
curl -X GET --header 'Accept: application/json' 'https://s0w1.fritz.box:7443/api/v1/jobs/heinric%24/ids/JOB08819/files'
```

Request URL

```
https://s0w1.fritz.box:7443/api/v1/jobs/heinric%24/ids/JOB08819/files
```

Response Body

```
[
  {
    "ddname": "JESMSGLG",
    "recfm": "UA",
    "lrecl": 133,
    "byteCount": 803,
    "recordCount": 18,
    "id": 2
  },
  {
    "ddname": "JESJCL",
    "recfm": "V",
    "lrecl": 136,
    "byteCount": 312,
    "recordCount": 5,
    "id": 3
  },
  {
    "ddname": "JESYSMSG",
```

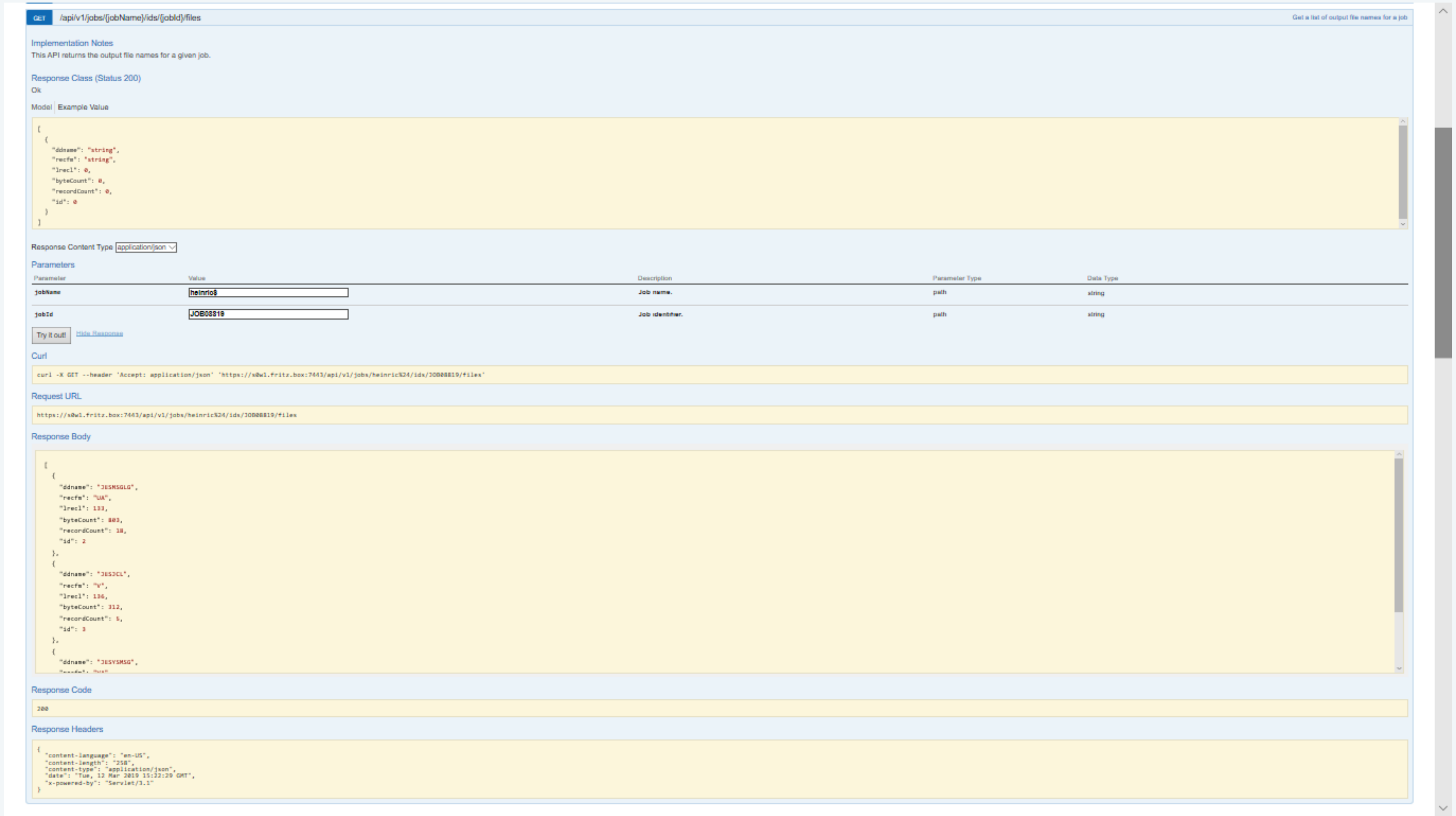
Response Code

```
200
```

Response Headers

```
{
  "content-language": "en-US",
  "content-length": "258",
  "content-type": "application/json",
  "date": "Tue, 12 Mar 2019 15:22:29 GMT",
  "x-powered-by": "Servlet/3.1"
}
```

Zowe examples – the API Catalog



GET /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

```
[
  {
    "ddname": "string",
    "recfm": "string",
    "lrecl": 0,
    "byteCount": 0,
    "recordCount": 0,
    "id": 0
  }
]
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
jobName	heinric$	Job name.	path	string
jobId	JOB08819	Job identifier.	path	string

Try it out! Hide Response

Curl

```
curl -X GET --header 'Accept: application/json' 'https://s0w1.fritz.box:7443/api/v1/jobs/heinric%24/ids/JOB08819/files'
```

Request URL

```
https://s0w1.fritz.box:7443/api/v1/jobs/heinric%24/ids/JOB08819/files
```

Response Body

```
[
  {
    "ddname": "JESMSGLG",
    "recfm": "UA",
    "lrecl": 133,
    "byteCount": 803,
    "recordCount": 18,
    "id": 2
  },
  {
    "ddname": "JESJCL",
    "recfm": "V",
    "lrecl": 136,
    "byteCount": 312,
    "recordCount": 5,
    "id": 3
  },
  {
    "ddname": "JESYSMSG",
```

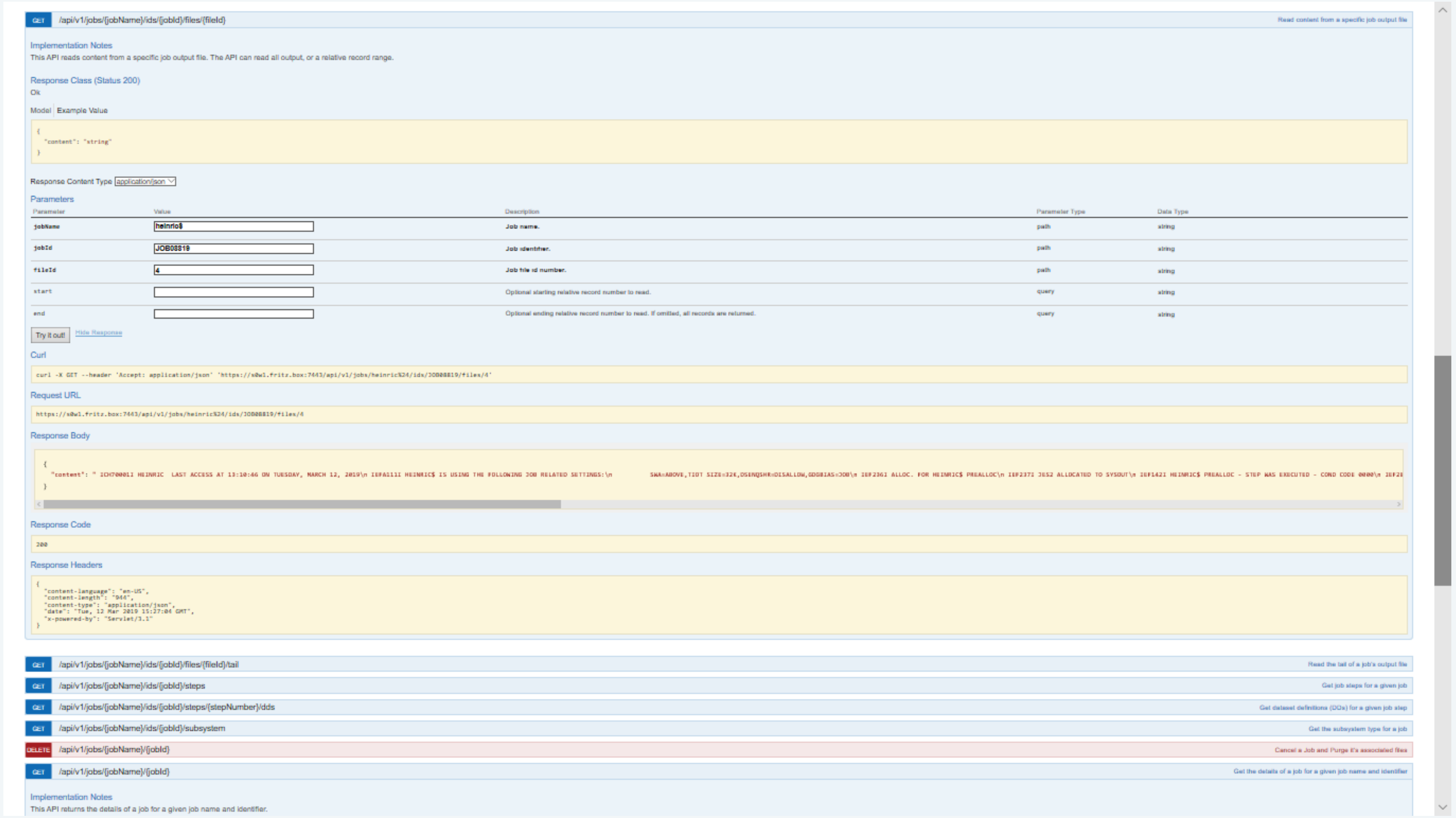
Response Code

```
200
```

Response Headers

```
{
  "content-language": "en-US",
  "content-length": "258",
  "content-type": "application/json",
  "date": "Tue, 12 Mar 2019 15:22:29 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

Parameters

Parameter	Value	Description	Parameter Type	Data Type
jobName	heinric$	Job name.	path	string
jobId	JOB08819	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://s0w1.fritz.box:7443/api/v1/jobs/heinric%24/ids/JOB08819/files/4'
```

Request URL

```
https://s0w1.fritz.box:7443/api/v1/jobs/heinric%24/ids/JOB08819/files/4
```

Response Body

```
{
  "content": " ICH70001I HEINRIC  LAST ACCESS AT 13:10:46 ON TUESDAY, MARCH 12, 2019\n IEFA111I HEINRIC$ IS USING THE FOLLOWING JOB RELATED SETTINGS:\n         SWA=ABOVE,TIOT SIZE=32K,DSENQSHR=DISALLOW,GDGBIAS=JOB\n IEF236I ALLOC. FOR HEINRIC$ PREALLOC\n IEF237I JES2 ALLOCATED TO SYSOUT\n IEF142I HEINRIC$ PREALLOC - STEP WAS EXECUTED - COND CODE 0000\n IEF2
}
```

Response Code

```
200
```

Response Headers

```
{
  "content-language": "en-US",
  "content-length": "944",
  "content-type": "application/json",
  "date": "Tue, 12 Mar 2019 15:27:04 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}/dds — Get dataset definitions (DDs) for a given job step

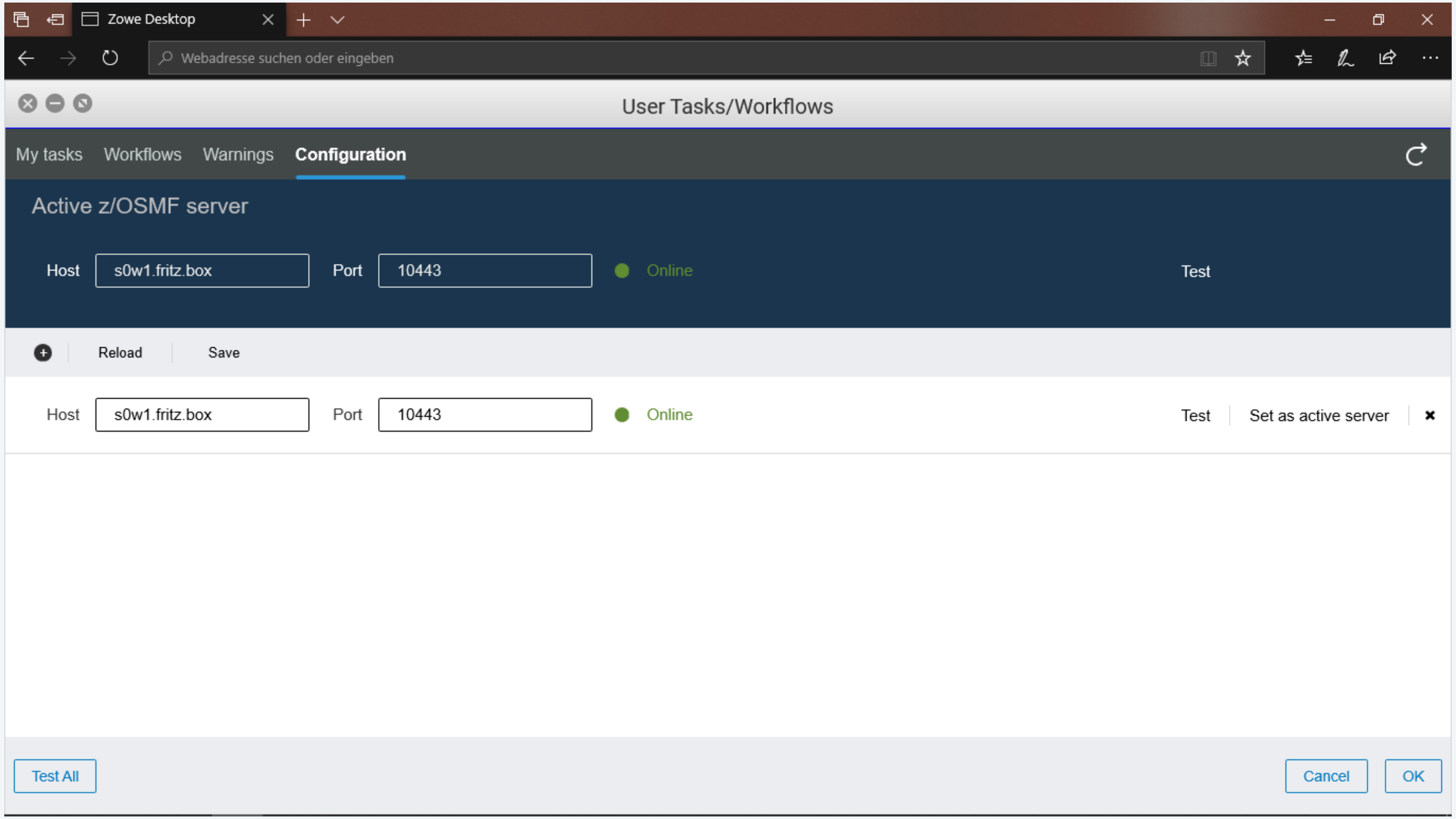
GET /api/v1/jobs/{jobName}/ids/{jobId}/subsystem — Get the subsystem type for a job

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

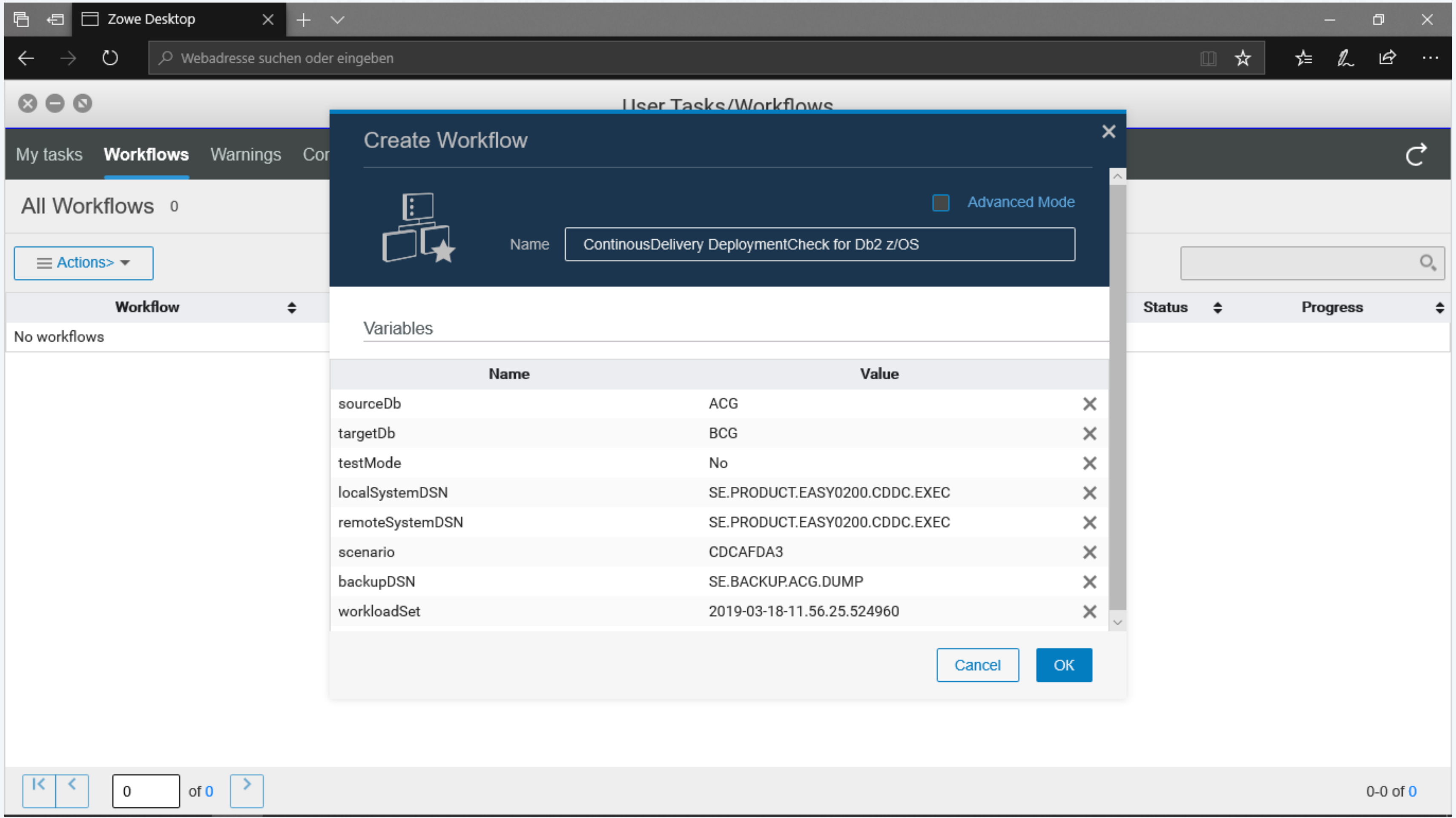


Active z/OSMF server

Host	s0w1.fritz.box	Port	10443	● Online

Host	s0w1.fritz.box	Port	10443	● Online

Zowe examples – User Tasks/workflows



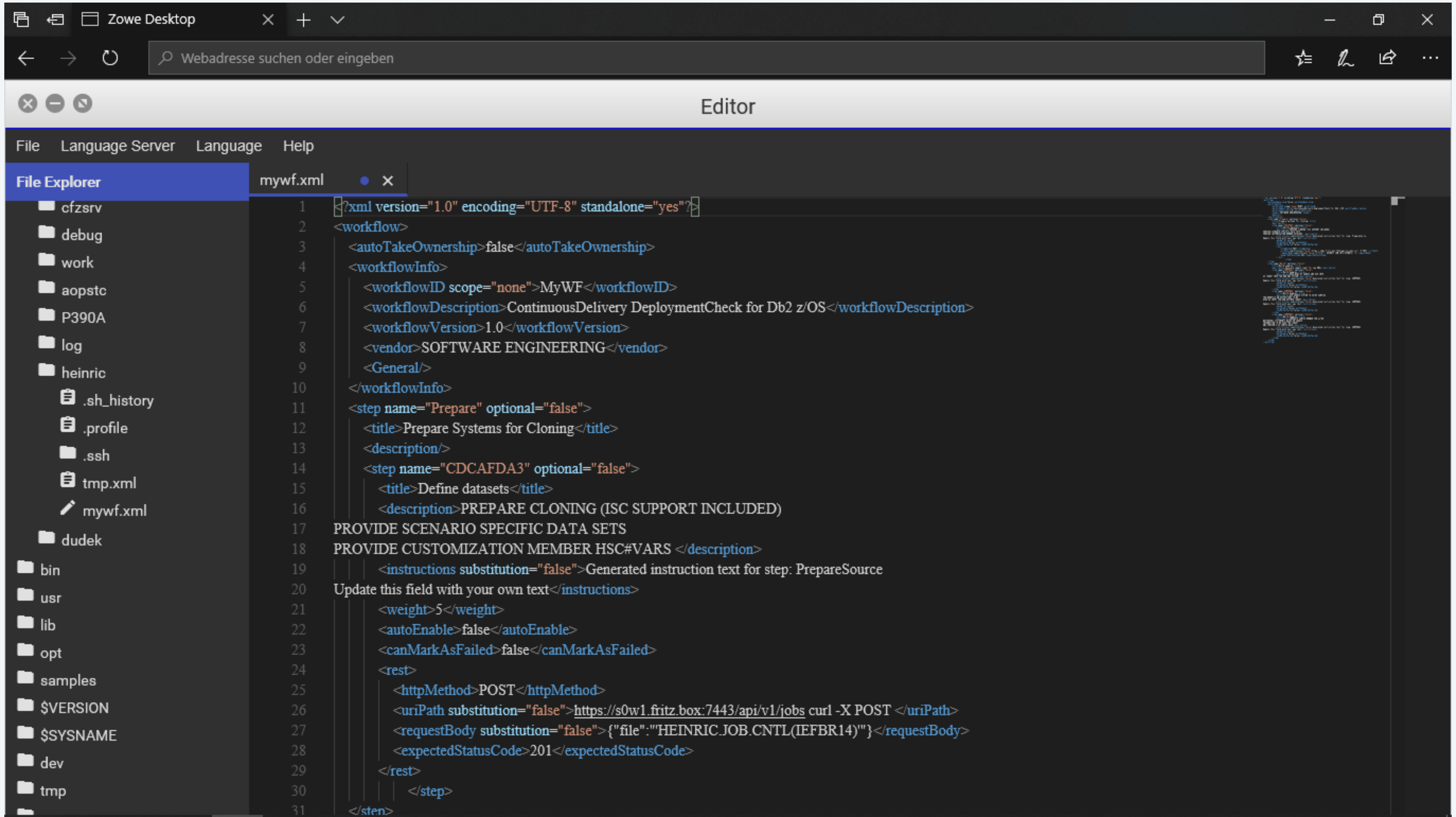
Name: ContinousDelivery DeploymentCheck for Db2 z/OS

Variables

Workflow	
No workflows	

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

Zowe examples – the Editor

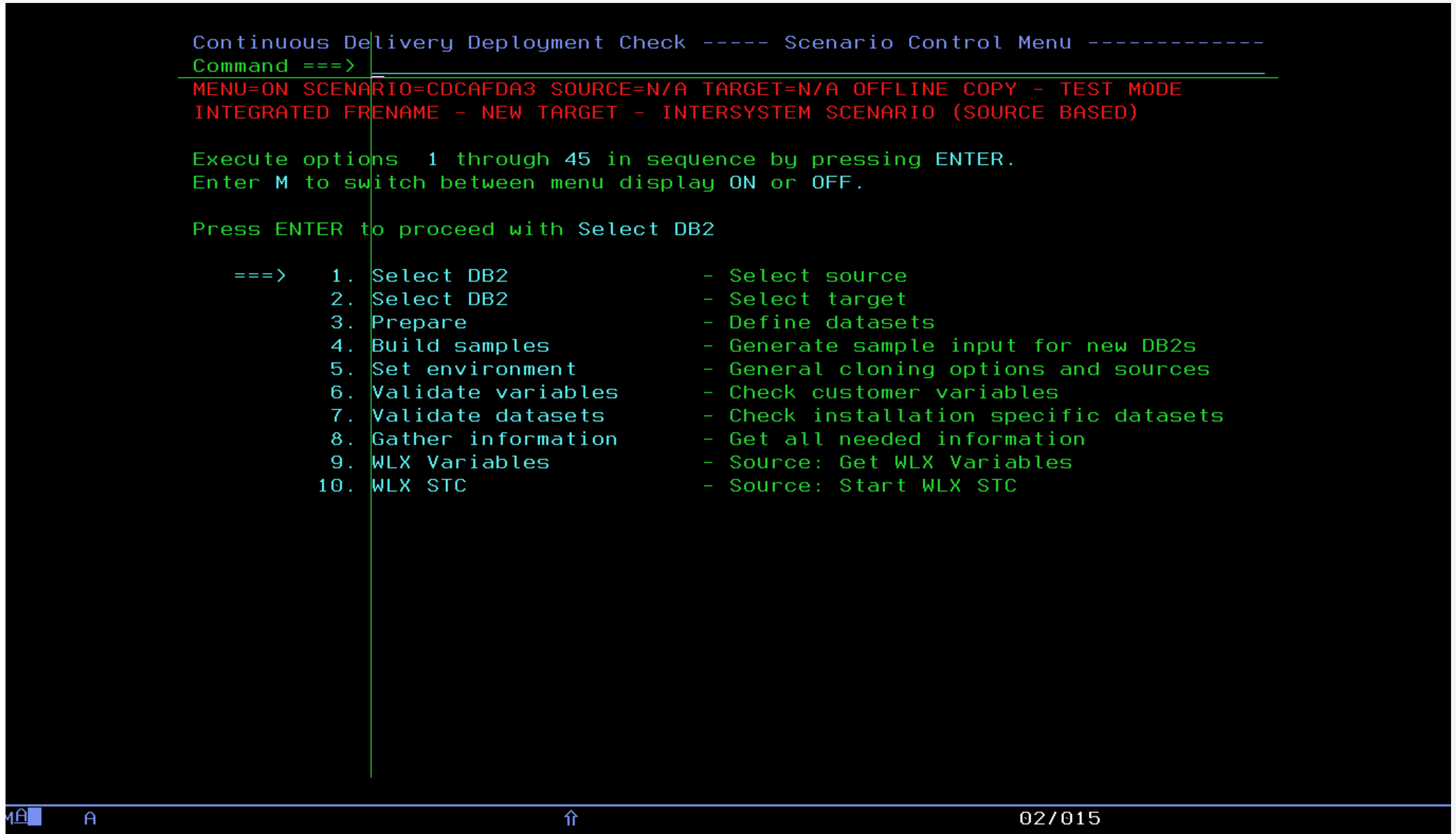


cfzsrv
debug
work
aopstc
P390A
log
heinric
.sh_history
.profile
.ssh
tmp.xml
mywf.xml
dudek
bin
usr
lib
opt
samples
$VERSION
$SYSNAME
dev
tmp

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<workflow>
  <autoTakeOwnership>false</autoTakeOwnership>
  <workflowInfo>
    <workflowID scope="none">MyWF</workflowID>
    <workflowDescription>ContinuousDelivery DeploymentCheck for Db2 z/OS</workflowDescription>
    <workflowVersion>1.0</workflowVersion>
    <vendor>SOFTWARE ENGINEERING</vendor>
    <General/>
  </workflowInfo>
  <step name="Prepare" optional="false">
    <title>Prepare Systems for Cloning</title>
    <description/>
    <step name="CDCAFDA3" optional="false">
      <title>Define datasets</title>
      <description>PREPARE CLONING (ISC SUPPORT INCLUDED)
PROVIDE SCENARIO SPECIFIC DATA SETS
PROVIDE CUSTOMIZATION MEMBER HSC#VARS </description>
      <instructions substitution="false">Generated instruction text for step: PrepareSource
Update this field with your own text</instructions>
      <weight>5</weight>
      <autoEnable>false</autoEnable>
      <canMarkAsFailed>false</canMarkAsFailed>
      <rest>
        <httpMethod>POST</httpMethod>
        <uriPath substitution="false">https://s0w1.fritz.box:7443/api/v1/jobs curl -X POST </uriPath>
        <requestBody substitution="false">{"file":"HEINRIC.JOB.CNTL(IEFBR14)"}</requestBody>
        <expectedStatusCode>201</expectedStatusCode>
      </rest>
        </step>
  </step>
```

Hands-on usage based on a cloning example

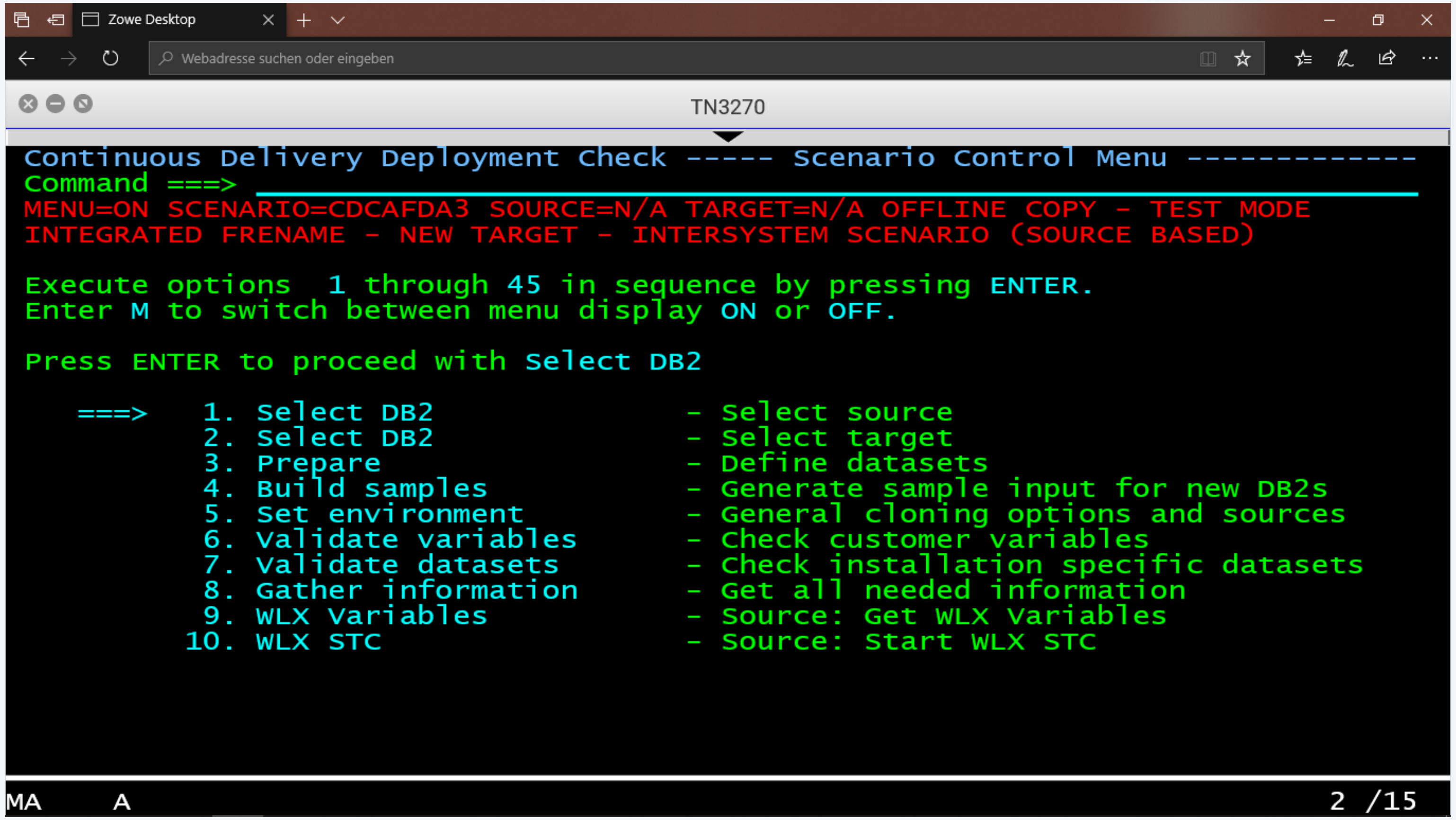
Goal: Run a batch job based Db2 system cloning process out of Zowe



```
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









MA      A                                    02/015
```

Hands-on usage based on a cloning example

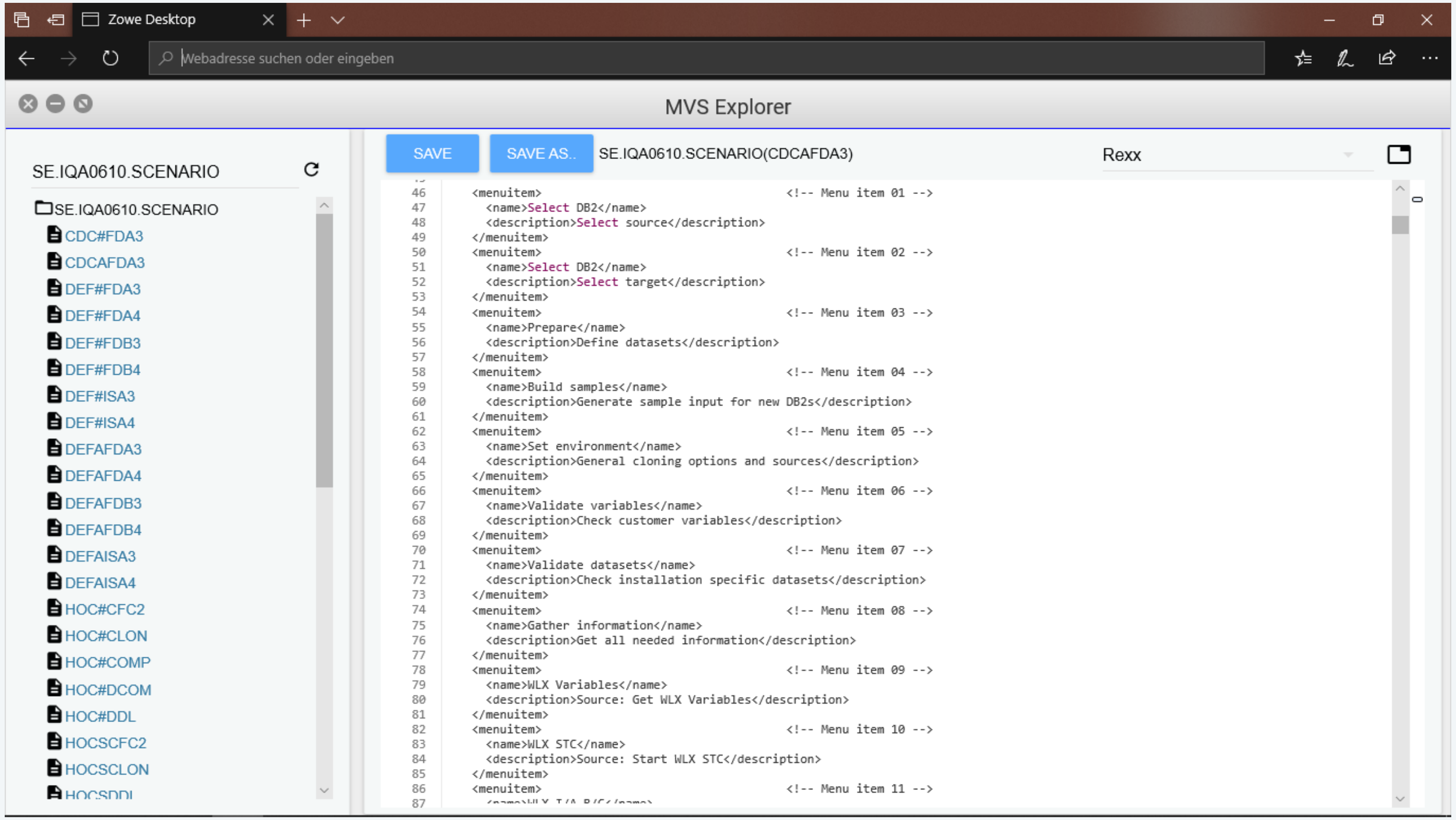
Goal: Run a batch job based Db2 system cloning process out of Zowe



```
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




MA      A                                                                2 /15
```

Hands-on usage based on a cloning example

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



SE.IQA0610.SCENARIO(CDCAFDA3)

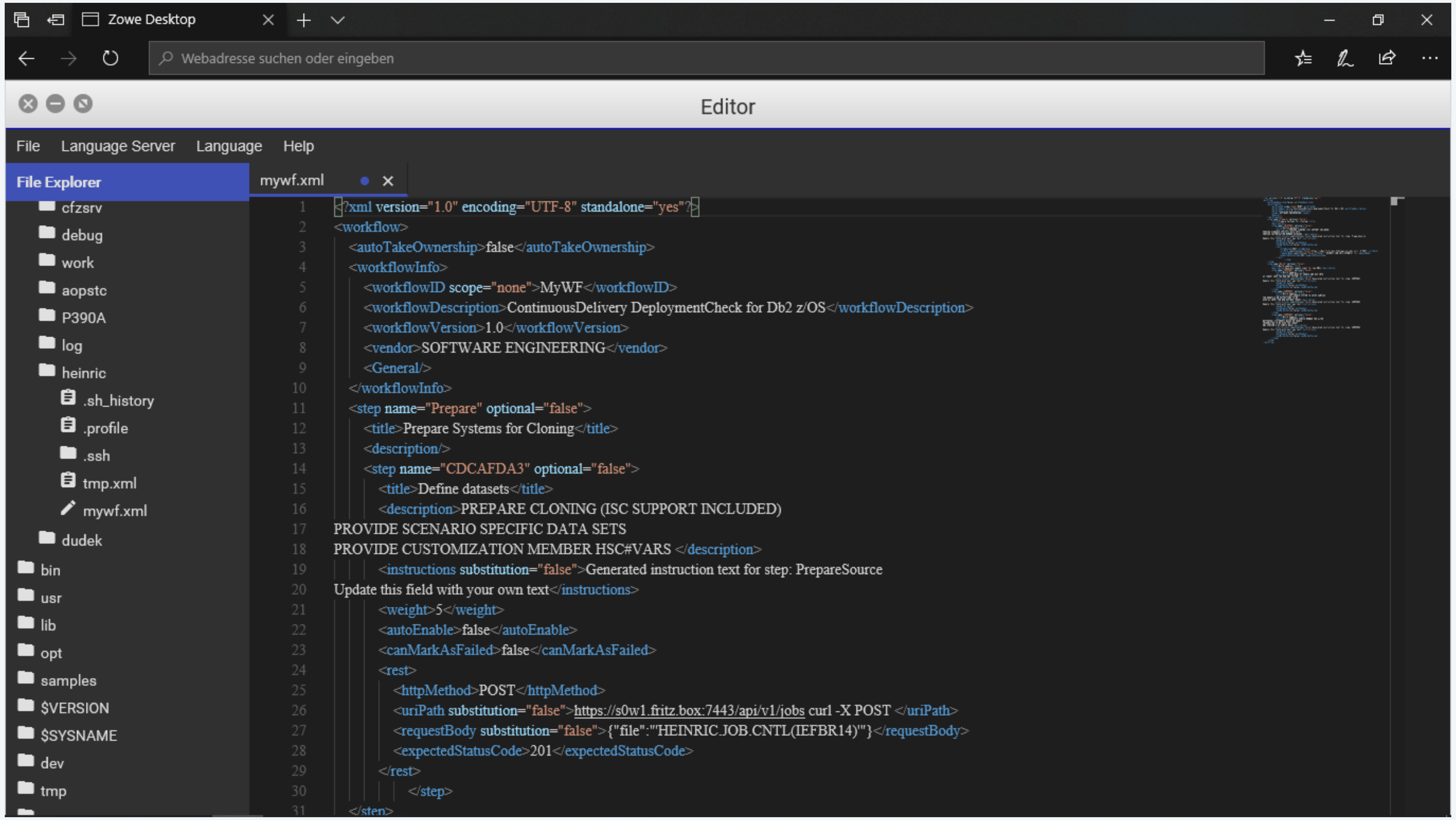
SE.IQA0610.SCENARIO

- SE.IQA0610.SCENARIO
  - CDC#FDA3
  - CDCAFDA3
  - DEF#FDA3
  - DEF#FDA4
  - DEF#FDB3
  - DEF#FDB4
  - DEF#ISA3
  - DEF#ISA4
  - DEFAFDA3
  - DEFAFDA4
  - DEFAFDB3
  - DEFAFDB4
  - DEFAISA3
  - DEFAISA4
  - HOC#CFC2
  - HOC#CLON
  - HOC#COMP
  - HOC#DCOM
  - HOC#DDL
  - HOCSCFC2
  - HOCSCLON
  - HOCSDD

```
<menuitem>                                             <!-- Menu item 01 -->
  <name>Select DB2</name>
  <description>Select source</description>
</menuitem>
<menuitem>                                             <!-- Menu item 02 -->
  <name>Select DB2</name>
  <description>Select target</description>
</menuitem>
<menuitem>                                             <!-- Menu item 03 -->
  <name>Prepare</name>
  <description>Define datasets</description>
</menuitem>
<menuitem>                                             <!-- Menu item 04 -->
  <name>Build samples</name>
  <description>Generate sample input for new DB2s</description>
</menuitem>
<menuitem>                                             <!-- Menu item 05 -->
  <name>Set environment</name>
  <description>General cloning options and sources</description>
</menuitem>
<menuitem>                                             <!-- Menu item 06 -->
  <name>Validate variables</name>
  <description>Check customer variables</description>
</menuitem>
<menuitem>                                             <!-- Menu item 07 -->
  <name>Validate datasets</name>
  <description>Check installation specific datasets</description>
</menuitem>
<menuitem>                                             <!-- Menu item 08 -->
  <name>Gather information</name>
  <description>Get all needed information</description>
</menuitem>
<menuitem>                                             <!-- Menu item 09 -->
  <name>WLX Variables</name>
  <description>Source: Get WLX Variables</description>
</menuitem>
<menuitem>                                             <!-- Menu item 10 -->
  <name>WLX STC</name>
  <description>Source: Start WLX STC</description>
</menuitem>
<menuitem>                                             <!-- Menu item 11 -->
```

Hands-on usage based on a cloning example

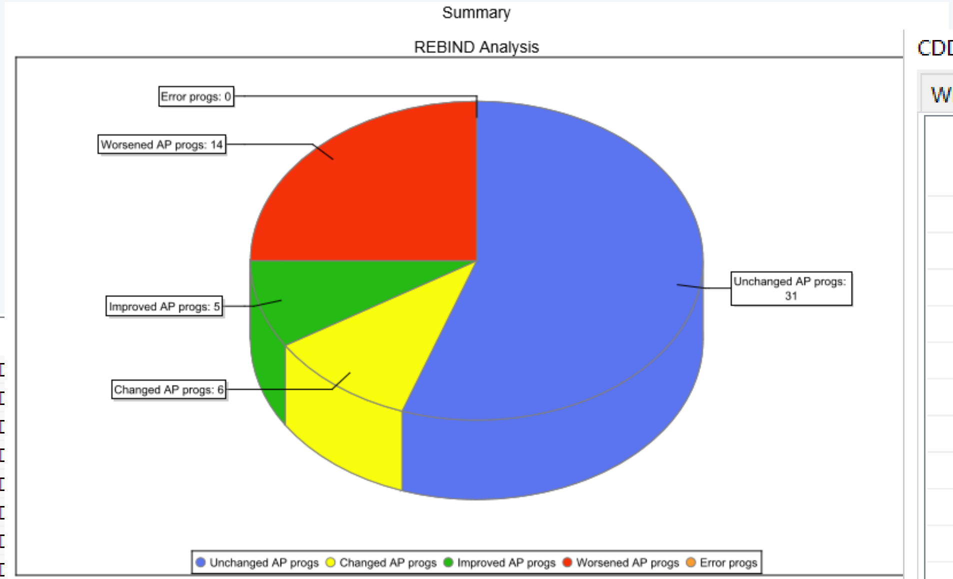
The flow of batch jobs is migrated to a workflow:

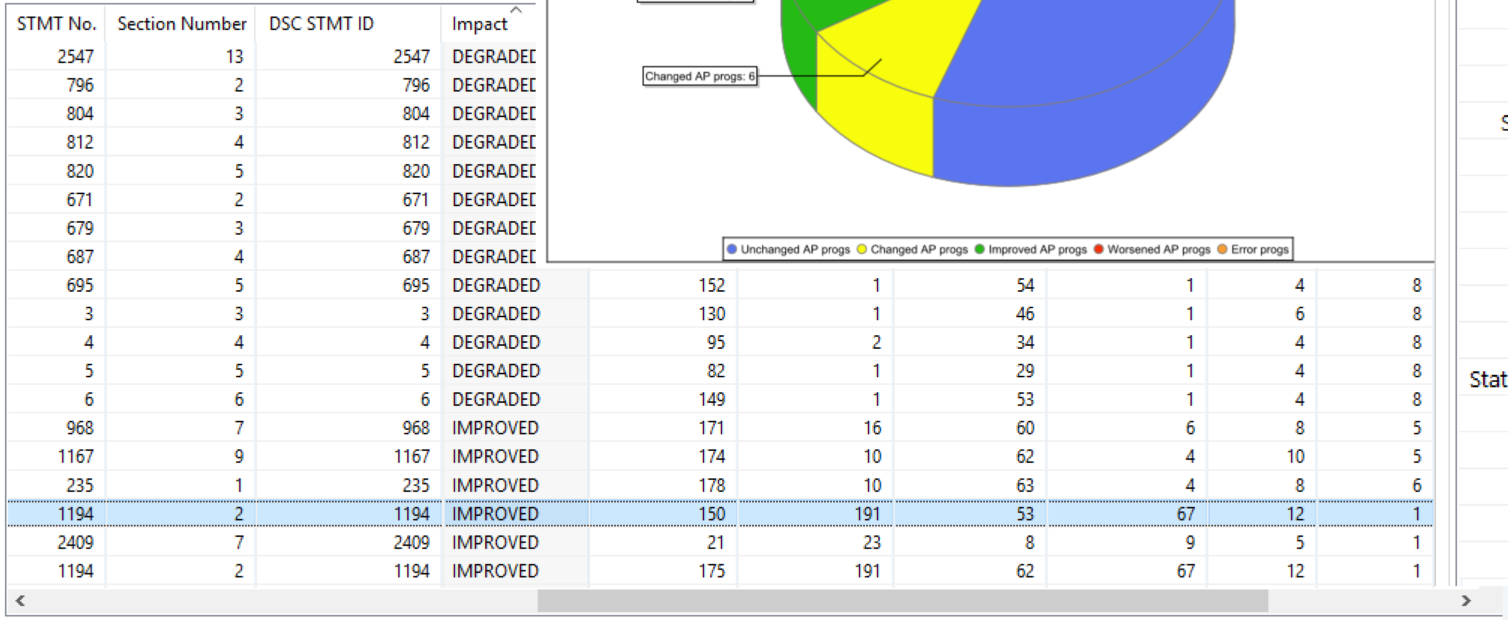


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<workflow>
  <autoTakeOwnership>false</autoTakeOwnership>
  <workflowInfo>
    <workflowID scope="none">MyWF</workflowID>
    <workflowDescription>ContinuousDelivery DeploymentCheck for Db2 z/OS</workflowDescription>
    <workflowVersion>1.0</workflowVersion>
    <vendor>SOFTWARE ENGINEERING</vendor>
    <General/>
  </workflowInfo>
  <step name="Prepare" optional="false">
    <title>Prepare Systems for Cloning</title>
    <description/>
    <step name="CDCAFDA3" optional="false">
      <title>Define datasets</title>
      <description>PREPARE CLONING (ISC SUPPORT INCLUDED)
PROVIDE SCENARIO SPECIFIC DATA SETS
PROVIDE CUSTOMIZATION MEMBER HSC#VARS </description>
      <instructions substitution="false">Generated instruction text for step: PrepareSource
Update this field with your own text</instructions>
      <weight>5</weight>
      <autoEnable>false</autoEnable>
      <canMarkAsFailed>false</canMarkAsFailed>
      <rest>
        <httpMethod>POST</httpMethod>
        <uriPath substitution="false">https://s0w1.fritz.box:7443/api/v1/jobs curl -X POST </uriPath>
        <requestBody substitution="false">{"file":"HEINRIC.JOB.CNTL(IEFBR14)"}</requestBody>
        <expectedStatusCode>201</expectedStatusCode>
      </rest>
        </step>
  </step>
```

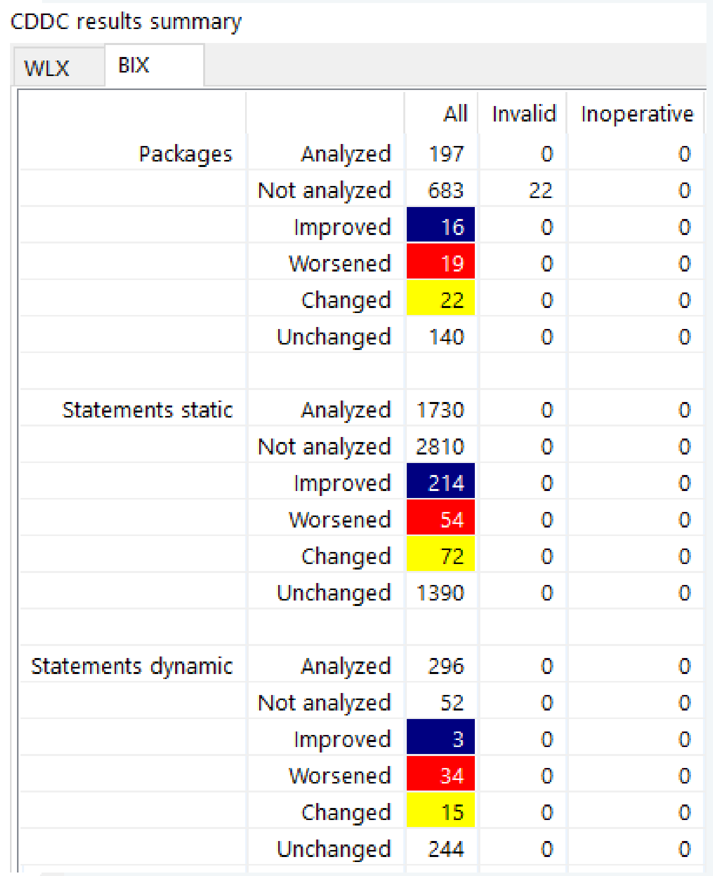
Hands-on usage based on a cloning example

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





STMT No.	Section Number	DSC STMT ID	Impact						
2547	13	2547	DEGRADE						
796	2	796	DEGRADE						
804	3	804	DEGRADE						
812	4	812	DEGRADE						
820	5	820	DEGRADE						
671	2	671	DEGRADE						
679	3	679	DEGRADE						
687	4	687	DEGRADE						
695	5	695	DEGRADED	152	1	54	1	4	8
3	3	3	DEGRADED	130	1	46	1	6	8
4	4	4	DEGRADED	95	2	34	1	4	8
5	5	5	DEGRADED	82	1	29	1	4	8
6	6	6	DEGRADED	149	1	53	1	4	8
968	7	968	IMPROVED	171	16	60	6	8	5
1167	9	1167	IMPROVED	174	10	62	4	10	5
235	1	235	IMPROVED	178	10	63	4	8	6
1194	2	1194	IMPROVED	150	191	53	67	12	1
2409	7	2409	IMPROVED	21	23	8	9	5	1
1194	2	1194	IMPROVED	175	191	62	67	12	1



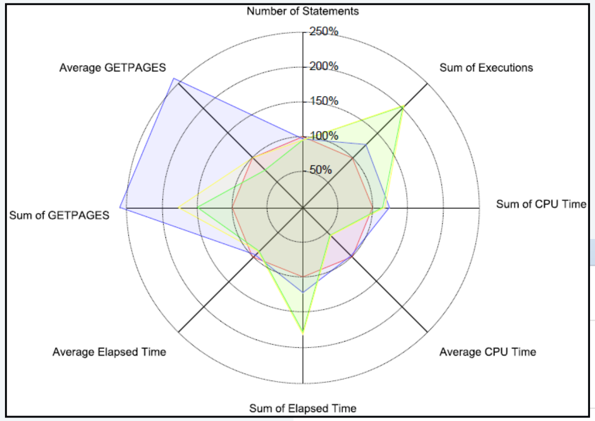
CDDC results summary

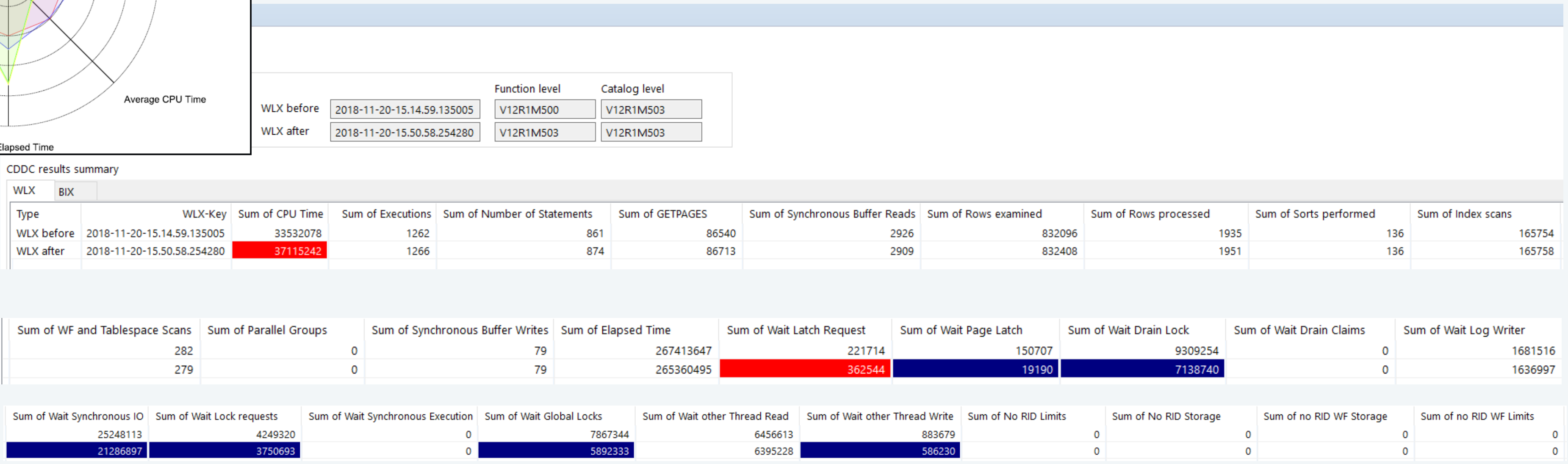
WLX | BIX

		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
WLX before	2018-11-20-15.14.59.135005	V12R1M500	V12R1M503
WLX after	2018-11-20-15.50.58.254280	V12R1M503	V12R1M503

CDDC results summary

WLX | BIX

Type	WLX-Key	Sum of CPU Time	Sum of Executions	Sum 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
WLX before	2018-11-20-15.14.59.135005	33532078	1262	861	86540	2926	832096	1935	136	165754
WLX after	2018-11-20-15.50.58.254280	37115242	1266	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
- …

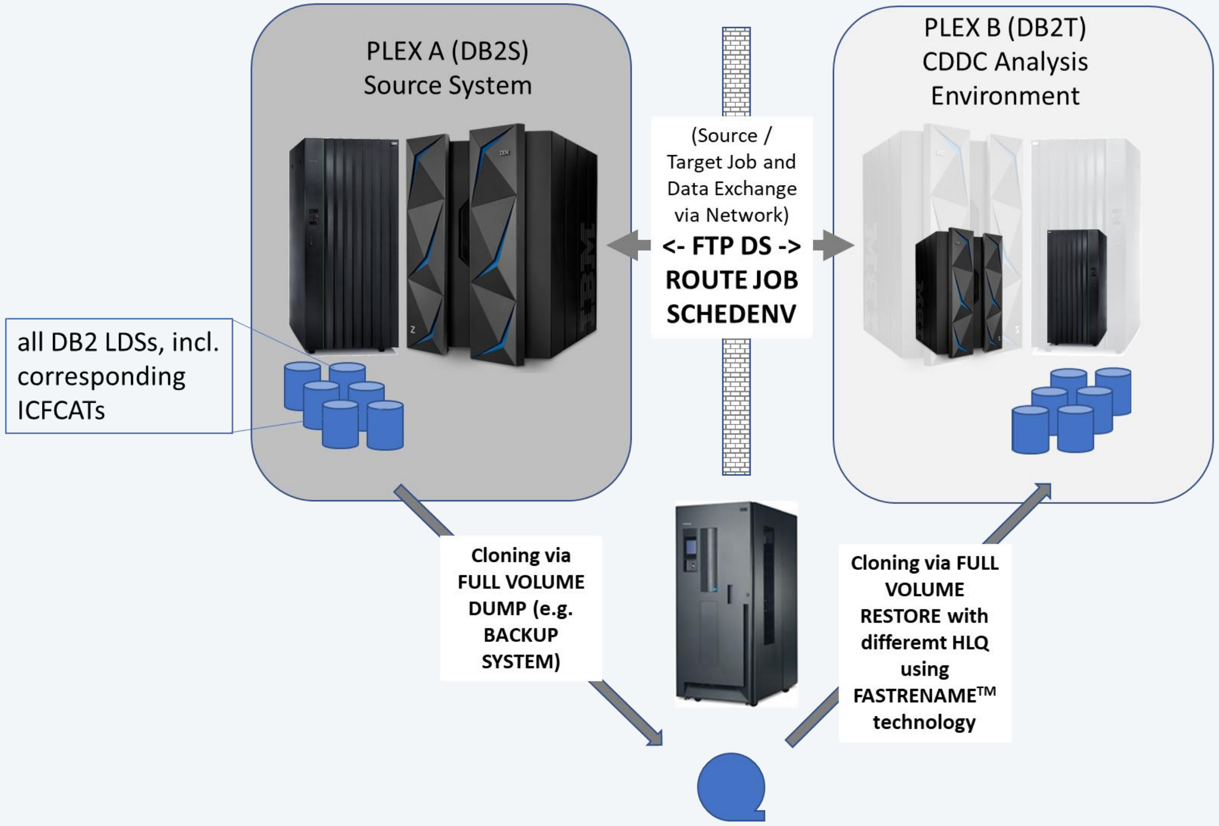
and any information can be accessed:

- 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


PLEX A (DB2S)
Source System
all DB2 LDSs, incl. corresponding ICFCATs
(Source / Target Job and Data Exchange via Network)
<- FTP DS ->
ROUTE JOB
SCHEDENV
PLEX B (DB2T)
CDDC Analysis Environment
Cloning via FULL VOLUME DUMP (e.g. BACKUP SYSTEM)
Cloning via FULL VOLUME RESTORE with differemt HLQ using FASTRENAME™ technology

Source System Hardware Simulation covering

- CPU
- BP
- RID/SORT Pool

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 …

Summary of experience

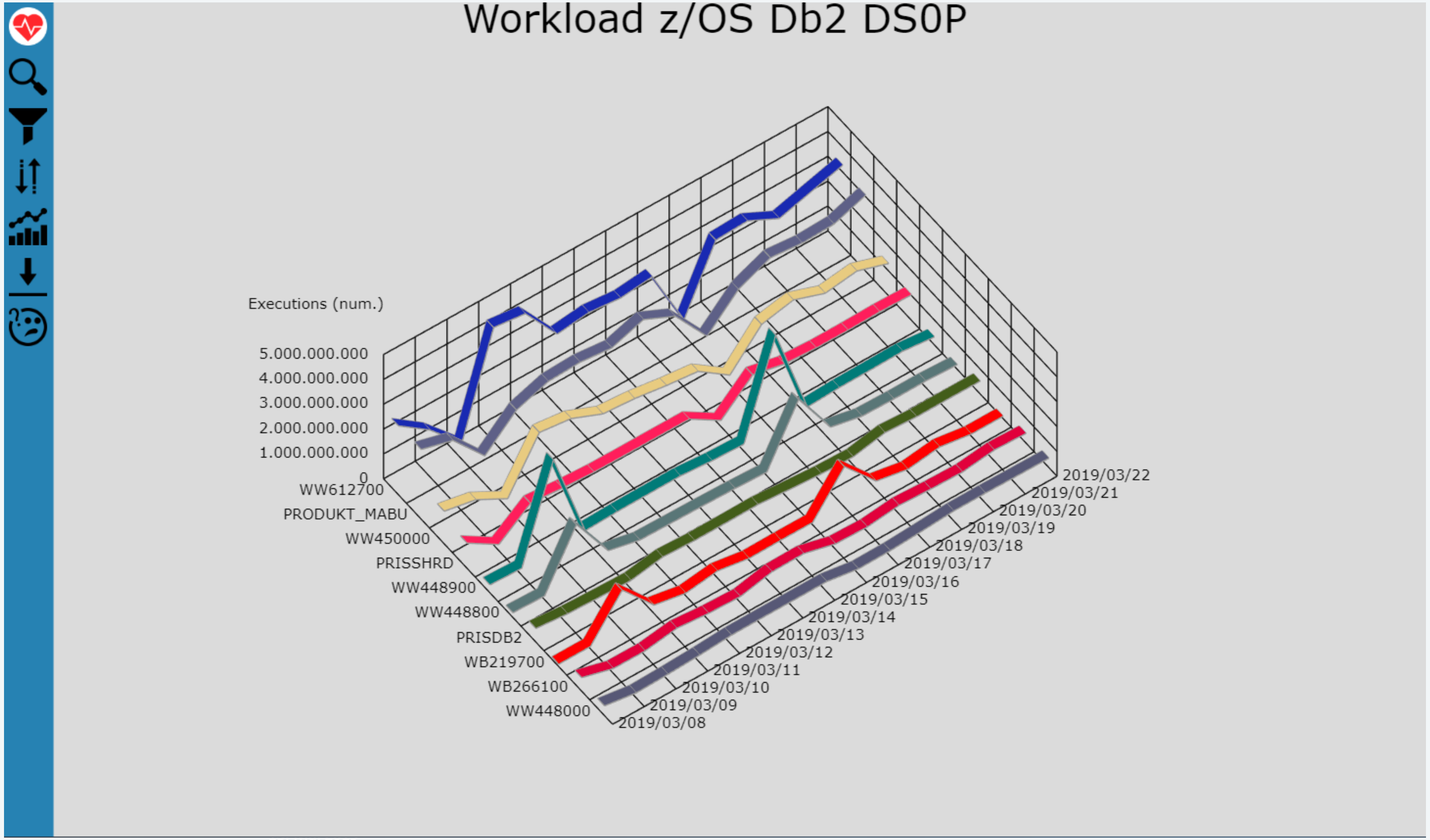
It’s worth it!!!

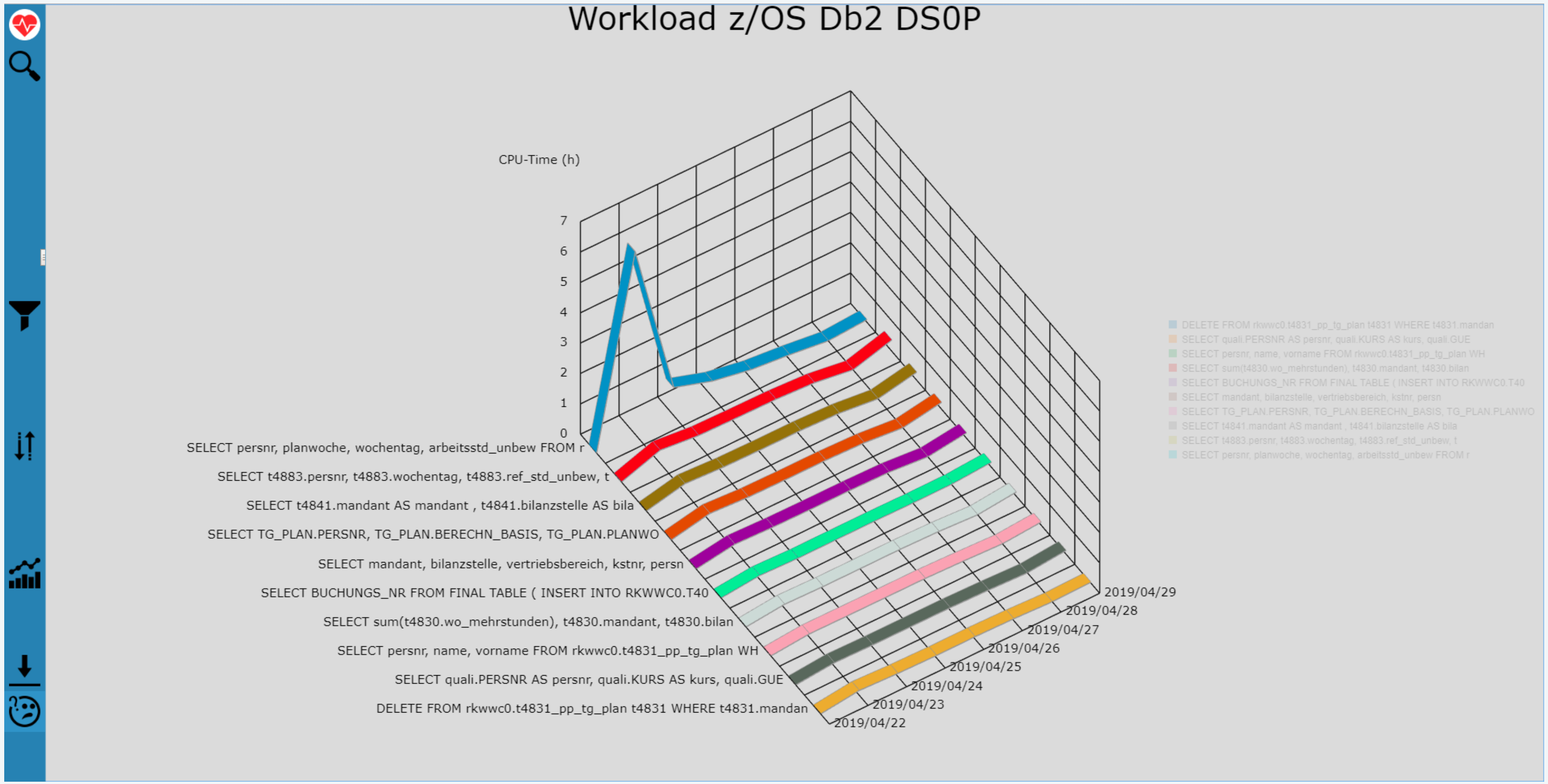
- We started with quite early (< 1.0) versions, but 1.2.0 was released on the 3rd of May

  → 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 our existing and upcoming tools and to contribute to the new ecosystem.

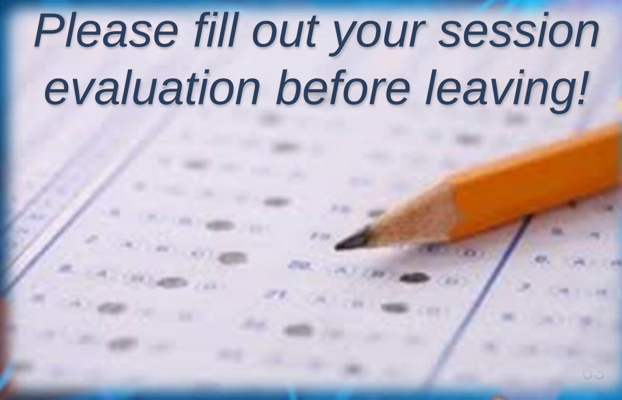

Workload z/OS Db2 DS0P
Executions (num.)
5.000.000.000
4.000.000.000
3.000.000.000
2.000.000.000
1.000.000.000
0
WW612700
PRODUKT_MABU
WW450000
PRISSHRD
WW448900
WW448800
PRISDB2
WB219700
WB266100
WW448000
2019/03/08
2019/03/09
2019/03/10
2019/03/11
2019/03/12
2019/03/13
2019/03/14
2019/03/15
2019/03/16
2019/03/17
2019/03/18
2019/03/19
2019/03/20
2019/03/21
2019/03/22


Workload z/OS Db2 DS0P
CPU-Time (h)
7
6
5
4
3
2
1
0
SELECT persnr, planwoche, wochentag, arbeitsstd_unbew FROM r
SELECT t4883.persnr, t4883.wochentag, t4883.ref_std_unbew, t
SELECT t4841.mandant AS mandant , t4841.bilanzstelle AS bila
SELECT TG_PLAN.PERSNR, TG_PLAN.BERECHN_BASIS, TG_PLAN.PLANWO
SELECT mandant, bilanzstelle, vertriebsbereich, kstnr, persn
SELECT BUCHUNGS_NR FROM FINAL TABLE (INSERT INTO RKWWC0.T40
SELECT sum(t4830.wo_mehrstunden), t4830.mandant, t4830.bilan
SELECT persnr, name, vorname FROM rkwwc0.t4831_pp_tg_plan WH
SELECT quali.PERSNR AS persnr, quali.KURS AS kurs, quali.GUE
DELETE FROM rkwwc0.t4831_pp_tg_plan t4831 WHERE t4831.mandan
2019/04/29
2019/04/28
2019/04/27
2019/04/26
2019/04/25
2019/04/24
2019/04/23
2019/04/22
DELETE FROM rkwwc0.t4831_pp_tg_plan t4831 WHERE t4831.mandan
SELECT quali.PERSNR AS persnr, quali.KURS AS kurs, quali.GUE
SELECT persnr, name, vorname FROM rkwwc0.t4831_pp_tg_plan WH
SELECT sum(t4830.wo_mehrstunden), t4830.mandant, t4830.bilan
SELECT BUCHUNGS_NR FROM FINAL TABLE (INSERT INTO RKWWC0.T40
SELECT mandant, bilanzstelle, vertriebsbereich, kstnr, persn
SELECT TG_PLAN.PERSNR, TG_PLAN.BERECHN_BASIS, TG_PLAN.PLANWO
SELECT t4841.mandant AS mandant , t4841.bilanzstelle AS bila
SELECT t4883.persnr, t4883.wochentag, t4883.ref_std_unbew, t
SELECT persnr, planwoche, wochentag, arbeitsstd_unbew FROM r

Roy Boxwell
SEGUS & SOFTWARE ENGINEERING
r.boxwell@seg.de

Session code: **[V1]**

IDUG
Leading the Db2 User Community since 1988


Please fill out your session evaluation before leaving!