# Phoenix Software International

Modernizing How You Work with the Mainframe

December 18, 2019

PHOENIX Software International®
SOLUTIONS FOR MAINFRAME AND PC PLATFORMS

# Objective

- Discuss how Phoenix Software International is applying the use of present-day tools with the mainframe such as:

  - (E)JES, (E)JES Web and Eclipse – A modern, lightweight browser-based system management tool for users who prefer not to work in a 3270 and for developers who want to leverage the Eclipse IDE for development

  - Zowe – Open source framework for the mainframe that provides solutions that allow development and operations teams to securely manage, control, script, and develop on the mainframe like any other cloud platform

  - z/OSMF Workflow – automation of routine procedures and tasks through the use of a modern interface and Zorow – An open source community dedicated to contributing and collaborating on z/OSMF Workflows

# Who We Are

- Phoenix has been providing enterprise software solutions around the globe since 1979.

- A privately held corporation not required to provide shareholders with quarterly profits, Phoenix can focus on long-range, customer-oriented projects and goals.

- Phoenix prides itself in being nimble, able to quickly capitalize on new hardware and software technology with a focus on software modernization, and providing the best possible user experience.

# Who We Are

## The three pillars that guide our development philosophy

Leveraging the latest hardware/software technology

Providing the best possible user experience

Software modernization

## Why do we focus on leveraging the latest technology?

We care about performance.

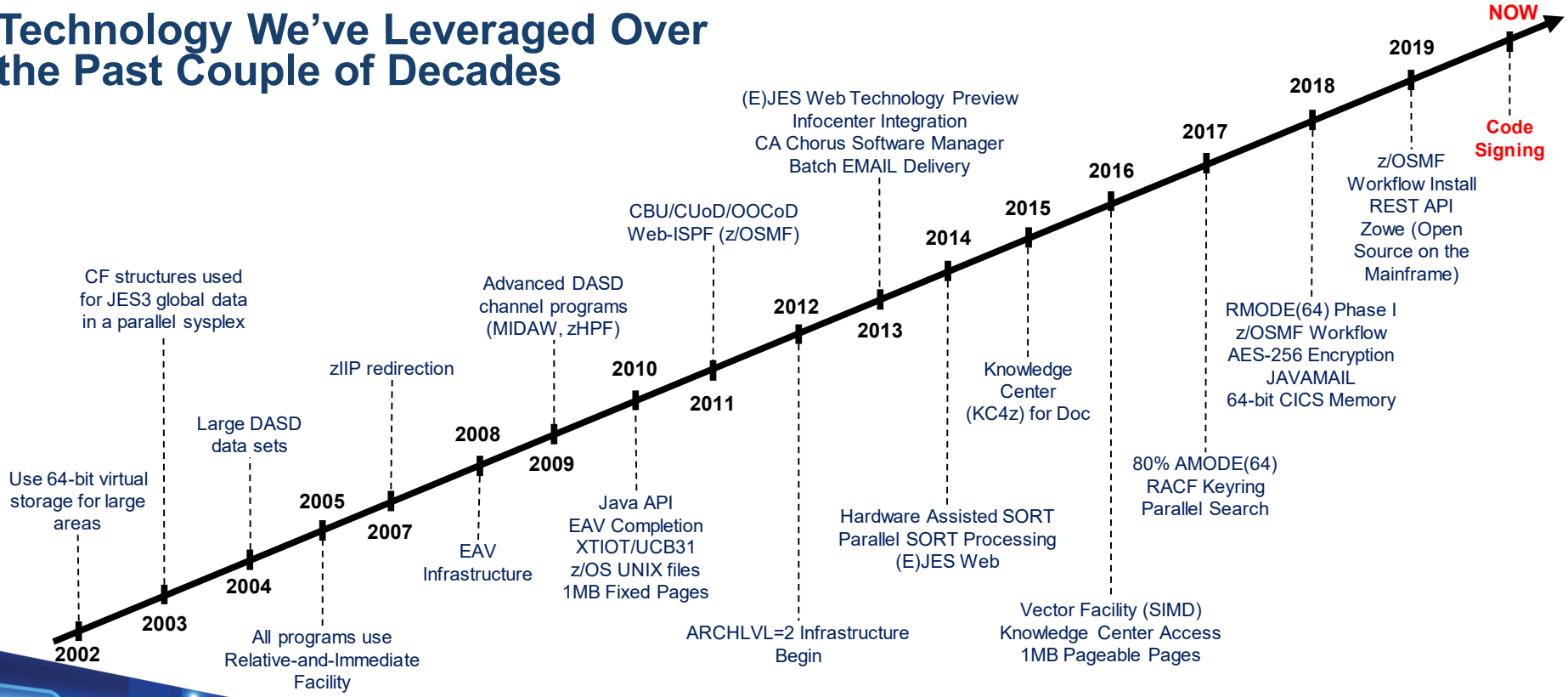
We care about reducing customer costs.

We care about the z/OS platform!

This unique philosophy has kept us in lock-step with IBM Z hardware and the z/OS operating system as they have evolved over time.

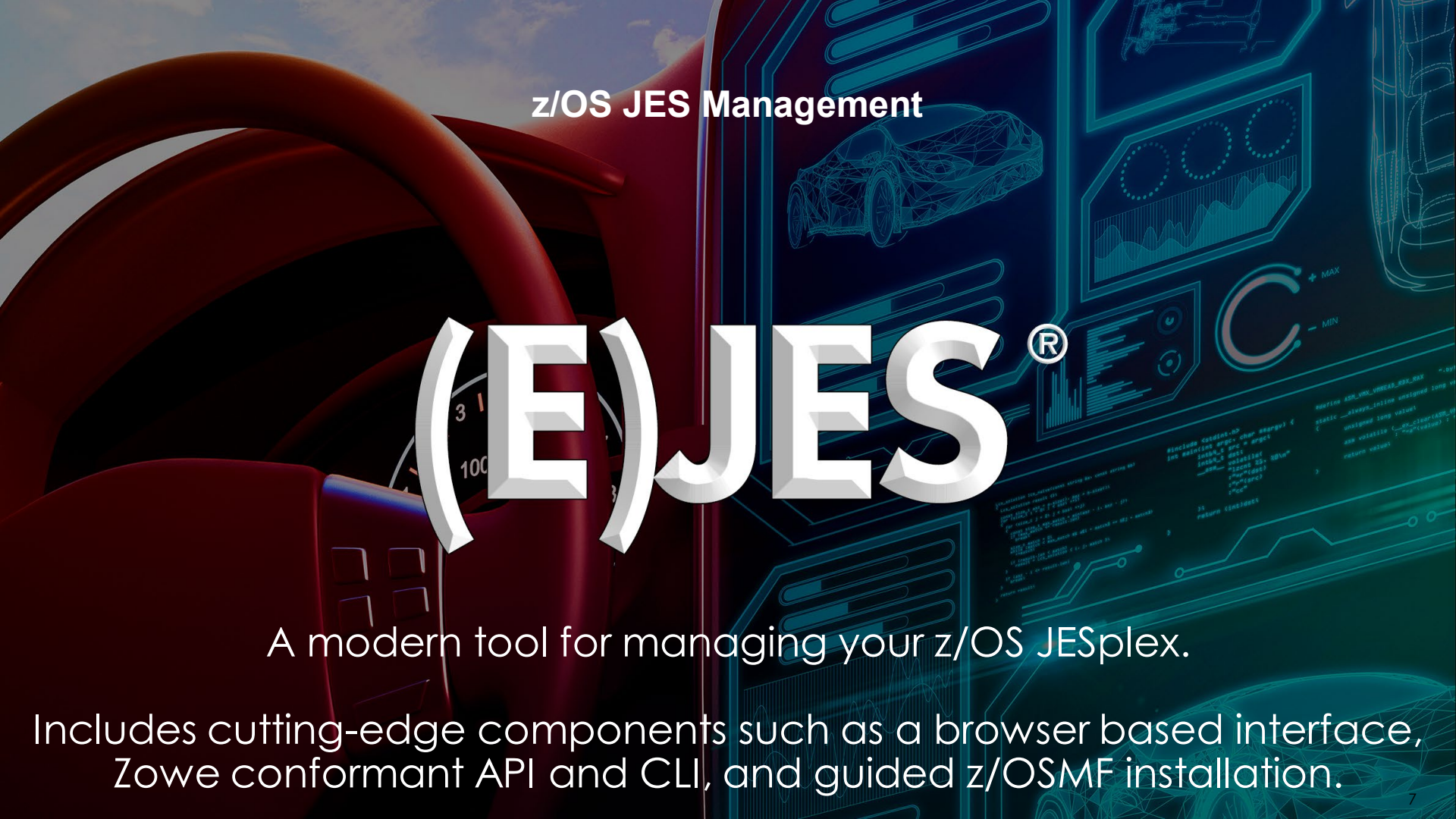
# Who We Are

## Technology We've Leveraged Over the Past Couple of Decades



**Phoenix Software JES3plus™ discussed in this presentation is a derivative work of IBM®'s JES3 licensed sourced code.**


## The Same, Only Better


z/OS JES Management
(E)JES®
A modern tool for managing your z/OS JESplex.
Includes cutting-edge components such as a browser based interface, Zowe conformant API and CLI, and guided z/OSMF installation.


z/OS Code Analysis
zHISR®
Generate reports that help tune applications by locating specific sections of your code that are the biggest CPU consumers.

# Objective

- Discuss how Phoenix Software International is applying the use of present-day tools with the mainframe such as:

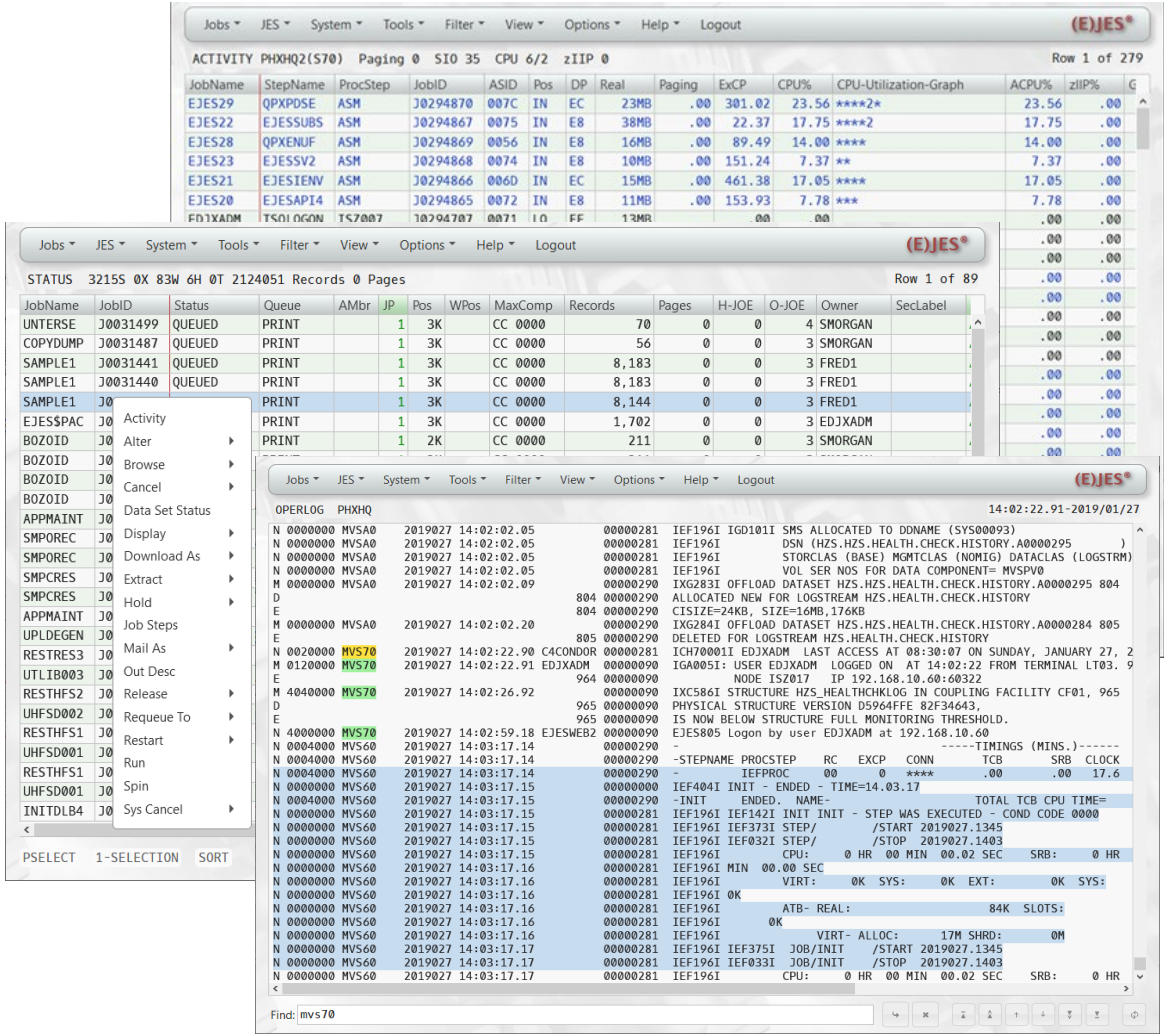
  - **(E)JES, (E)JES Web and Eclipse – A modern, lightweight browser-based system management tool for users who prefer not to work in a 3270 and for developers who want to leverage the Eclipse IDE for development**
  - Zowe – Open source framework for the mainframe that provides solutions that allow development and operations teams to securely manage, control, script, and develop on the mainframe like any other cloud platform
  - z/OSMF Workflow – automation of routine procedures and tasks through the use of a modern interface and Zorow – An open source community dedicated to contributing and collaborating on z/OSMF Workflows

# What is (E)JES?

- (E)JES is a system management tool that provides information to monitor, manage, and control a z/OS JESplex.
- (E)JES consists of a series of applications that provide immediate and current information about jobs, devices, queues, and other z/OS system resources.
- From these applications, you can observe the system's operation, browse data on SPOOL, and securely control its processing.
- Simple commands and rapid accessibility not only save you time, but also make interacting with z/OS and JES quick and easy.
- (E)JES can be used interactively via 3270 terminal, Web Browser, Zowe CLI, or programmatically using REST API and procedural APIs using popular mainframe languages.

# (E)JES Web

- Implemented in Java
- Nothing is stored on the user's PC
- No Java is run on the PC or in the browser
- Runs under IBM Open Liberty or Apache Tomcat
- Uses (E)JES Java API on the server side – therefore CPU cycles are redirected to zIIP
- Supports every (E)JES display – not just a convenient subset
- Performance is similar to 3270



# (E)JES Web

## Performance Objective – NO SACRIFICE!

Our goal is to replicate, as closely as possible, the performance experience currently enjoyed by (E)JES 3270 users, while adding the power and flexibility of a modern web interface.

## Average Refresh Times in our Environment

z13s 2965-D03 w/zIIP and OSA Express 5 connected (via GbE) to PCs with Intel Core i7 CPUs, running Mozilla Firefox under Microsoft 64-bit Windows 10 Professional.

| Average Timings | ACTIVITY 54 jobs | ACTIVITY 5 jobs | STATUS 54 jobs | STATUS 15 jobs |
|---|---|---|---|---|
| Server | .073 | .036 | .098 | .052 |
| Network | .005 | .007 | .006 | .004 |
| Total | .078 | .043 | .104 | .056 |

## 3270 Primary and Line Commands Supported

Most of the familiar mainframe primary and line commands are supported so experienced users feel comfortable.

## Familiar 3270 FIND Capabilities

ISPF-like find options such as prefix, suffix, word, chars, hex, etc. are all fully supported.

## Ultra-thin Client Model

(E)JES Web is an "ultra-thin" – fully virtualized – client with an intelligent server. Sorting, filtering, searching, etc. is performed on server – where the data resides. The client is merely a user interface for requesting and viewing the results of these data manipulations.

## Time-oriented Log Browsers

System log browsers are naturally oriented by date and time. The title line shows the date/time, associated with the top line, rather than line/record number.

## Metafilters Management

A virtually unlimited number of complex metafilter sets can be created, named and saved so you can apply them as needed simply by clicking on the name of the set.

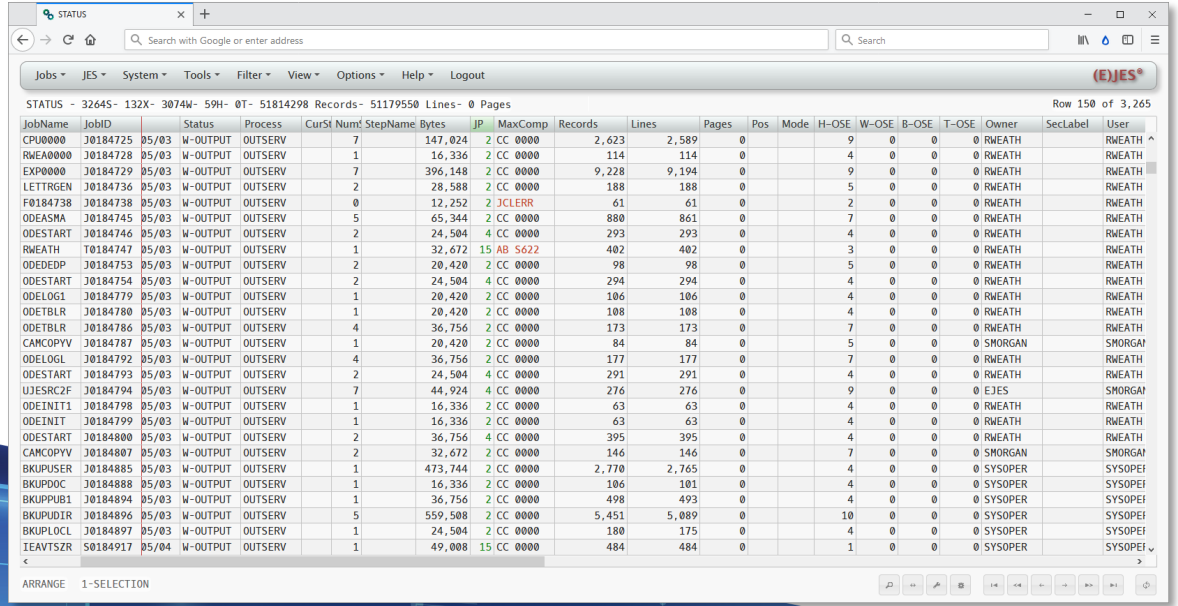
## Download as Text or PDF

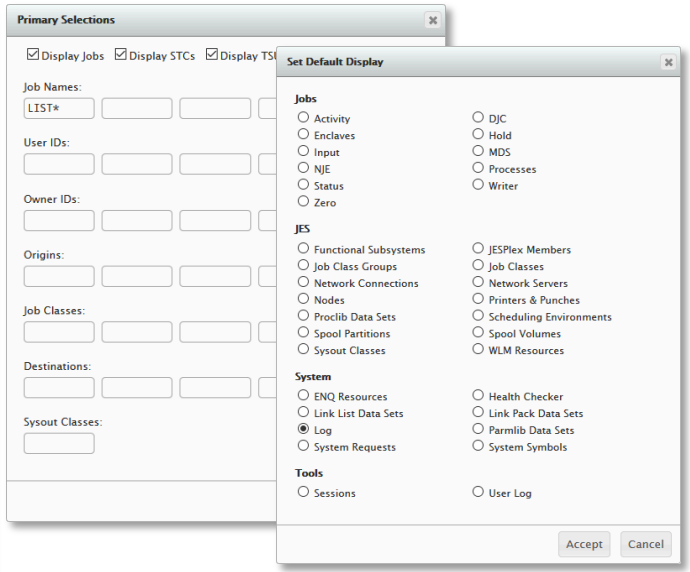
Automatic carriage control conversion for local printing. Full-featured PDF document creation.

# (E)JES Web

- When you log in for the first time, the default display is the Status display. It is filtered by your Owner ID.

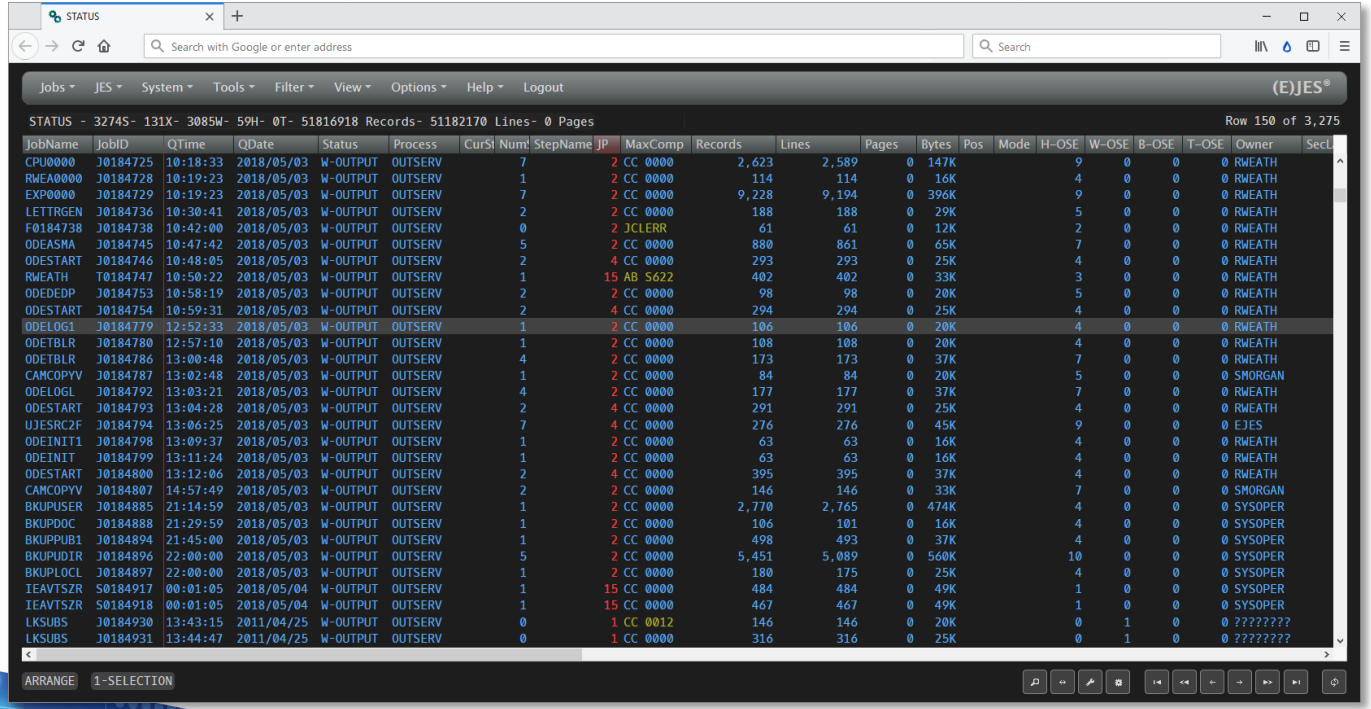
- You can change your filters and default display via dialog boxes. Your preferences are stored on the host and are used even when you log in from a different device or browser.



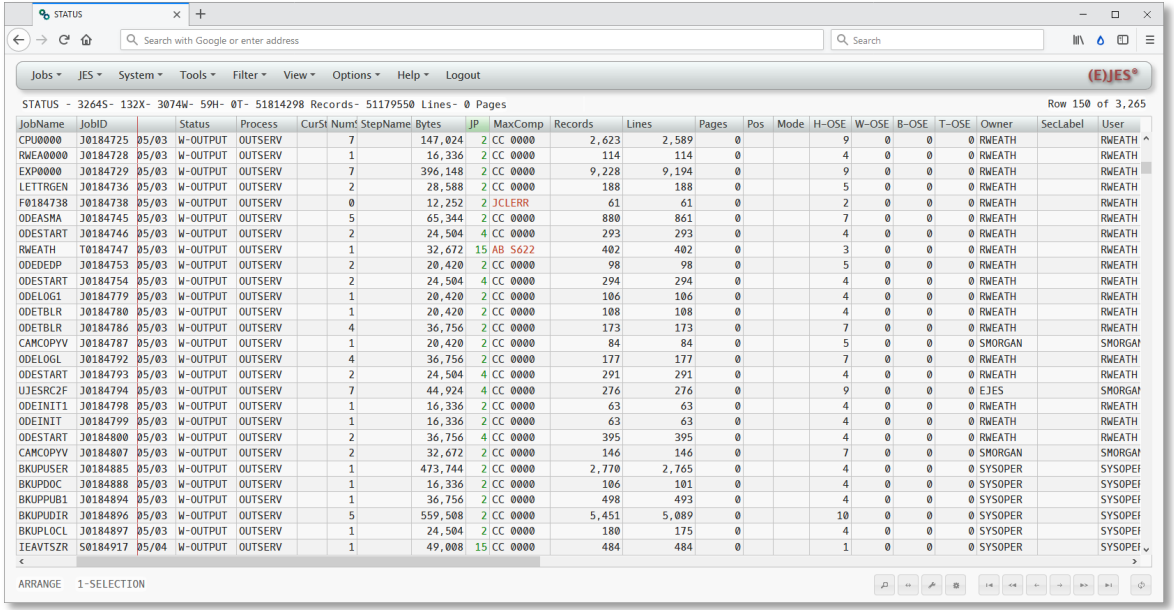
# (E)JES Web

- A number of different color schemes and other display options are supported. This color scheme approximates the colors use on a 3270 terminal.



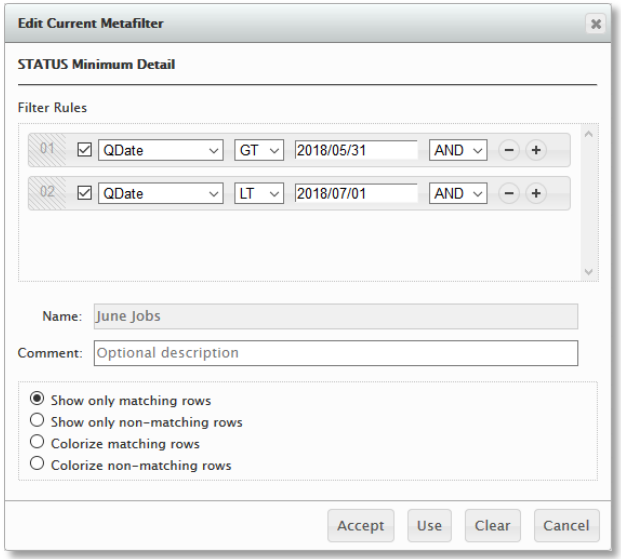
# (E)JES Web

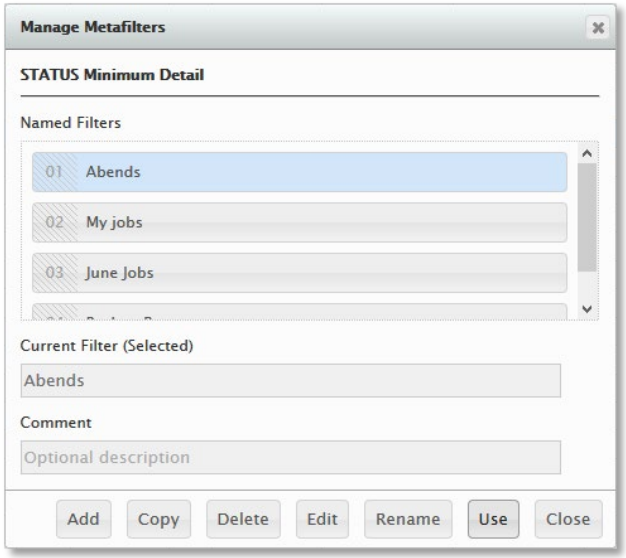
- Full featured browser-based application – not just a viewer
- Column reorder via drag and drop
- Column resizing
- Sort data by clicking column headers – up to four sort keys are supported



# (E)JES Web

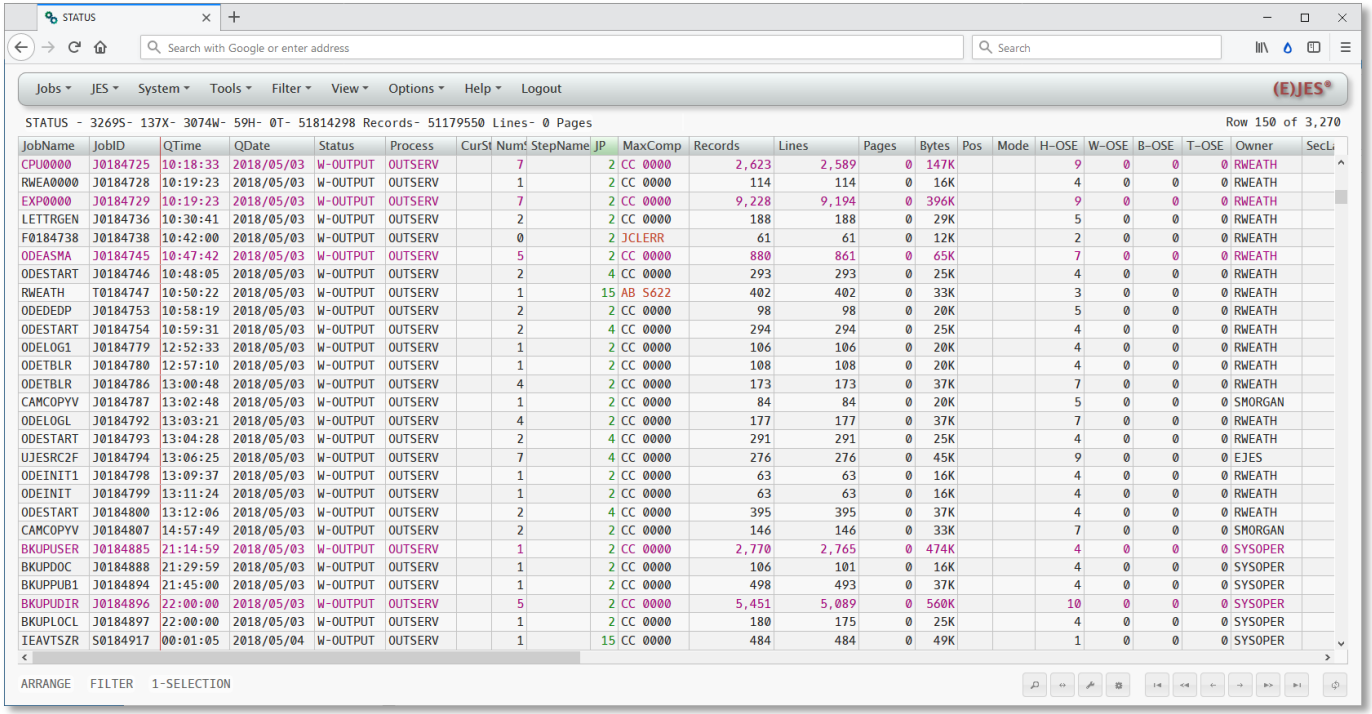
- (E)JES Web supports metafilters, which are customized column-based filters for tabular displays.  Users can save multiple metafilters.
- Metafilters are stored on the host and can be used even when you log in from a different device or browser.



# (E)JES Web

- You can have (E)JES Web colorize the records that match your filter

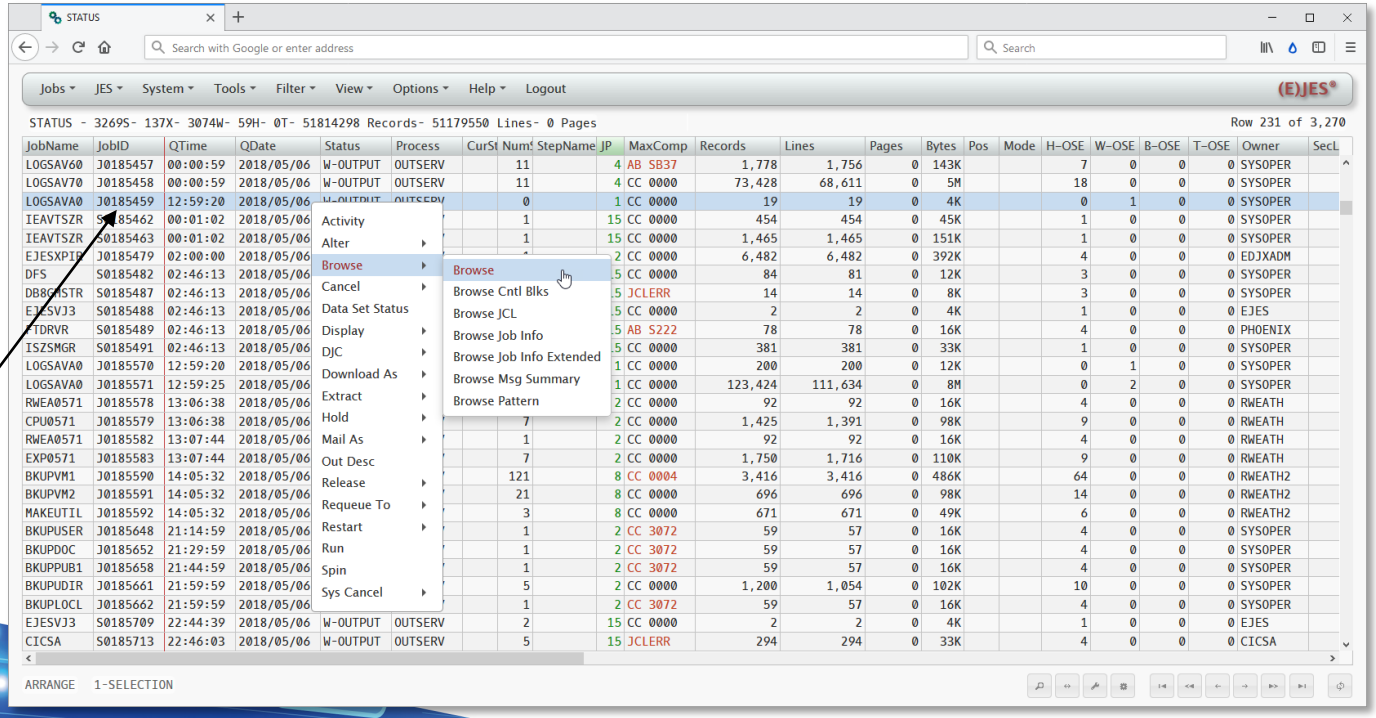


# (E)JES Web

- Select rows using your mouse or your keyboard using standard conventions

- Line commands are accessible via a contextual menu when you right-click

**Right-click a row to access commands.**

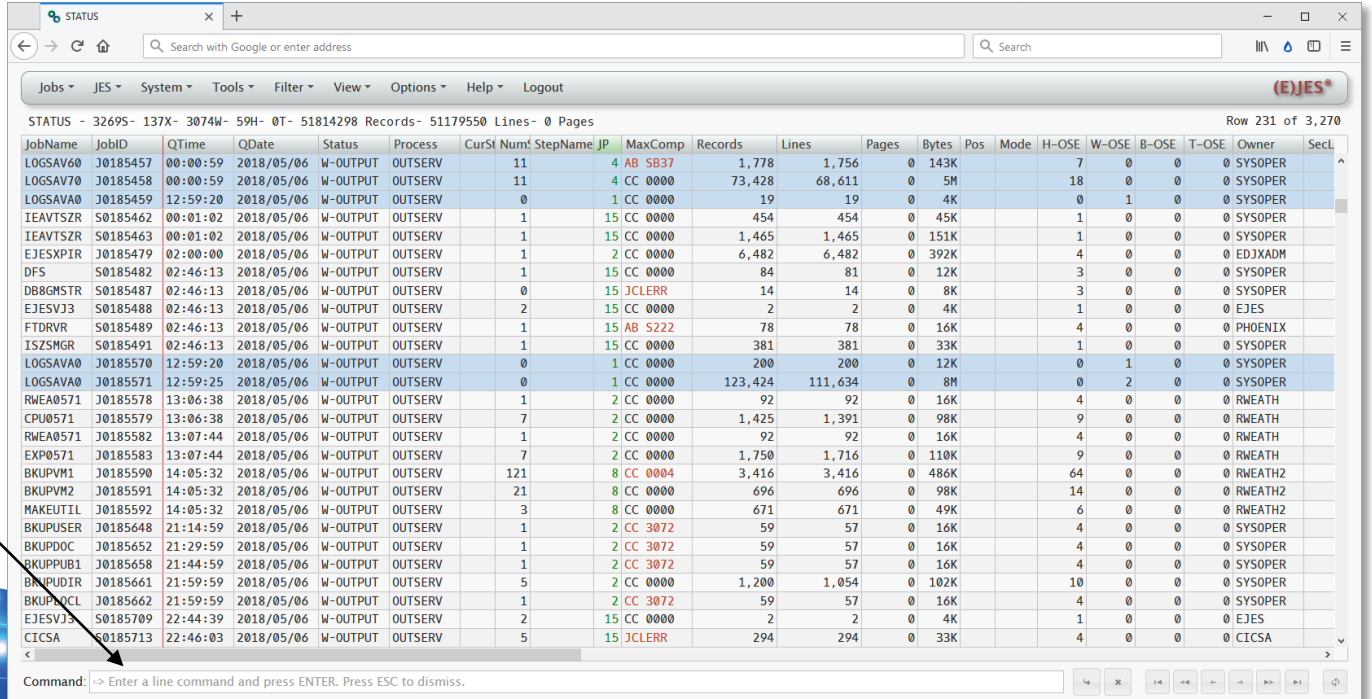


# (E)JES Web

- You can also issue line commands by typing a **colon**. An input field appears.

- Type a **semi-colon** to display a primary command field. Here you can enter a navigation command instead of using the menu.

**Type your command here.**

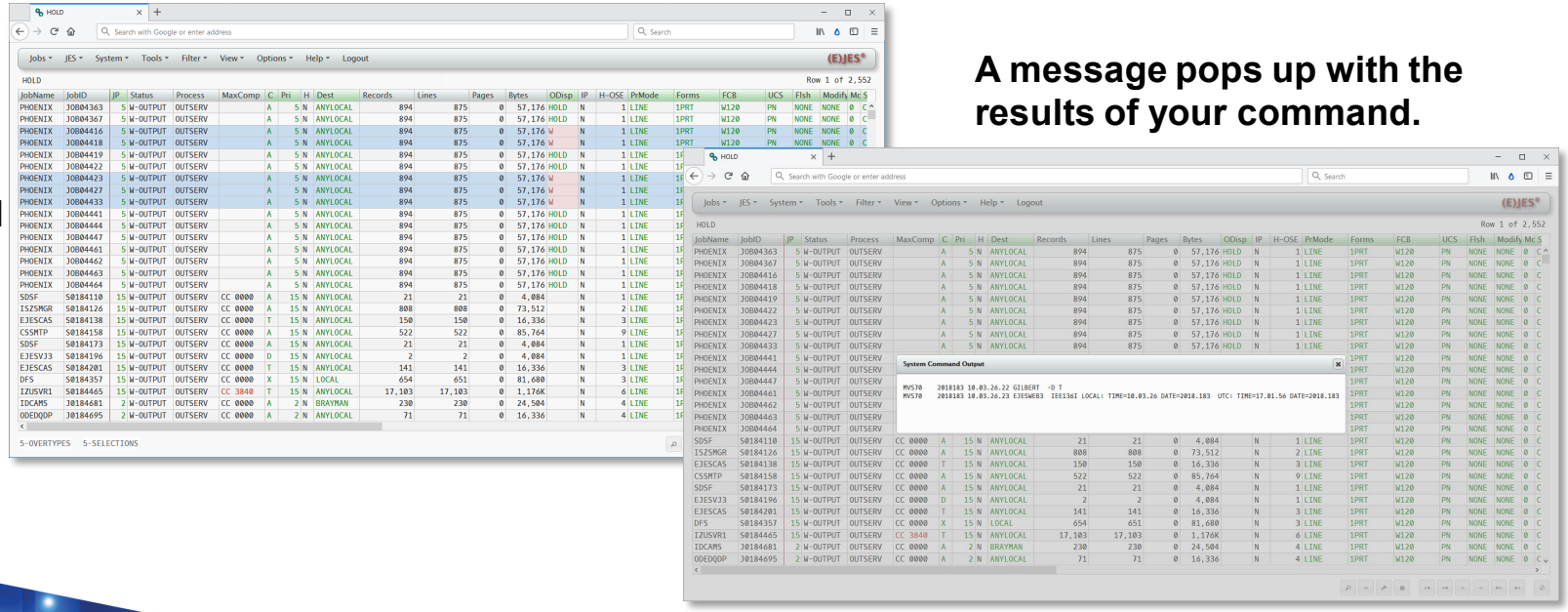


# (E)JES Web

- A green background on a column heading means a column is overtypeable.

- Propagate overtyped values by selecting multiple rows.

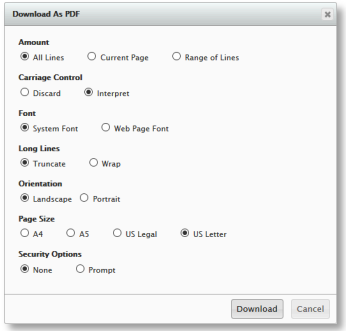
**Press Enter to issue the command or Esc to cancel.**

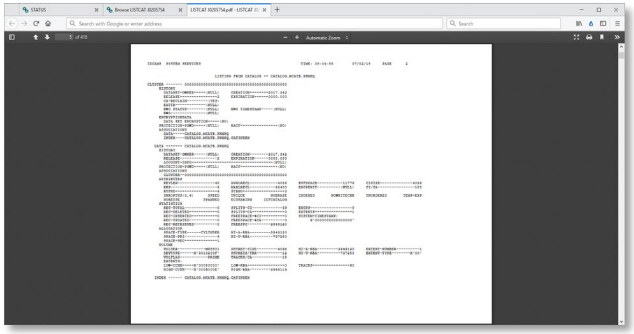


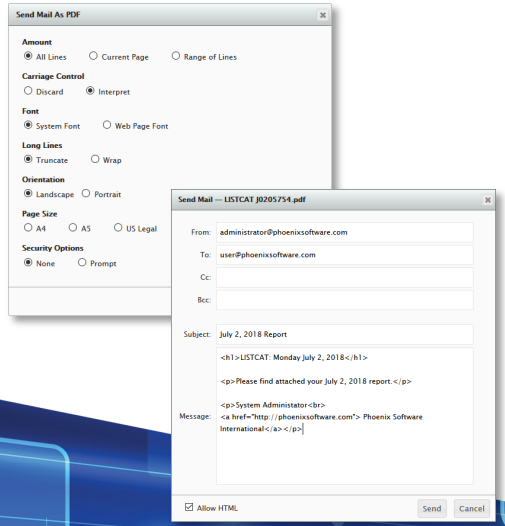

**A message pops up with the results of your command.**

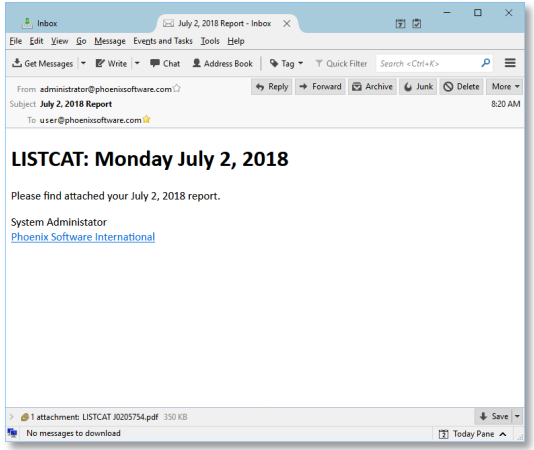
# (E)JES Web

- You can download and/or email data as text or PDF by right-clicking on a row and choosing **Download as Text**, **Download as PDF**, **Mail as Text**, or **Mail as PDF**.



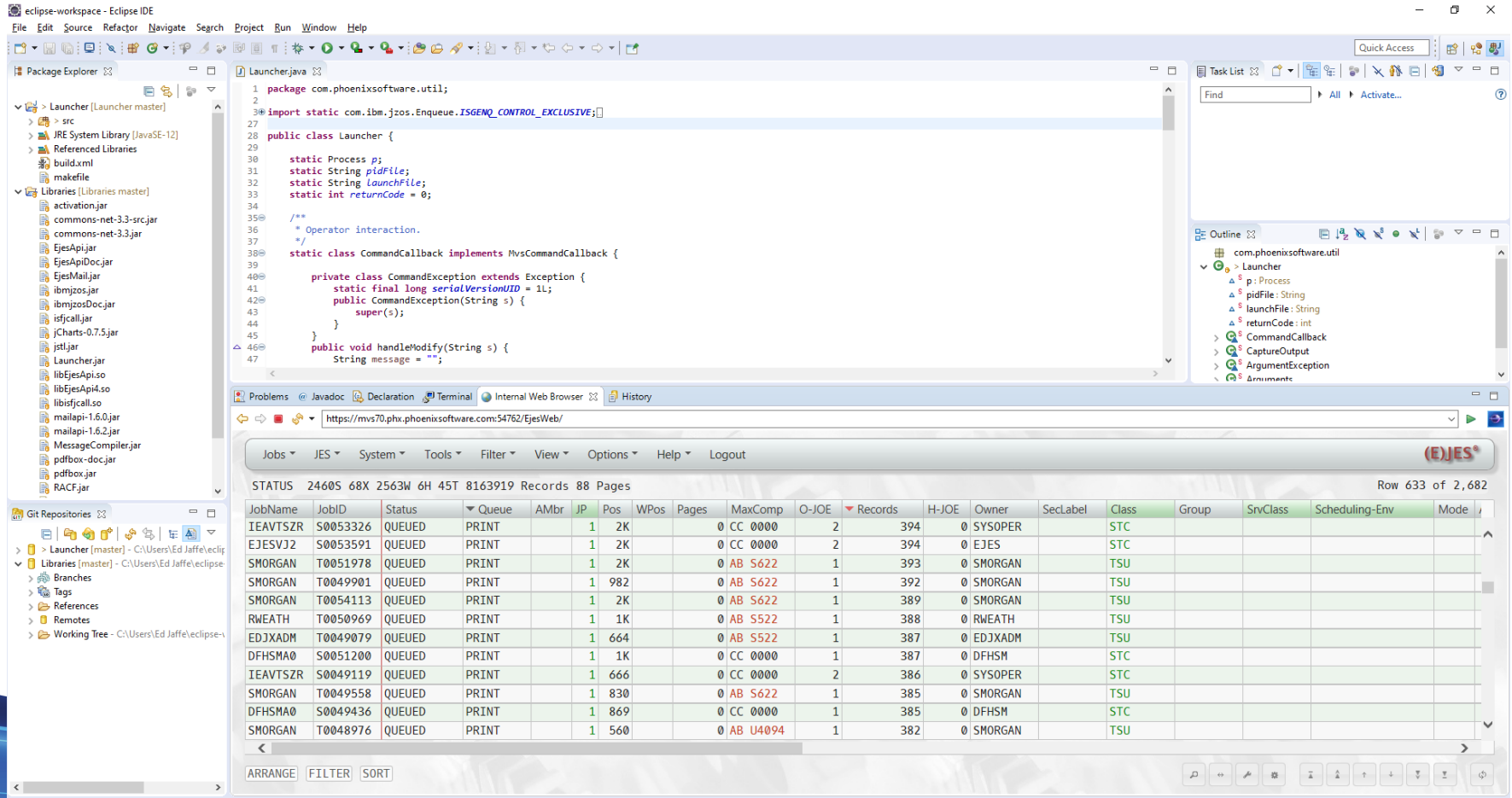


# Eclipse-based IDEs

- Eclipse is an open source framework built on software originally provided by IBM.
- It can be used to build any user interface, but is most traditionally associated with integrated development environments (IDEs).
- Eclipse provides the de facto Java IDE and is used by programmers worldwide.
- Current generation programmers have almost certainly used Eclipse.
- It's extremely extensible, built on a plug-in architecture that allows features to be created and added on demand.
- There's a rich set of existing plug-ins available via the Eclipse marketplace, some free and some charged.
- To truly attract the best and the brightest, a modern, state-of-the-art development environment is a must.
- Expect your next generation of mainframe programmers to already be familiar with Eclipse.

# Eclipse-based IDEs



# Eclipse-based IDEs

Want to learn more about Eclipse: https://www.eclipse.org/

# Objective

- Discuss how Phoenix Software International is applying the use of present-day tools with the mainframe such as:

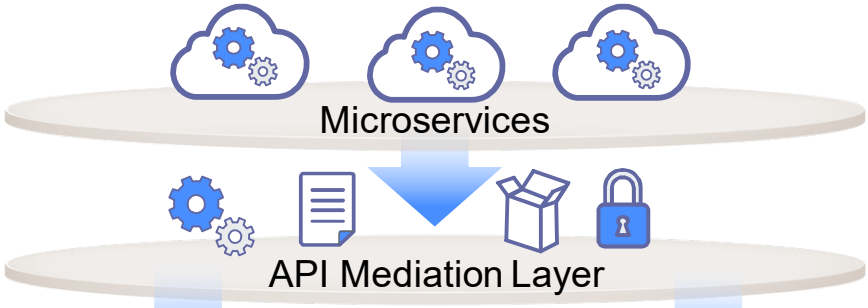
  - (E)JES, (E)JES Web and Eclipse – A modern, lightweight browser-based system management tool for users who prefer not to work in a 3270 and for developers who want to leverage the Eclipse IDE for development
  - **Zowe – Open source framework for the mainframe that provides solutions that allow development and operations teams to securely manage, control, script, and develop on the mainframe like any other cloud platform**
  - z/OSMF Workflow – automation of routine procedures and tasks through the use of a modern interface and Zorow – An open source community dedicated to contributing and collaborating on z/OSMF Workflows

# Zowe

- An extensible framework for connecting applications and tools to mainframe data and applications.
- Aims to make the mainframe an integrated and agile platform within the changing IT architectural landscape.
- First open source project on z/OS. All code is licensed under the Eclipse Public License version 2.0

# Zowe

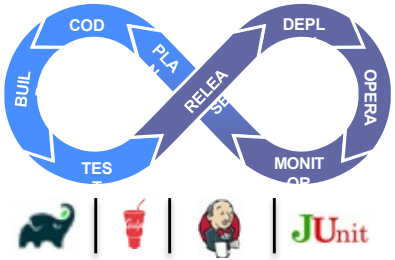
## Framework and Ecosystem at a Glance



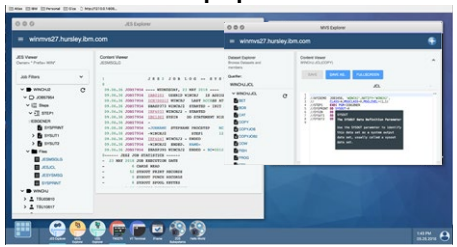

Zowe Conformant

- Plugins
- Plugins
- Plugins

CLI




App



Zowe Conformant

- Plugins
- Plugins
- Plugins

# Zowe

## Anyone can participate

- The Zowe community is vital to driving innovation
- By joining you can become part of the development of a vibrant ecosystem of applications for the mainframe
- Zowe is open to all for participation (contributor, committer, conformant)
- Any mainframe organization, vendor, ISV, consultant, or user in the mainframe community can participate in the development of Zowe

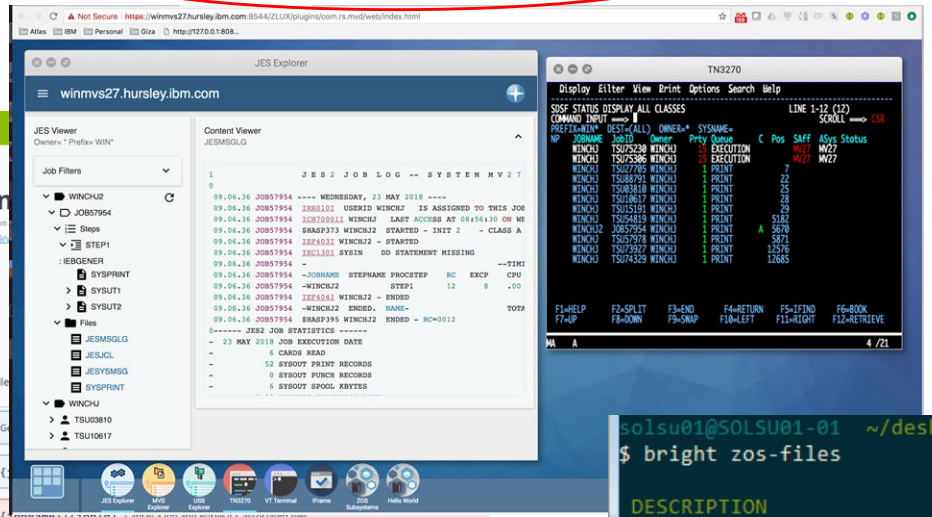
# Zowe

## Ecosystem enablement thru Zowe Conformance

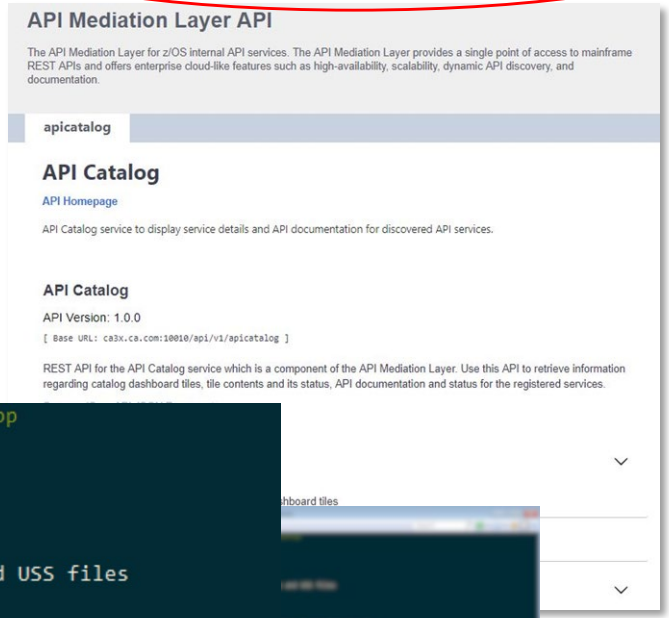
- Vendors and ISVs are able to build applications that leverage or build on top of the Zowe Framework
- Building on this framework speeds up vendors time to market and enables easier integration with other Zowe Conformant applications

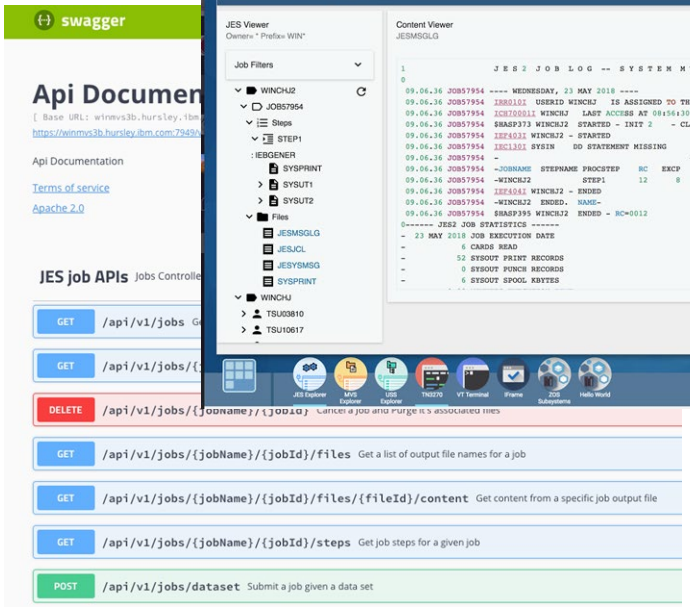
# What is Zowe?

Browser-based Web Desktop

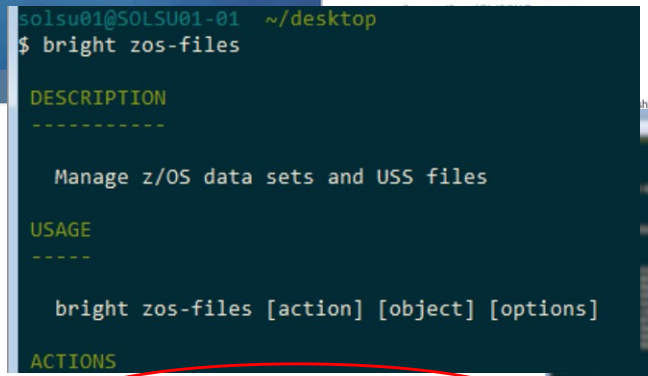



API Mediation Layer
(Gateway, Discovery Service, Catalog)



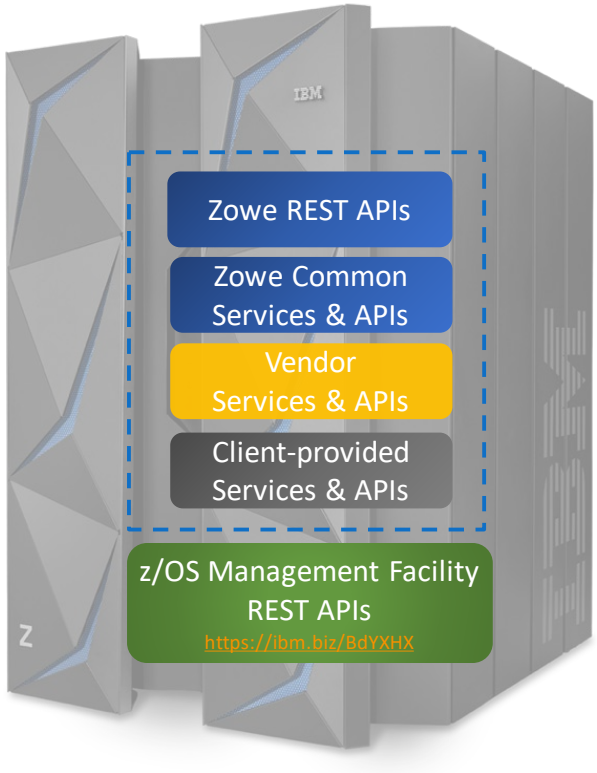

Swagger-defined z/OS REST APIs




Node.js- based CLI

# Zowe REST Services

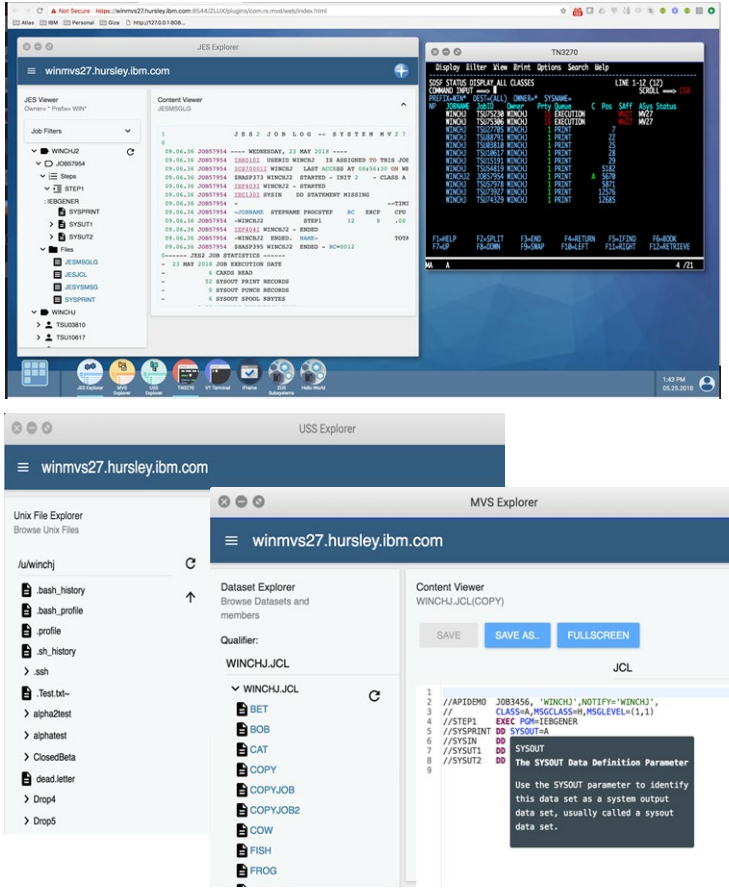
- Industry standard REST interfaces to z/OS resources that are language and platform neutral, stateless, and scalable
- Foundational building blocks for system services
  - *Dataset APIs*
    - Create, read, update, delete, and list data sets
  - *JES APIs*
    - View the information and files of jobs, and submit and cancel job
  - *USS APIs*
    - Create, read, update, and delete USS files
  - *System APIs*
    - View information about PARMLIB, SYSPLEX, and USER



# Zowe Web Desktop – an app container in a browser

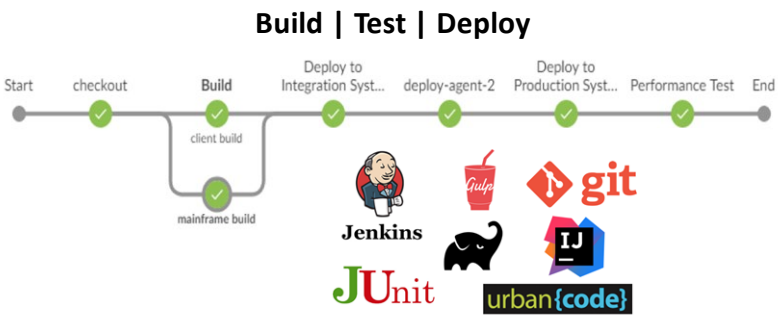
- Known as zLUX, the Zowe web UI is a virtual desktop system that offers a rich and open platform for a web-based mainframe user experience

  - *Mainframe Virtual Desktop*
    - A web-based window manager that provides full screen interactive experience
  - *Zowe Node Server*
    - Runs zLUX; uses Express.js as web service framework for communication between applications and z/OS services and components, pre-reqs Node.js for z/OS
  - *ZSS Server*
    - Provides secured REST API services
  - *Application Plug-in*
    - Data services, Configuration data service, URI broker, app-to-app communication, Error reporting UI, Logging utility
  - *Explorers*
    - JES, MVS, USS explorers
    - Basic editing support for REXX and JCL



# Zowe CLI – Enables cloud-like access to mainframe

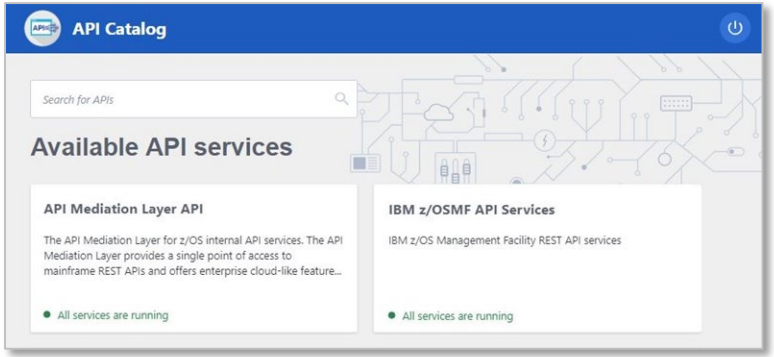
- Enables app developers and DevOp engineers to interact with the mainframe easily through a command-line interface (CLI) from any terminal on Windows, MacOS, and Linux
- Easily integrates with IDEs, shell commands, bash scripts, and build tools; installs using NPM
  - *Interact with mainframe files*
    - Create, edit, download, and upload mainframe files (data sets) directly
  - *Submit jobs*
    - Submit JCL from data sets or local storage, monitor status, view and download output automatically
  - *Issue TSO and z/OS console commands*
    - Issue TSO and console commands to the mainframe directly
  - *Integrate z/OS actions into scripts*
    - Build local scripts that accomplish both mainframe and local tasks
  - *Produce responses as JSON documents*
    - Return data in JSON format on request for consumption in other programming languages
  - *CLI plug-ins*
    - Access to CICS and DB2

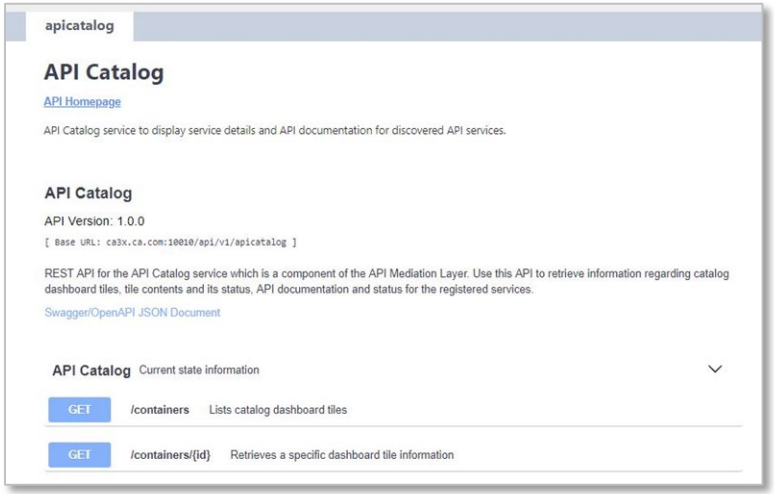



```
GROUPS
------

  plugins                Install and manage plug-ins
  profiles               Create and manage configuration profiles
  provisioning | pv      Perform z/OSMF provisioning tasks on Published Templates
                         in the Service Catalog and Provisioned Instances in the
                         Service Registry.
  zos-console | console  Issue z/OS console commands and collect responses
  zos-files | files      Manage z/OS data sets
  zos-jobs | jobs        Manage z/OS jobs
  zos-tso | tso          Issue TSO commands and interact with TSO address spaces
  zosmf                  Interact with z/OSMF

OPTIONS
-------

  --version  | -V (boolean)

     Display the current version of CA Brightside

GLOBAL OPTIONS
--------------

  --response-format-json  | --rfj (boolean)

     Produce the command response as a JSON document

  --help  | -h (boolean)
```

# Zowe API Mediation Layer –Gateway to mainframe APIs

- Enables a single point of access to mainframe APIs with high-availability, scalability, dynamic API discovery, consistent security, “one-time” sign-on experience, and unified standard API documentation (OpenAPI / Swagger)
  - *API catalog*
    - UI Catalog of available APIs with their Swagger doc and service status
  - *Gateway*
    - Single secure point of entry to an ecosystem of API services. Hides complexity. Highly available.  Based on Netflix Zuul.
  - *Discovery service*
    - Discover APIs across many applications. Repository of active API services. Based on Netflix Eureka.



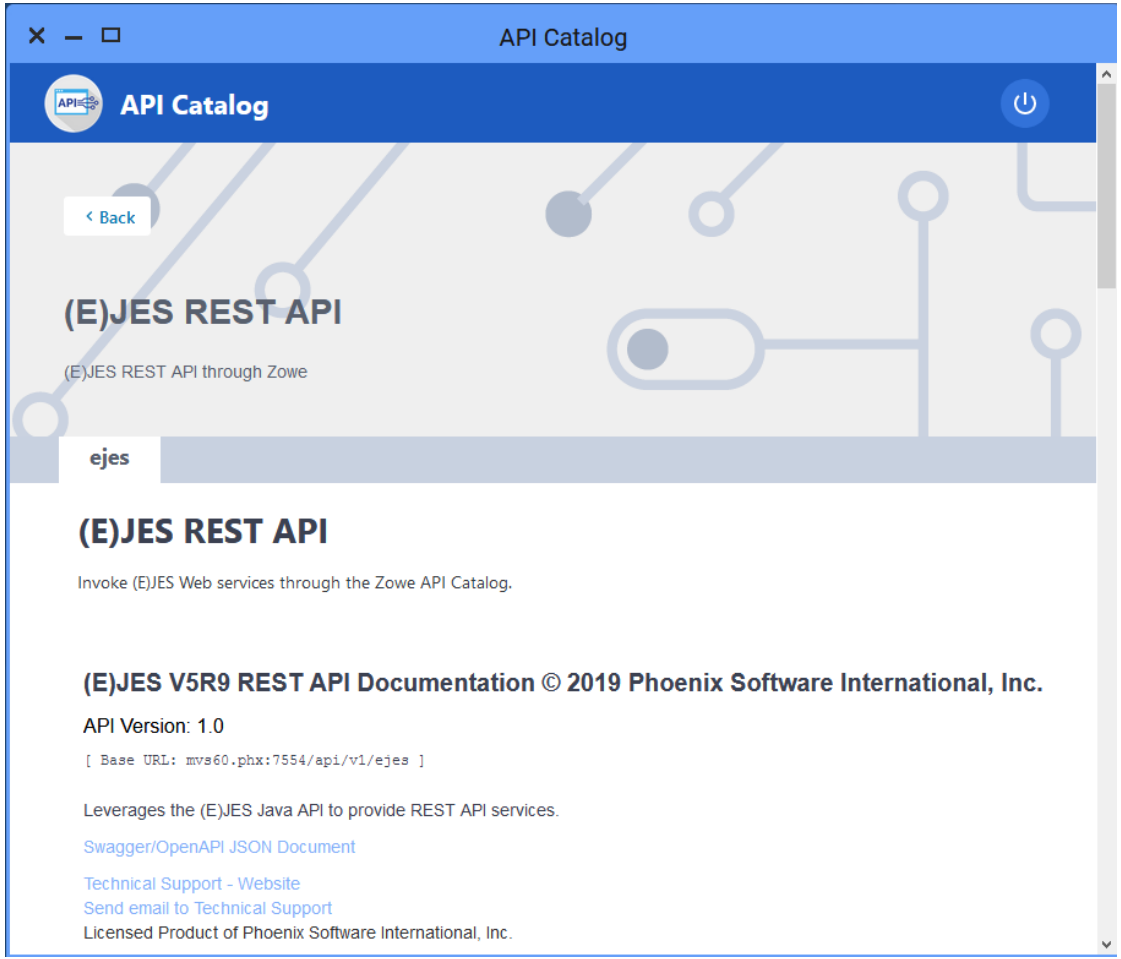



# Zowe – New for (E)JES V5R9

- Our Zowe conformant offering consists of three parts which are delivered and enhanced using a continuous delivery model:
  - A **REST API** (a component of the (E)JES Web server) providing remote users with access to the complete (E)JES API.
  - A **Command Line Interface (CLI)** which leverages the REST API to make accessing (E)JES mainframe resources easy.
    - Currently supports real-time streaming of operlog/syslog and is being enhanced to support accessing all of the information (E)JES provides.
    - The CLI hides the complexities of the REST API and streams the result to STDOUT
  - A **Desktop Application** (planned). This will be an implementation of the existing (E)JES Web interactive browser interface, but will seamlessly integrate with the Zowe Desktop, including look and feel.

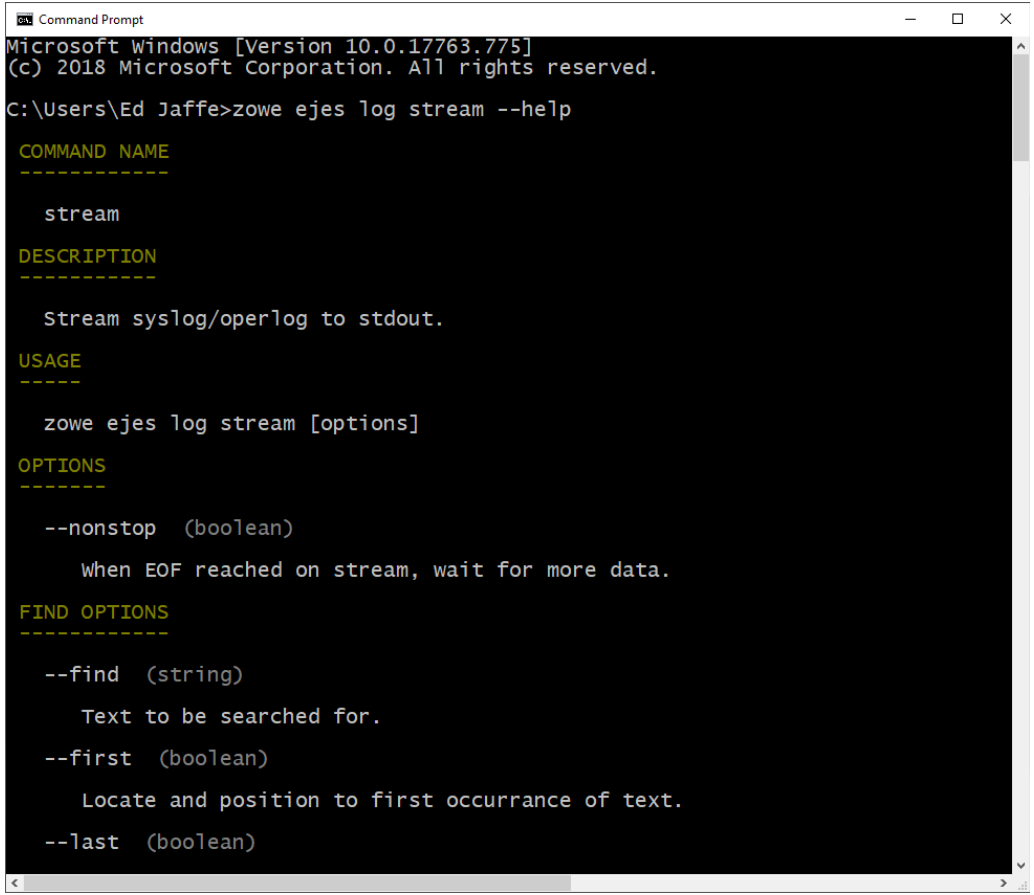
# Zowe (E)JES REST API

- The REST API is a thin layer around the (E)JES Java API.
  - A complete API
- With only a few specialized exceptions, all responses are JSON objects.
- Interfaces are documented using Swagger/OpenAPI



# Zowe (E)JES Command Line Interface (CLI)

- The CLI hides REST API complexities behind a simple command line interface.
- Supports Windows, MacOS and Linux.
- Output is to STDOUT.
- Can be used interactively or in a script.
- Easy to use --help and --help-web options limit the need for written documentation.
- Roadmap for enhancements



# Zowe (E)JES CLI Query Syntax Roadmap

EJES query

[--h | --help]
[--H | --HELP]
[--v | --version]
[--n | --dry-run]
[[--u | **--use]** [**status |** list | *table-name*] ]
[[--t | --table] [table-default | default | list]]...
[[--r | --report] [**browse-default** | list]]
[[--c | --command] [list | command]]
[[--m | --metafilter] [**off** | list | *metafilter_construct*...]]…
[[--s | --select] [**off** | list | *select_construct*]
[**--all]**
[--start [*num* | **first**]
[--count [*num* | **last**]
[--**cc** [**discard** | keep | interpret]
[--**header on** | off]

[[--o | --**owner**] [**logged_on_userid** | "ownername..."]]
[[--j | --jobname] jobname...]
[--user "*username*..."]
[--origin "*origin*..."]
[--class "*class*..."]
[--dest "*destination*..."]
[--sysclass "*sysclass*..."] ]
[--**job *on*** | *off*]
[--**stc *on*** | *off*]
[--**tsu *on*** | *off*]
[--**atx *on*** | *off*]
[--sort [list | std | *sort_construct* ]
[--showcols *column-key*... | list]
[--hidecols *column-key...* | list]
[--rows **60** | *num* | all]
[--lines **1000** | *num*]
[[--maxcommands | -x] **1** | *num*]

# Objective

- Discuss how Phoenix Software International is applying the use of present-day tools with the mainframe such as:

  - (E)JES, (E)JES Web and Eclipse – A modern, lightweight browser-based system management tool for users who prefer not to work in a 3270 and for developers who want to leverage the Eclipse IDE for development

  - Zowe – Open source framework for the mainframe that provides solutions that allow development and operations teams to securely manage, control, script, and develop on the mainframe like any other cloud platform

  - **z/OSMF Workflow – automation of routine procedures and tasks through the use of a modern interface and Zorow – An open source community dedicated to contributing and collaborating on z/OSMF Workflows**

# What is a z/OSMF Workflow?

- Workflows first appeared in 2013 with z/OS 2.1.
  - Initially supported instructions and batch job submission only.
  - When immediate execution steps, feedback, and other features were added via continuous delivery late in the z/OS 2.2 timeframe, it started looking interesting.
- With z/OSMF Workflow, a framework is available to z/OS system programmers to allow them to define a guided flow (workflow) through steps to accomplish a system management or configuration task.
- z/OSMF Workflow also provides RESTful APIs that allow users to run workflows programmatically

# What is a z/OSMF Workflow?

- The z/OSMF Workflow is useful to:
  - Assist people unfamiliar with how to perform a given task, or a task that they perform rarely
  - Ensure that all tasks are performed in the right order and only when their dependencies have been met
  - Ensure that all steps are completed
    - Even if many of the tasks have been delegated to a number of different colleagues
  - Monitor and track progress toward the completion of the task
  - Provide a history (audit trail) of the steps performed for a task
  - Perform the same tasks on multiple systems
    - Enabling a function (e.g. zEDC)
    - Upgrading a new release of software (e.g., z/OS)

# Current challenges with z/OSMF Workflow adoption

- z/OS system programmers have historically built their own home grown processes to perform common system management tasks
- Organizations want to reduce the complexity of their z/OS management processes so they can transfer their knowledge to the early tenure staff
  - Additional skills are needed to create z/OSMF Workflows
- How can early tenure and more experienced z/OS system programmers share best practices and common workflow patterns?

# Zorow (Z Open Repository of Workflows)

- Provides a repository for z/OS systems programmers and product vendors to contribute and share z/OSMF workflows
  - All workflows made available under an Apache 2 license
- Community is led by both vendors along with customers, and open to anyone to participate.
  - Vendor-neutral open source governance established with the guidance of the Open Mainframe Project.

# z/OSFM Workflow Components

- Prologue
- Parent steps and leaf steps
- Content for each step
  - Name, Title & Description
  - Instructions (shown at “Perform” time)
- Metadata for each step
  - Dependencies (prerequisites and conditions)
  - Weight and various flags
  - Variable names, descriptions, help pop-ups
- Program templates (support variable substitution)
  - For JCL you have a batch job
  - For TSO-REXX-JCL, TSO-REXX, and TSO-UNIX-REXX you have a REXX exec
  - For shell-JCL and TSO-UNIX-shell you have a shell script

Convert from existing product documentation

# Creating a Workflow from Existing Documentation

- Our existing product documentation lives in Microsoft Word documents.
- From that base we create and distribute Adobe PDF, Raw HTML, and Eclipse Plug-Ins intended for use with IBM Knowledge Center for z/OS (KC4z).
  - KC4z has been around over four years (since z/OS 2.2).
  - If you haven't deployed it yet, you should!
  - If you're an ISV and you don't provide plug-ins for it, you should!
- We convert the Word documents to these other formats using helpful software (MadCap Flare) and self-authored scripts.
- Ideally, our Product Installation Workflow should eventually be able to completely replace our existing installation documentation. Therefore, it should look every bit as good as the other formats. It should support colors, sizeable fonts, embedded graphics, internal links, external links, and everything else that makes modern documentation usable.

Cut Copy Format Painter Paste — Clipboard

Switzerland 14 — Font

Paragraph

AaBbCcD Bold | AaBbCcDc ¶ Caption | AaBbCcDd Change | AaBbCcDc ¶ Code Lis... | AaBbCcDd CodeRed — Styles

Find Replace Select — Editing

## Navigation

Search document

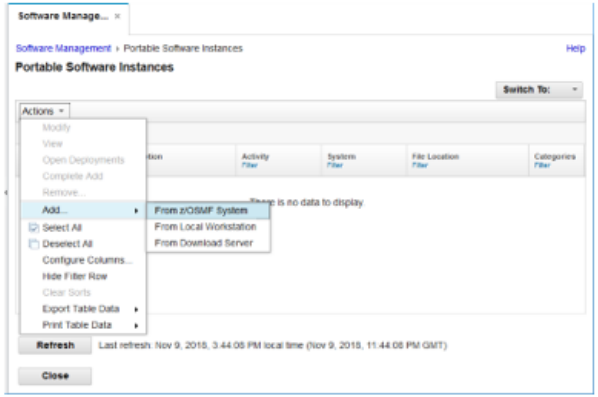
Headings Pages Results

- Perform APPLY of Product Components
- Allocate Distribution Libraries
- Perform ACCEPT of Product Components
- Chapter 4. Perform Tasks for CSM Installation Path
- Run the EJPINST Utility
- Interact with the EJESPFI Utility
- Add (EJES) Product to the CSM Software Catalog
- Add (EJES) Install Package to CSM Software Cata...
- Initiate Product Install
- Choose Install Type
- Specify System Data Set Names
- Extract Customization Library from CSM Package
- Perform Post-CSM Install Product Customization
  - Customization USERMOD/PTF Member Trans...
  - Installing Customization USERMODs/PTFs wit...
- Chapter 5. Perform Tasks for ZMF Installation Path
- Unlock and Unpack the Portable Software Instan...
- Add the Portable Software Instance to Software...
- Deploy the Portable Software Instance

WORD

## Add the Portable Software Instance to Software Management

From the z/OSMF Software Management main tab select "Portable Software Instances". Then click "Actions" > "Add…" > "From z/OS System" to invoke the "Add Portable Software Instance" dialog.

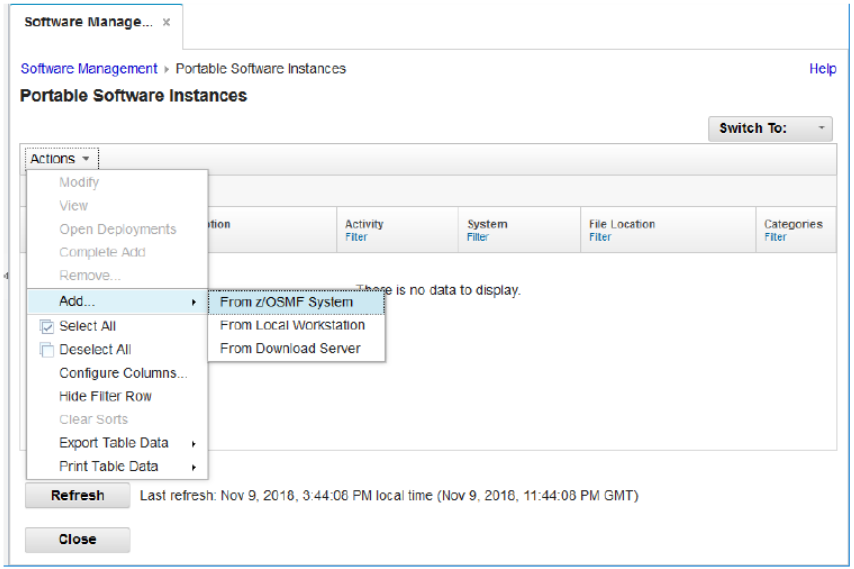


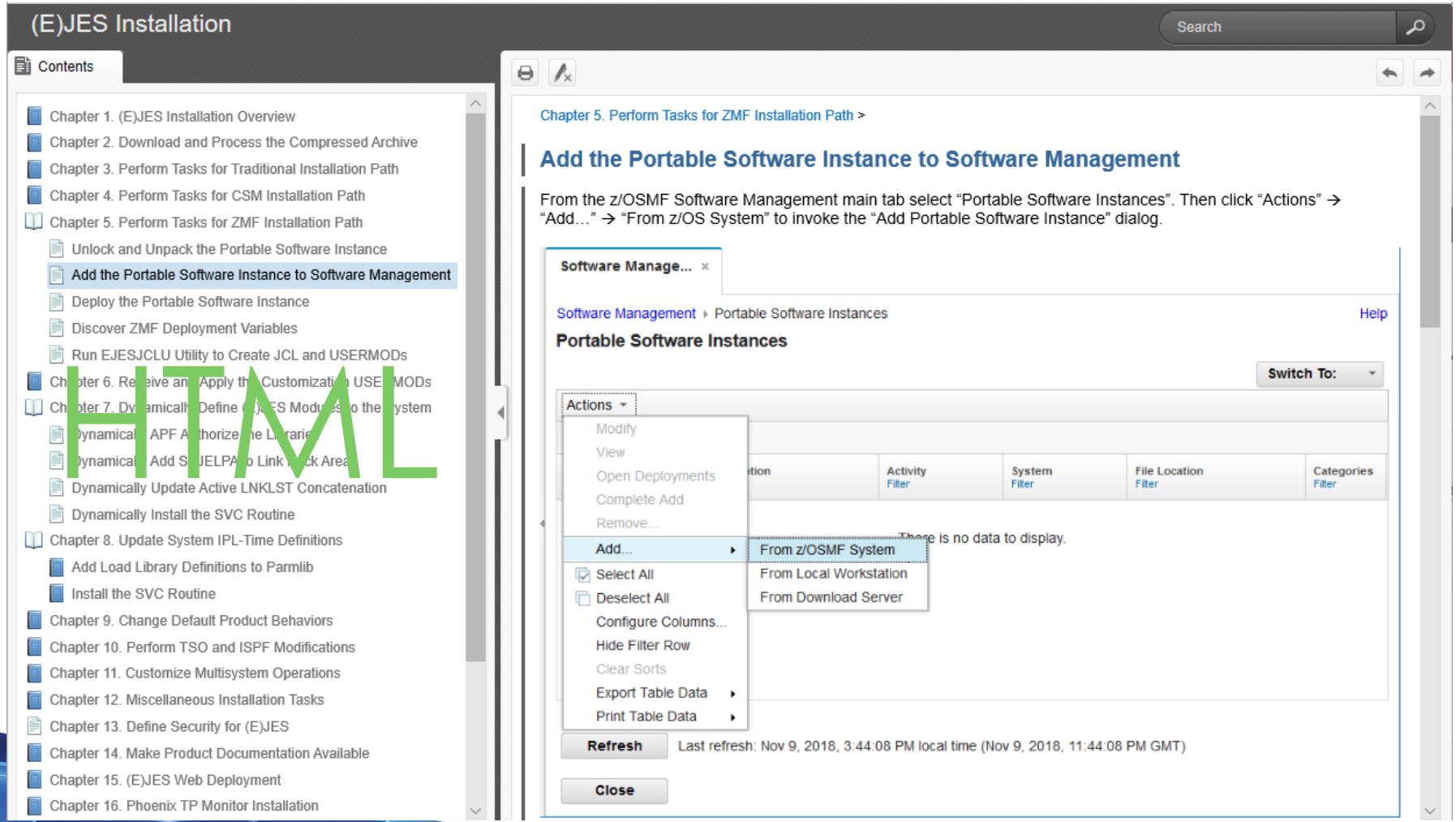

39

# Add the Portable Software Instance to Software Management

From the z/OSMF Software Management main tab select “Portable Software Instances”. Then click “Actions” → “Add…” → “From z/OS System” to invoke the “Add Portable Software Instance” dialog.

PDF




(E)JES Installation
Search
Contents
Chapter 1. (E)JES Installation Overview
Chapter 2. Download and Process the Compressed Archive
Chapter 3. Perform Tasks for Traditional Installation Path
Chapter 4. Perform Tasks for CSM Installation Path
Chapter 5. Perform Tasks for ZMF Installation Path
Unlock and Unpack the Portable Software Instance
Add the Portable Software Instance to Software Management
Deploy the Portable Software Instance
Discover ZMF Deployment Variables
Run EJESJCLU Utility to Create JCL and USERMODs
Dynamically Update Active LNKLST Concatenation
Dynamically Install the SVC Routine
Chapter 8. Update System IPL-Time Definitions
Add Load Library Definitions to Parmlib
Install the SVC Routine
Chapter 9. Change Default Product Behaviors
Chapter 10. Perform TSO and ISPF Modifications
Chapter 11. Customize Multisystem Operations
Chapter 12. Miscellaneous Installation Tasks
Chapter 13. Define Security for (E)JES
Chapter 14. Make Product Documentation Available
Chapter 15. (E)JES Web Deployment
Chapter 16. Phoenix TP Monitor Installation
HTML
Chapter 5. Perform Tasks for ZMF Installation Path >
Add the Portable Software Instance to Software Management
From the z/OSMF Software Management main tab select “Portable Software Instances”. Then click “Actions” → “Add…” → “From z/OS System” to invoke the “Add Portable Software Instance” dialog.
Software Manage...
Software Management ▸ Portable Software Instances
Help
Portable Software Instances
Switch To:
Actions
Modify
View
Open Deployments
Complete Add
Remove...
Add...
Select All
Deselect All
Configure Columns...
Hide Filter Row
Clear Sorts
Export Table Data
Print Table Data
From z/OSMF System
From Local Workstation
From Download Server
Activity
Filter
System
Filter
File Location
Filter
Categories
Filter
There is no data to display.
Refresh
Last refresh: Nov 9, 2018, 3:44:08 PM local time (Nov 9, 2018, 11:44:08 PM GMT)
Close

IBM Knowledge Center

English

Search

Search Filters: (E)JES V5R9 × Auto-select | Clear All | Add Products...

< Previous | Next > | Topic View

Table of Contents

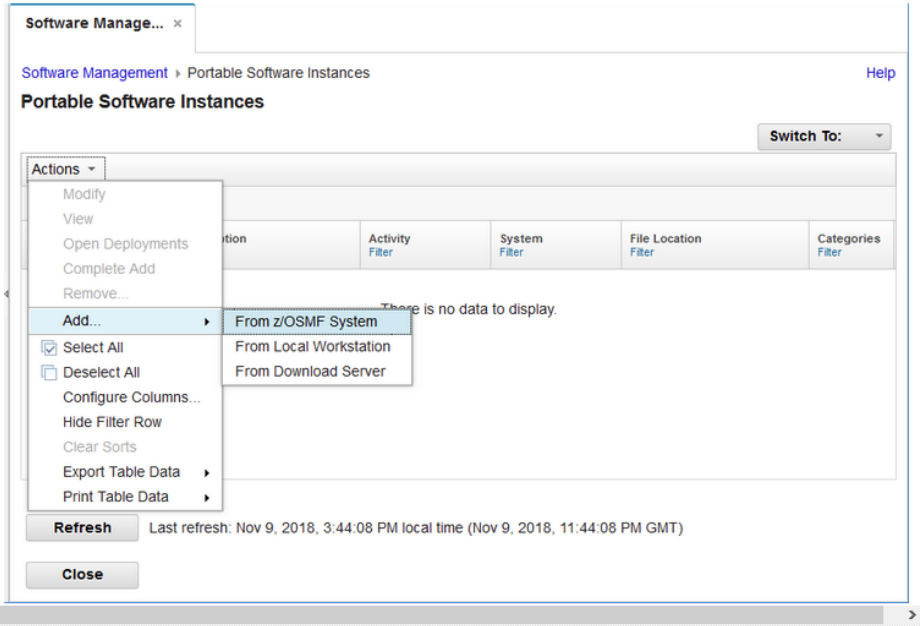
- Allocate Distribution Libraries
- Perform ACCEPT of Product Components
- **Chapter 4. Perform Tasks for CSM Installation Path**
- Run the EJESINST Utility
- Interact with the EJESPFI Utility
- Add (E)JES Product to the CSM Software Catalog
- Add (E)JES Install Package to CSM Software Catalog
- Initiate Product Install
- Choose Install Type
- Specify System Data Set Name
- Extract Customization Library from CSM Package
- Perform Post-CSM Install Product Customization
- **Chapter 5. Perform Tasks for ZMF Installation Path**
- Unlock and Unpack the Portable Software Instance
- Add the Portable Software Instance to Software Mana
- Deploy the Portable Software Instance
- Discover ZMF Deployment Variables
- Run EJESJCLU Utility to Create JCL and USERMODs
- **Chapter 6. Receive and Apply the Customization US**
- **Generate System Environment Table**
- Common Problems Generating the System Enviro
- Specify Assembled Installation Options
- Install the License

Search Results

(E)JES V5R9 > (E)JES Installation > Chapter 5. Perform Tasks for ZMF Installation Path > Add the Portable Software Instance to Software Management

## Add the Portable Software Instance to Software Management

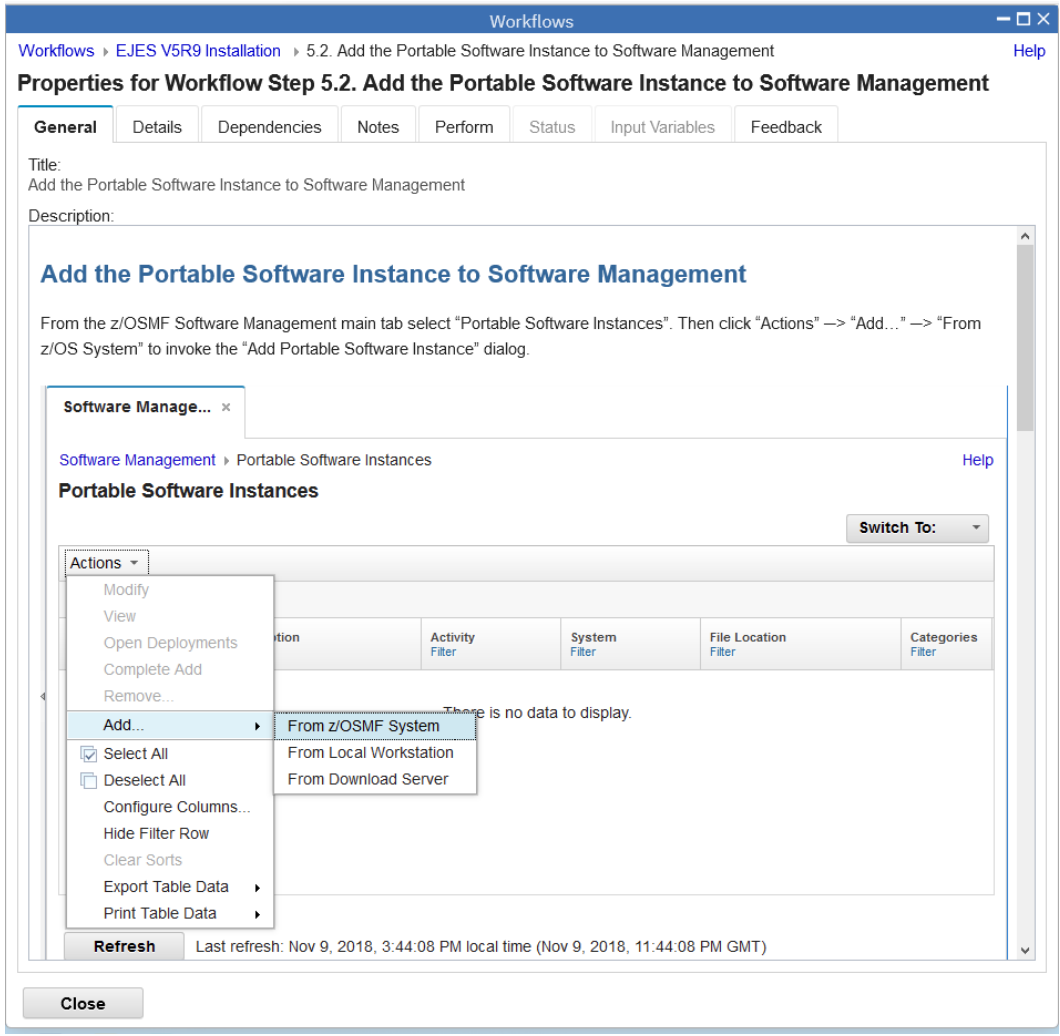
From the z/OSMF Software Management main tab select “Portable Software Instances”. Then click “Actions” à “Add…” à “From z/OS System” to invoke the “Add Portable Software Instance” dialog.




Contact | Privacy | Terms of use | Accessibility | Feedback

# z/OSMF Workflow Result

- Unlike the other formats, which are documentation only, the workflow is a tabbed display.
- Some tabs like **Details**, **Dependencies**, **Status** and **Input Variables** contain metadata that might be of interest to the sysprog user.
- The **Notes** tab lets you author and keep your own notes associated with this step.
- **Perform** actually “runs” the step.
- **Feedback** lets you answer survey questions to provide useful feedback about your experience.



# Our Content Transformation Approach

- It is possible to save a Word document as a **Word XML Document**.
- That's the format we probably would have used if we weren't already using MadCap Flare to help us create the HTML and Eclipse versions of our product documentation. No reason to believe it wouldn't work just as well for you.
- Instead, we had MadCap Flare convert the native Word document into XHTML documents – one for each chapter in the book.
  - What is XHTML?
  - From Wikipedia: XHTML documents are well-formed and may therefore be parsed using standard XML parsers, unlike HTML, which requires a lenient HTML-specific parser.
- We then ran a self-authored script (a Windows BAT file) against the XHTML files to transform them into XML files that are compatible with z/OSMF workflows.
  - That script is here: ftp://www.phoenixsoftware.com/pub/demo/workflow.bat

# Our Content Transformation Approach

- Workflows do not support style sheets (CSS). That is a MAJOR restriction for anyone attempting to directly author content using an XML editor such as the z/OSMF Workflow Editor. However, since our approach is to script a conversion from Word, it's really not terrible at all.
- The script uses the **sed** utility (originally from UNIX) on Windows to do the necessary transformations. Here's what one of the commands looks like to convert our "Railroad" style (used to document command syntax) into HTML:

- sed -i -E "s/<pre Railroad><code>/<pre style=\"padding:0;\"><code style=\"font-family: 'Courier New', Courier, monospace;font-size:100%%;border:none;background-color:#ffffff;\">/g" ..*Chapter*.xml
- At first we ran into an unacceptable number of restrictions. Generally, only simple HTML is supported by the Workflow and the Workflow Editor. We found this *extremely* disappointing because the published result looked like we were in an "HTML for Dummies" class learning the basics. ☹

# Our Content Transformation Approach

- Then we discovered the **<![CDATA[*some stuff*]]>** XML tag is accepted!
  - CDATA stands for Character Data and it means that the data in between the innermost brackets includes data that could be interpreted as XML markup, but should not be.
  - It looks a bit like an XML comment, but isn't. It's actually part of the document.
  - It's not perfect because there is no way to escape the CDEnd sequence **]]>**. Just don't use that anywhere and you should be OK.
- By wrapping all of our content in the CDATA tag, we were able to convince Workflow and Workflow Editor to ignore (and pass along) HTML tags they previously rejected as not supported and since the browsers understand those tags, that's what matters most.
- It would be best if Workflow and Workflow Editor were enhanced to natively understand everything, but what's there now is still very usable, especially for a scripted conversion situation.

# Our Content Transformation Approach

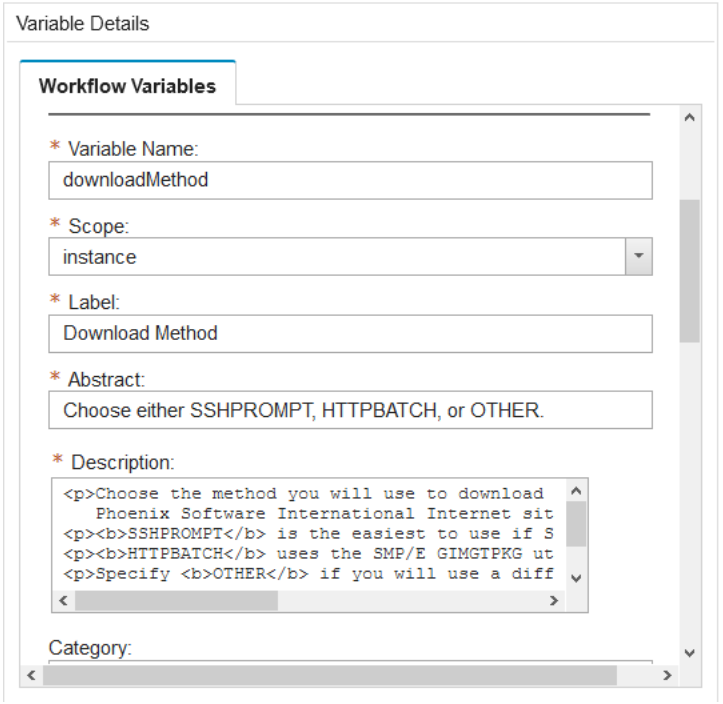
- We struggled with images for a while.
- When you “create” a workflow, you are actually copying it from whichever source location you specify into some undisclosed location deep inside z/OSMF.
- All file references are relative to that undisclosed location rather than the original source location. Explicit file references suffer “out of zone” security errors.
- Therefore, the only workable approach for images is to embed them as base-64 text within the workflow. Again, the trusty CDATA tag allows this to occur.
- This restriction is not all bad. There are advantages to having a single, self-contained workflow file that isn’t dependent on anything else. But it can become quite large if you have many images. It can also be cumbersome to edit the XML.

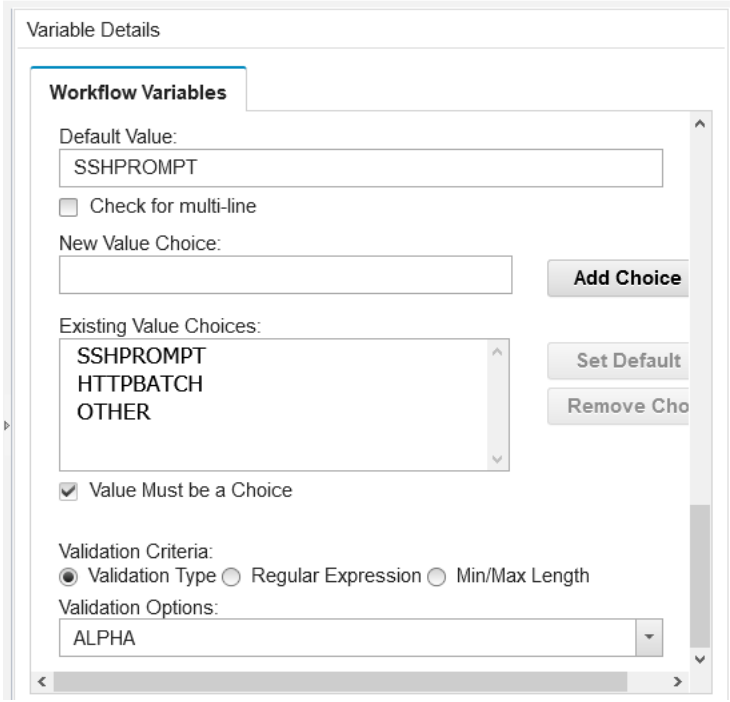
# Our Metadata Creation Approach

- The Workflow Editor does a pretty good job with most of the metadata
  - Beware! It has been shown to strip the CDATA tags around non-complex content if it thinks it understands everything therein.
  - This does not alter what the end user sees, but can cause confusion for anyone attempting to later inspect/edit the content XML directly. You might need to replace it with newly-converted text.
- Other than within a chapter, we generally implement dependencies on prior chapters (parent steps) rather than on any singular leaf steps within them.
- Within a chapter, we generally make each leaf step dependent on the prior one unless we want to allow parallel steps.
- We generally try to set the weight of a step that actually performs an action (submits a job, runs a script) ten times higher than one that is instructions only.
- We use simple, hand-written HTML for our variable descriptions and help/information pop-ups. You really don't need anything fancy there.

# Defining a Value Choice Variable

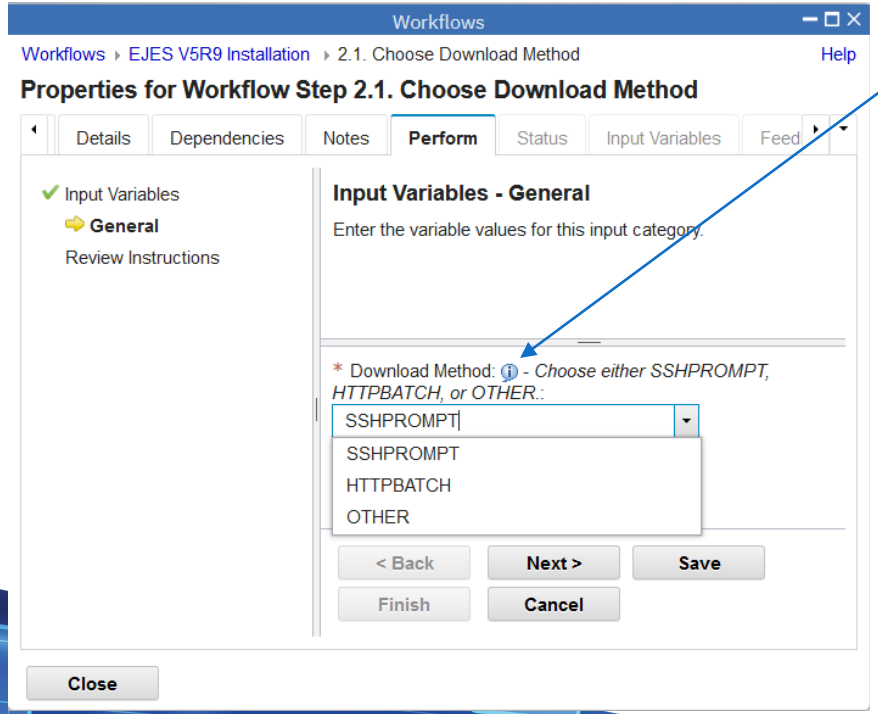
- Creates a drop-down list of choices



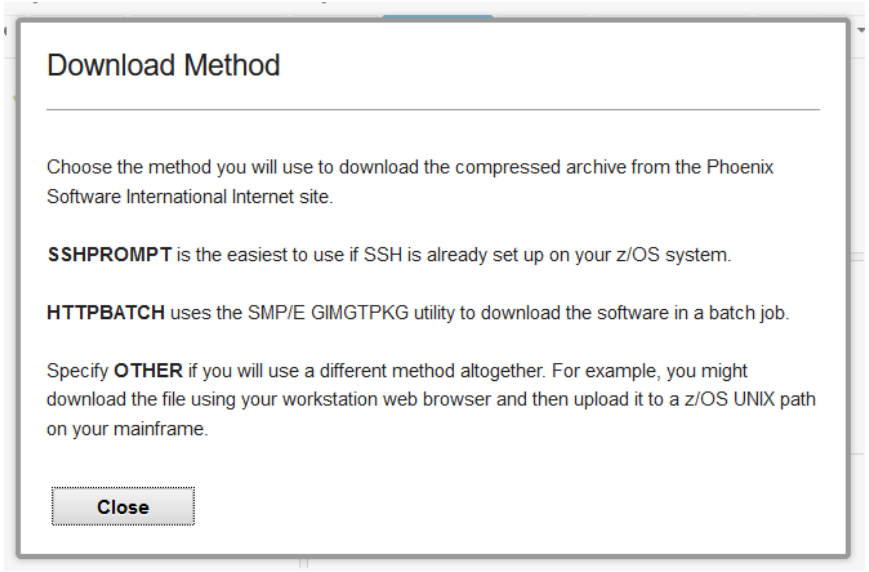



# Our Value Choice Variable with Help Pop-up

- This choice populates the workflow instance variable **downloadMethod** with the chosen string in the drop-down list. It can be used for step conditions, in templates (JCL, REXX execs, shell scripts), etc.

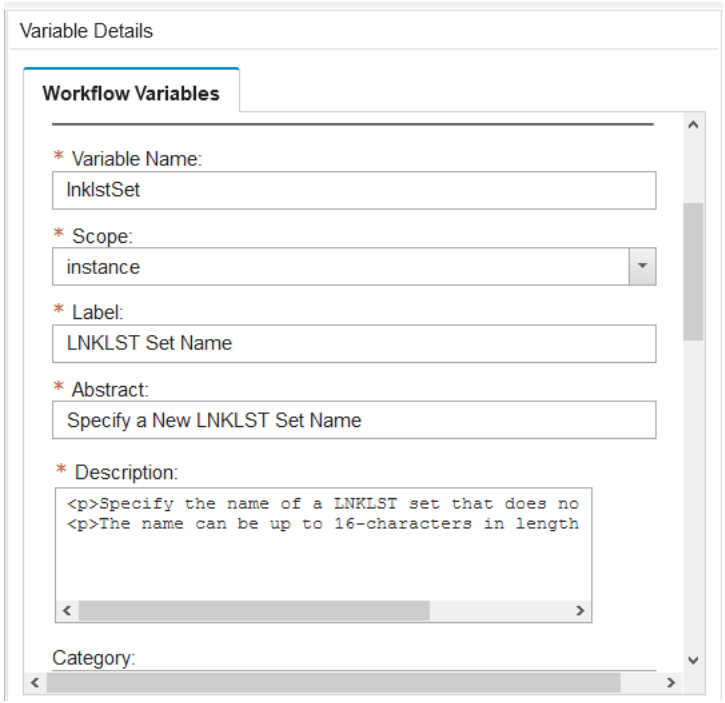


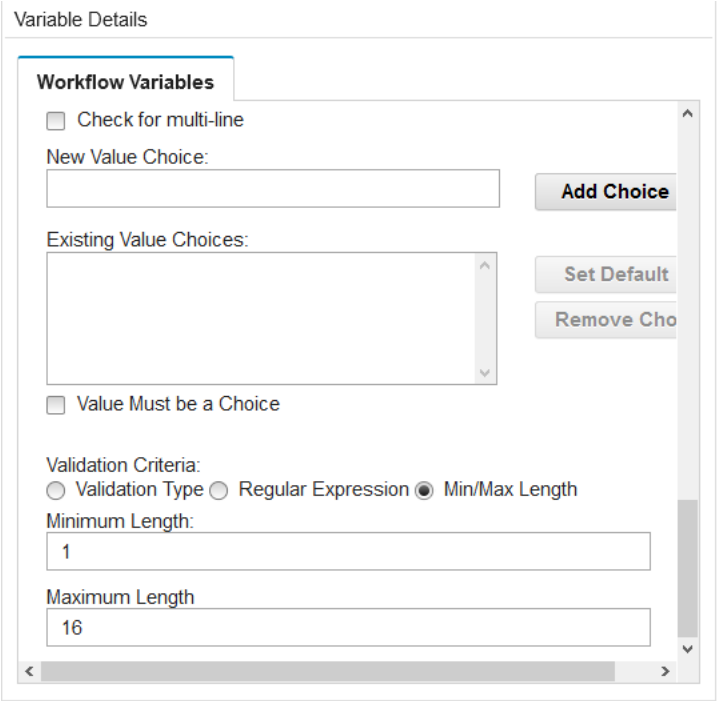

- Clicking on the blue information bubble produces this pop-up



# Defining a String Variable

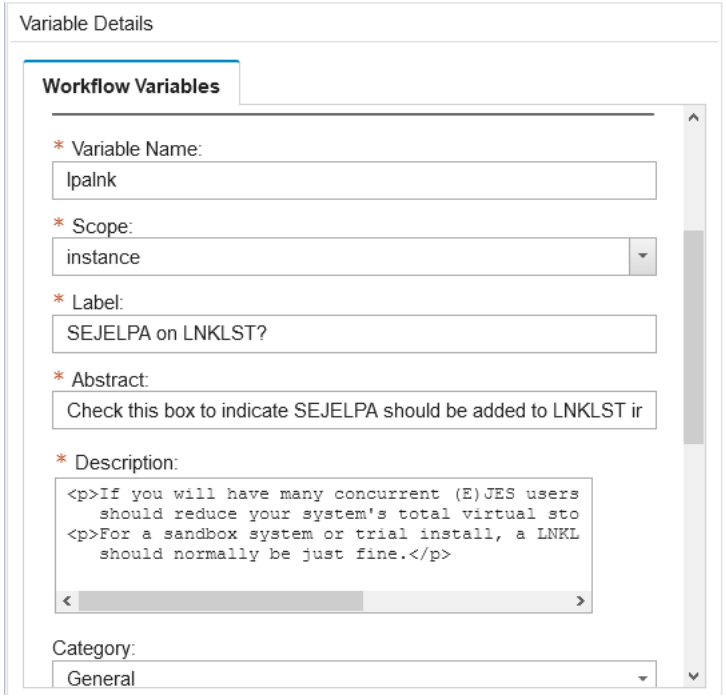
- Creates an ordinary entry field

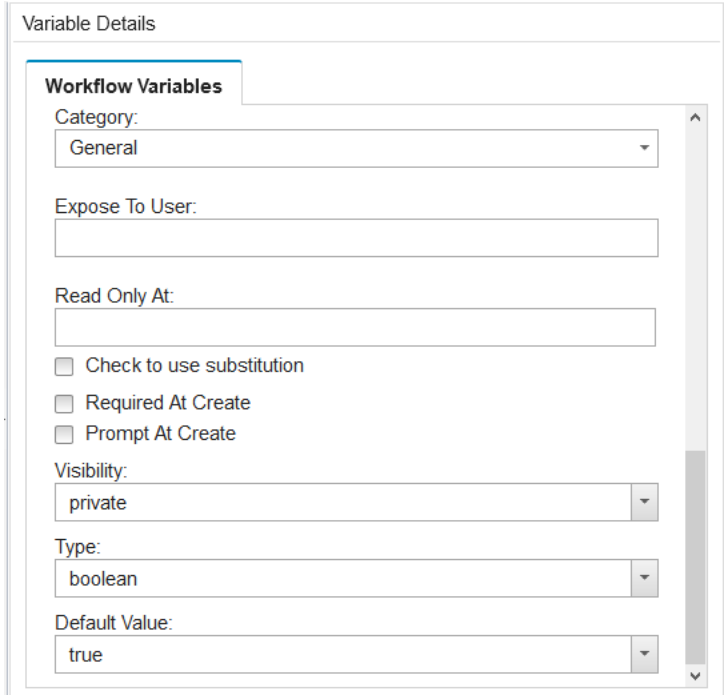


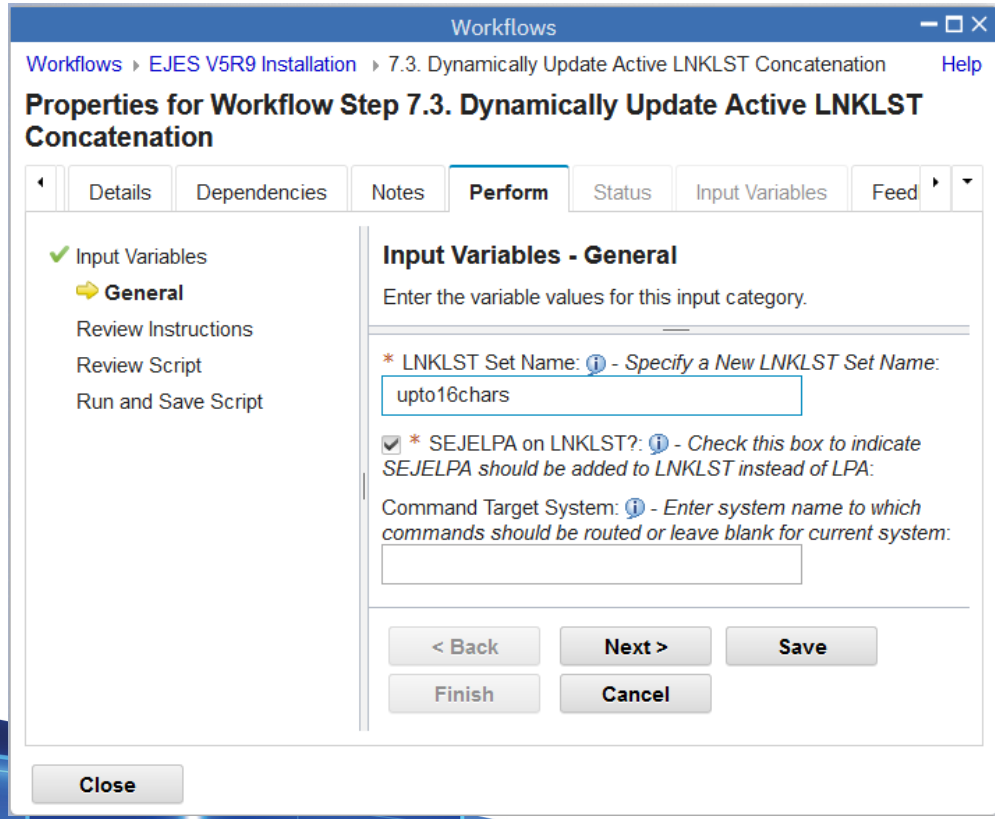
# Defining a Boolean Variable

- Creates a check-box (checked = TRUE)

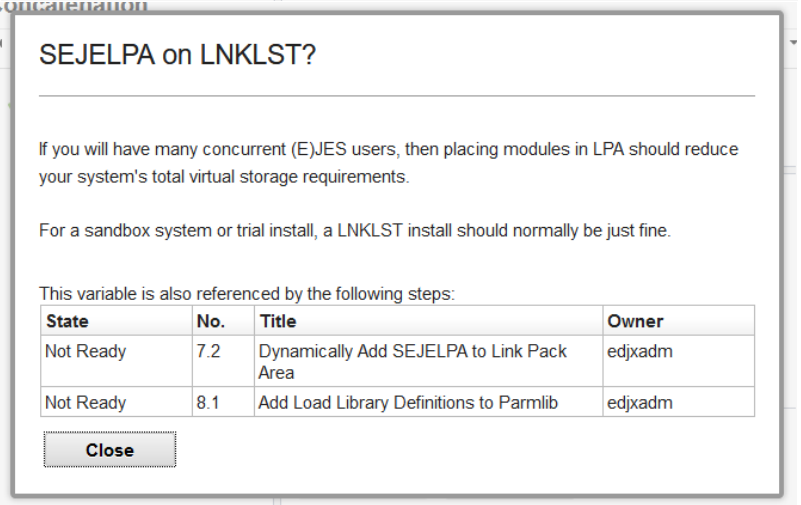


# Our String and Boolean Variables with Help Pop-ups

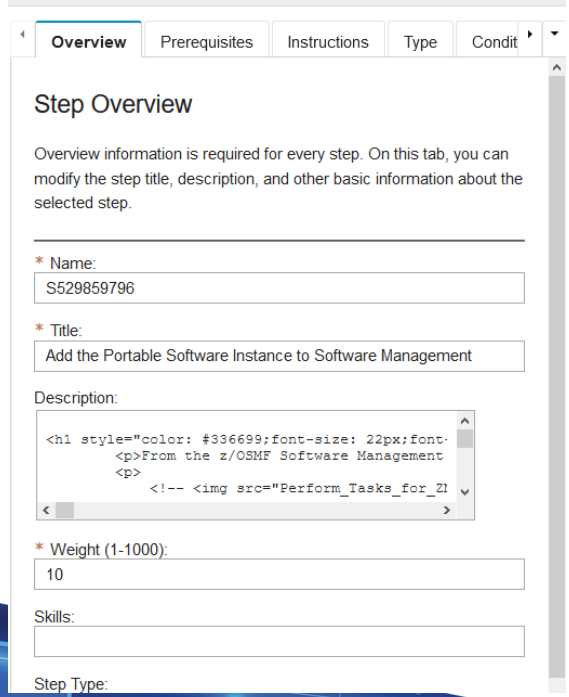


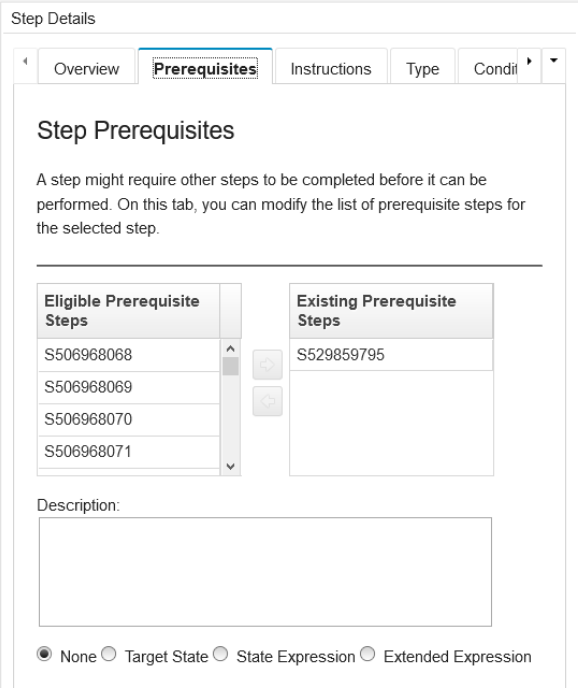


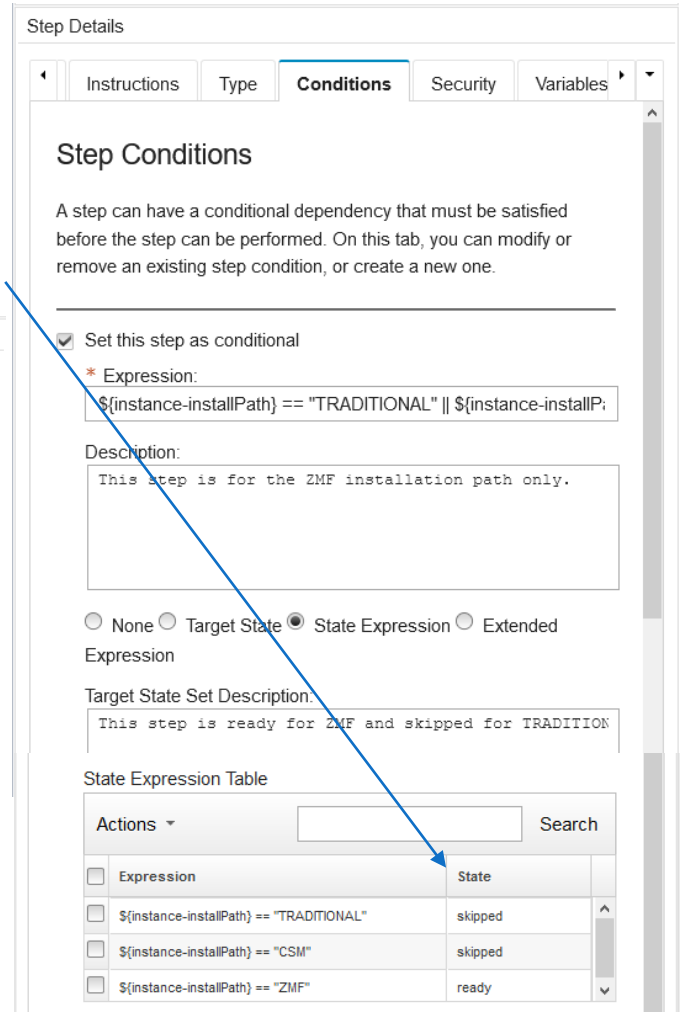


# Defining a Conditional Step

- You need conditions in addition to prerequisites
- In this case variable **installPath** controls the state



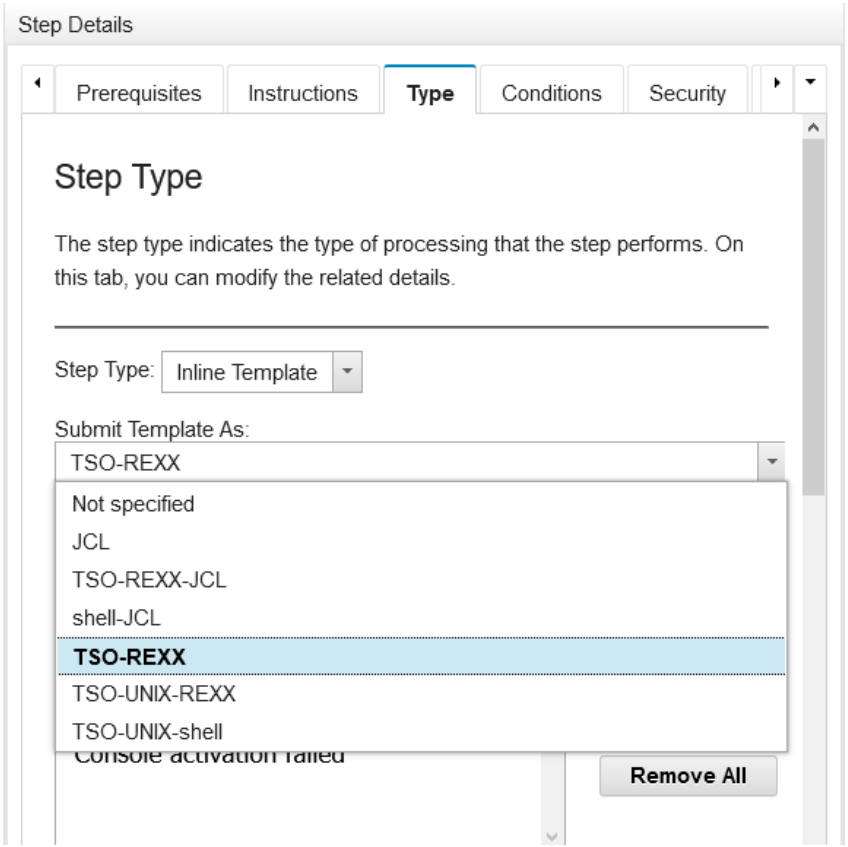



# Types of Steps

- A File or Inline Template lets you use an Apache Velocity Template alone or to run a batch job, REXX exec, or z/OS UNIX shell script.
  - A File Template is external to the workflow.
  - https://velocity.apache.org/engine/2.0/user-guide.html
- A REST API call allows you to use services that are defined and available in your instance of z/OSMF or outside servers whose URLs you know.
- Instructions Only is, as it's name suggests, a step that educates you about something or asks you to perform some action manually.
  - When you first convert a book into a workflow, all steps will be Instructions Only.
- A Calling Step is how you launch one workflow as a subroutine to another.

# Template Steps Run Batch, REXX, or Shell

- **Not specified** uses template functions only
- **JCL** is a batch job
- **TSO-REXX-JCL** is a batch wrapper around a TSO/E REXX exec
- **shell-JCL** is a batch wrapper around a z/OS UNIX shell script
- **TSO-REXX** is an immediate execution of a REXX exec under TSO/E
- **TSO-UNIX-REXX** is an immediate execution of a REXX exec running in the OMVS z/OS UNIX environment
- **TSO-UNIX-shell** is an immediate execution of a shell script running in the OMVS z/OS UNIX environment
- You can have inline or file templates
- The templates can have embedded workflow variables.
- You can save the results as a data set or z/OS UNIX file

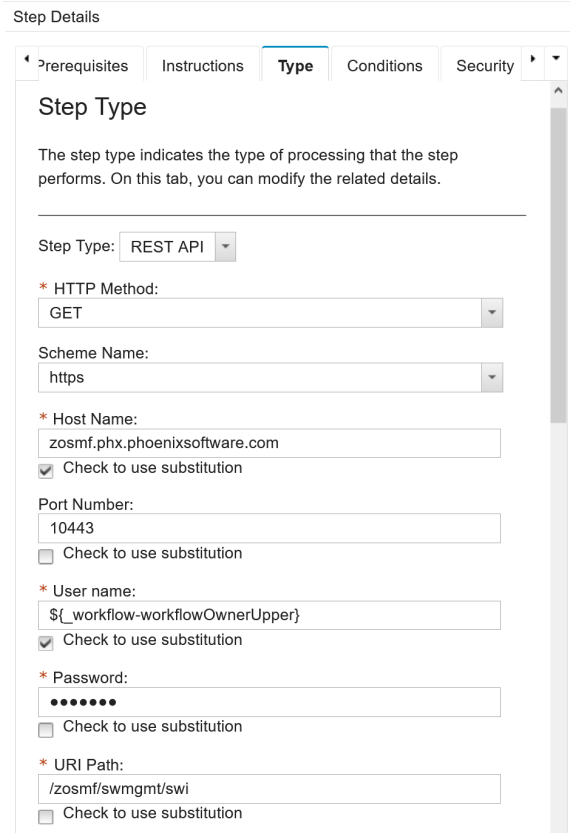


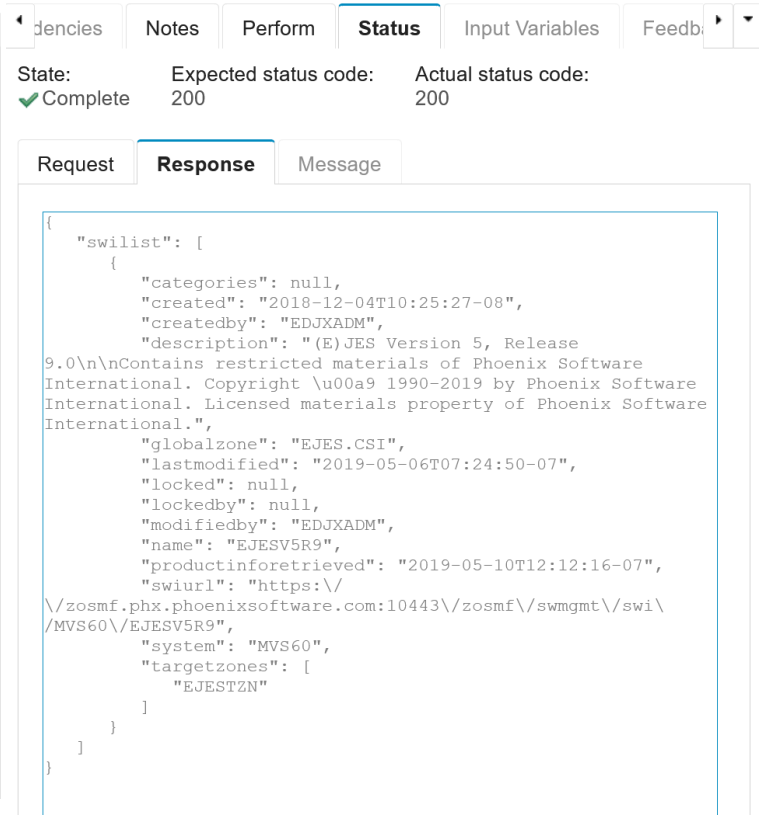
# Some Tips and Techniques for Scripts

- Scripts complete normally only when they produce the message defined as “Message for Program Success.”
- Scripts complete with failure if they produce any of the messages defined as “Message for Program Failure.”
- You can’t do ANYTHING in your script after you produce one of these messages. Workflow pulls the “rug” out from under you. So make sure message issuance is the very last thing the script does. (We put a “dummy” EXIT instruction after.)
- Scripts that produce none of the defined messages will time out and fail that way.
- Workflow variables can be prompted for and used in the same step.
- If you wish to use ISPF services, be sure to allocate ISPLOG to DUMMY or any action you take will cause prompting for log data set disposition at ISPF session termination causing your script to hang and time out.
  - Thanks to Marvin Knight of IBM for helping our team with this!
- If you have a z/OS background, stick to REXX. It’s easier to use **syscalls()** to get z/OS UNIX functionality in a REXX exec than it is to go the other way around.

# REST API Steps

- GET, PUT, POST and DELETE methods supported.
- The workflow editor makes required some fields that are documented as optional. ☹
- Your workflow can “map” the response and set workflow variables from the results.
- So far, no PSWI management capabilities ☹



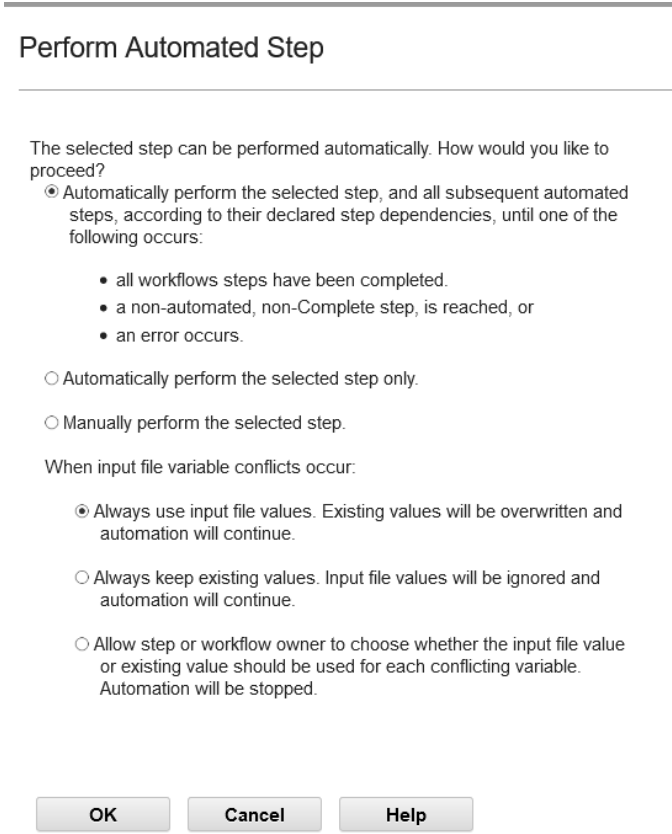
# Using Array Variables in a Template

- Templates can use workflow array variables
- In this simplistic case, we create a PDS member that contains the data set name and volume serial for every data set in the software instance.
- See SYS1.SAMPLIB(IZUDWFVR) for more examples of variable use.

```
<template>
<inlineTemplate substitution="true">
List of Data Sets Used:
##
## Loop For All Data Sets
##
#foreach($item in ${instance-izud-datasets})
DSNAME=${item.izud-dsname}) VOLUME=${item.izud-volumes[0]}
#end
</inlineTemplate>
<saveAsDataset substitution="true">
${instance-izud-createdby}.PARMLIB(DSNLIST)</saveAsDataset>
</template>
```

# Workflow Steps can be Automated

- Check the “Auto-Enable” box on General tab.
- When the user clicks “Perform” from the Steps page, they get a confirmation dialog.
- You must prompt (or provide discovered defaults) for every required variable needed for the steps you wish to automate together as a group.
- Reasons automation will stop:
  - Processing reaches an automated step for which one or more required variables are not satisfied.
  - Processing reaches an automated step that is somehow not eligible for automatic processing.
    - For example, if the step is Unassigned.
  - Processing is stopped through a user request.
- Press <Refresh> to see the results!



# General Rules of Thumb

- Organize your actionable steps as leaf steps under at least one parent level.
- In our opinion, you should put your primary step content under the **<description>** tag. That's the content users see when they select a step from Steps page. The text under the **<instructions>** tag isn't shown until after **Perform** is selected and is better suited to helping explain what to expect from the workflow than it is to explaining how to perform an activity on your z/OS system.
- Use existing high-quality content wherever possible.
- Use high-quality graphics to explain things. A picture is worth 1000 words!
- Keep abreast of APAR activity in this space.
  - Development is ongoing and usually satisfied through SPEs.
- Documentation is not as thorough as one might hope. Be prepared to experiment. Use the Velocity Template doc provided by Adobe!
- Collaborate with colleagues, join the Zorow community and ask and share!

# Objective

- Discussed how Phoenix Software International is applying the use of present-day tools with the mainframe such as:
  - **(E)JES, (E)JES Web and Eclipse** – A modern, lightweight browser-based system management tool for users who prefer not to work in a 3270 and for developers who want to leverage the Eclipse IDE for development
    - Providing modern interfaces for system management for those not comfortable with 3270
    - Leveraging modern development tools such as REST APIs and the Eclipse IDE
  - **Zowe** – Open source framework for the mainframe that provides solutions that allow development and operations teams to securely manage, control, script, and develop on the mainframe like any other cloud platform
    - What is Zowe?
    - How does Zowe work?
    - How can you participate in and leverage Zowe?
    - How are we participating in and leveraging Zowe?
  - **z/OSMF Workflow** – automation of routine procedures and tasks through the use of a modern interface and **Zorow** – An open source community dedicated to contributing and collaborating on z/OSMF Workflows
    - What is z/OSMF Workflow?
    - What is Zorow?
    - How can you participate in and leverage Zorow?
    - How are we participating in and leveraging Zorow?

# Want to learn more?

- Phoenix Products: https://www.phoenixsoftware.com/
- Eclipse: https://www.eclipse.org/
- Zowe: https://www.openmainframeproject.org/projects/zowe
- Zorow: https://www.openmainframeproject.org/projects/zorow

# THANK YOU