Closed Beta for Project Zoe

User's Guide

Edition notice

This edition applies to the Closed Beta of Project Zoe and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2018.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

© Copyright Rocket Software, Inc. or its affiliates 2018. All Rights Reserved.

Copyright © 2018 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

About this documentation

This documentation describes how to install, configure, and use Closed Beta for Project Zoe.

Who should read this documentation

This documentation is intended for system programmers who are responsible for installing Project Zoe, developers who want to use Project Zoe to improve z/OS user experience, and anyone who wants to know about Project Zoe.

To use this documentation, you must be familiar with the mainframe and z/OSMF configuration.

How to send your feedback on this documentation

We value your feedback. If you have comments about this documentation, you can open an issue in Github to request documentation to be updated, improved, or clarified by providing a comment.

To do this, complete the following steps.

1. Go to the Project Zoe issues repository in GitHub.
2. Select **Issues**.
3. Click **New issue**.
4. Provide a title and enter your comment on the documentation. Be sure to include the specific location of the text that you are commenting on, for example, the section heading.
5. Click **Submit new issue**.

Summary of changes

Learn about what is new, changed, and removed in Project Zoe.

Version 0.8.3 (June 2018)

What's changed

Zoe Brightside

The following plug-ins are no longer packaged with Zoe Brightside:

- Zoe Brightside Plug-in for CA Endevor® Software Change Manager
- Zoe Brightside Plug-in for CA File Master™ Plus

Version 0.8.2 (June 2018)

What's new

Zoe Brightside

Zoe Brightside is now built on a framework that lets you install plug-ins to extend the capabilities of the product. You can install the following plug-ins for Zoe Brightside:

- Zoe Brightside plug-in for CA Endevor® Software Change Manager
- Zoe Brightside plug-in for CA File Master™ Plus
- Zoe Brightside plug-in for IBM® Db2® Database

Learn more

The new framework also improves the stability and quality of Zoe Brightside.

Installation procedure

- zLUX and explorer server installation
  - Added RACF authorization for Zoe.
  - Zoe start and stop are now available as shell scripts that you can issue from the USS command line without using TSO LOGON and SDSF.
  - You can now start and stop zLUX server, explorer server, and zSS server together.
  - The JCL for the Zoe server started task is now *automatically* copied to a suitable PROCLIB.
  - The install script now writes a date-time-stamped log as it runs, which you can use for debugging or reference.

  Learn more
- Zoe Brightside installation

  You can now install Zoe Brightside using simplified and flexible installation process. You can use either of the following methods:

  - Install Zoe Brightside using the installation package that is contained on the Project Zoe Downloads repository.
  - Install Zoe Brightside using the Node.js Package Manager (npm).

  Important! Both of the installation methods require Internet access on client PCs.

Learn more

zLUX

- zLUX application plug-in definition and Configuration Dataservices
  - Information about the plug-in definition file and the zLUX application plug-in filesystem structure was added.

    Learn more
  - Information about Configuration Dataservices that enable you to set plug-in default behavior was added.

    Learn more
- zLUX terminal application plug-in initial configuration steps

  Steps to initially configure the zLUX terminal application plug-ins were added.

  Learn more

What's changed

Zoe Brightside

- **CA Brightside** was renamed to **Zoe Brightside**.

What's removed:

Zoe Brightside

Experimental command groups have been removed from Zoe Brightside.

Version 0.8.1 (May 2018)

What's new

VT Terminal

zLUX now provides a VT Terminal application plug-in that provides a connection to USS and UNIX. Learn more

What's changed

Product naming

Project Giza is renamed to Project Zoe. Atlas is renamed to explorer server.

Installation procedure

Project Zoe now provides an enhanced and simplified installation process to improve your installation experience. Learn more

Known issues

When you initially open the MVD, a security message alerts you that you are attempting to open a site that has an invalid HTTPS certificate. Other apps within the MVD may also encounter this message. To prevent this message, add the URLs that you see to your list of trusted sites.

NOTE: If you clear the browser cache, you must add the URL to your trusted sites again.

zLUX APIs exist but are under development. Features might be reorganized if it simplifies and clarifies the API, and features might be added if applications can benefit from them.

Chapter 1. Project Zoe overview

Project Zoe offers modern interfaces to interact with z/OS and allows you to work with z/OS in a way that is identical to what you experience on cloud platforms today. You can use these interfaces as delivered or through programmable extensions that are created by clients or third-party vendors.

Project Zoe consists of the following three main components. Each component provides a different interface.

- zLUX, which contains a browser-based user interface (UI) that provides a full screen interactive experience. It includes all interactions that exist in 3270 terminals and web interfaces like z/OSMF.
- Explorer server, which provides a range of APIs for the management of jobs, data sets, z/OS UNIX System Services files, and persistent data.
- Zoe Brightside, which provides a command line interface that lets you interact with the mainframe remotely and use common tools such as Integrated Development Environments (IDEs), shell commands, bash scripts, and build tools for mainframe development.

For details of each component, see the corresponding section.

zLUX

zLUX is a product that modernizes and simplifies working on the mainframe. With zLUX, you can create applications to suit your specific needs. zLUX contains a web UI that has the following features:

- The web UI works with the underlying REST APIs for data, jobs, and subsystem, but presents the information in a full screen mode as compared to the command line interface.
- The web UI makes use of the leading-edge web presentation technology and is also extensible through web UI plug-ins to capture and present any variety of information.
- The web UI includes common z/OS developer or system programmer tasks such as an editor for common text-based files like REXX or JCL along with general purpose data set actions for both Unix System Services (USS) and Partitioned Data Sets (PDS) plus Job Entry System (JES) logs.

zLUX consists of the following components:

- **Mainframe Virtual Desktop (MVD)**

  The desktop, accessed through a browser.
- **Zoe Node Server**

  The Node.js server plus the Express.js as a webservices framework, and the proxy applications that communicate with the z/OS services and components.
- **zLUX Secure Services address space**

  A server that provides secure REST services to support the Zoe Node Server.
- **Application plug-ins**

  Several application-type plug-ins are provided. For more information, see Using zLUX application plug-ins.

Explorer server

The explorer server is a z/OS® RESTful web service and deployment architecture for z/OS microservices. The server is implemented as a Liberty Profile web application that uses z/OSMF services to provide a range of APIs for the management of jobs, data sets, z/OS UNIX™ System Services (USS) files, and persistent data.

These APIs have the following features:

- These APIs are described by the Open API Specification allowing them to be incorporated to any standard-based REST API developer tool or API management process.
- These APIs can be exploited by off-platform applications with proper security controls.

Any client application that calls RESTful APIs directly can use the explorer server.

As a deployment architecture, the explorer server accommodates the installation of other z/Tool microservices into its Liberty instance. These microservices can be used by explorer server APIs and client applications.

Zoe Brightside

Zoe Brightside is a command-line interface that lets application developers interact with the mainframe in a familiar format. Zoe Brightside helps to increase overall productivity, reduce the learning curve for developing mainframe applications, and exploit the ease-of-use of off-platform tools. Zoe Brightside lets application developers use common tools such as Integrated Development Environments (IDEs), shell commands, bash scripts, and build tools for mainframe development. It provides a set of utilities and services for application developers that want to become efficient in supporting and building z/OS applications quickly.

Zoe Brightside provides the following benefits:

- Enables and encourages developers with limited z/OS expertise to build, modify, and debug z/OS applications.
- Fosters the development of new and innovative tools from a PC that can interact with z/OS.
- Ensure that business critical applications running on z/OS can be maintained and supported by existing and generally available software development resources.
- Provides a more streamlined way to build software that integrates with z/OS.

The following sections explain the key features and details for Zoe Brightside:

Note: For information about prerequisites, software requirements, installing and upgrading Zoe Brightside, see Installing Project Zoe.

Zoe Brightside capabilities

With Zoe Brightside, you can interact with z/OS remotely in the following ways:

- **Interact with mainframe files:** Create, edit, download, and upload mainframe files (data sets) directly from Zoe Brightside.
- **Submit jobs:** Submit JCL from data sets or local storage, monitor the status, and view and download the output automatically.
- **Issue TSO and z/OS console commands:** Issue TSO and console commands to the mainframe directly from Zoe Brightside.
- **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.

For more information about the available functionality in Zoe Brightside, see Zoe Brightside Command Groups.

Extending Zoe Brightside

You can install plug-ins to extend the capabilities of Zoe Brightside. Plug-ins add functionality to the product in the form of new command groups, actions, objects, and options. The following plug-ins are available:

- IBM Db2 Database plug-in

For more information, see Installing Plug-ins.

Reporting defects and issues

To report problems that you might encounter while using the product, enter issues in the Brightside GitHub repository.

Third-Party software agreements

Zoe Brightside uses the following third-party software:

Third-party Software	Version	File name
chalk	2.3.0	Legal_Doc_00002285_56.pdf
cli-table2	0.2.0	Legal_Doc_00002310_5.pdf
dataobject-parser	1.2.1	Legal_Doc_00002310_36.pdf
find-up	2.1.0	Legal_Doc_00002310_33.pdf
glob	7.1.1	Legal_Doc_00001713_45.pdf
js-yaml	3.9.0	Legal_Doc_00002310_16.pdf
jsonfile	4.0.0	Legal_Doc_00002310_40.pdf
jsonschema	1.1.1	Legal_Doc_00002310_17.pdf
levenshtein	1.0.5	See UNLICENSE
log4js	2.5.3	Legal_Doc_00002310_37.pdf
merge-objects	1.0.5	Legal_Doc_00002310_34.pdf
moment	2.20.1	Legal_Doc_00002285_25.pdf
mustache	2.3.0	Legal_Doc_mustache.pdf
node.js	6.11.1	Legal_Doc_nodejs.pdf
node-ibm_db	2.3.1	Legal_Doc_00002310_38.pdf
node-mkdirp	0.5.1	Legal_Doc_00002310_35.pdf
node-progress	2.0.0	Legal_Doc_00002310_7.pdf
prettyjson	1.2.1	Legal_Doc_00002310_22.pdf
rimraf	2.6.1	Legal_Doc_00002310_8.pdf
stack-trace	0.0.10	Legal_Doc_00002310_10.pdf
string-width	2.1.1	Legal_Doc_00002310_39.pdf
wrap-ansi	3.0.1	Legal_Doc_00002310_12.pdf
yamljs	0.3.0	Legal_Doc_00002310_13.pdf
yargs	8.0.2	Legal_Doc_00002310_1.pdf

Note: All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

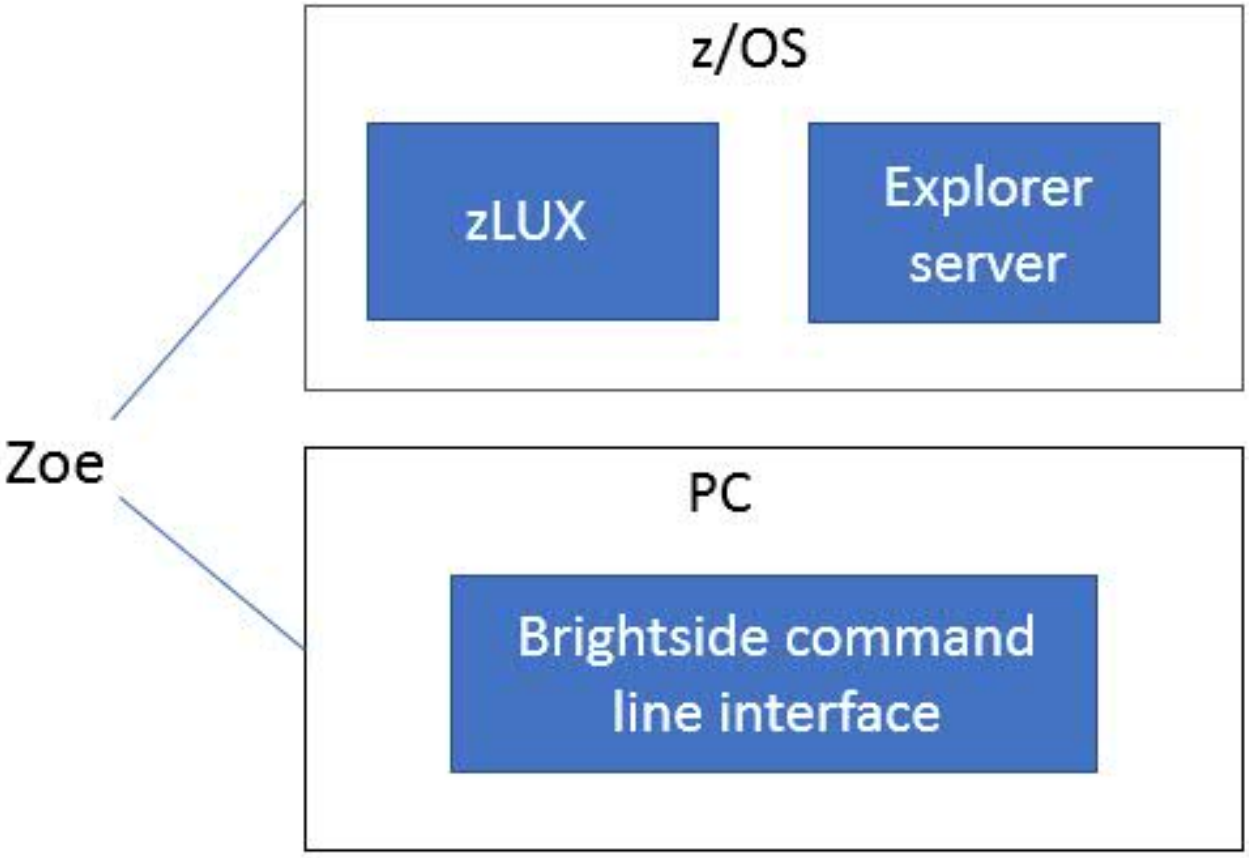
To read each complete license, navigate to the GitHub repository and download the file named Zoe_Brightside_TPSRs.zip. The .zip file contains the licenses for all of the third-party components that Zoe Brightside uses.

More Information:

- System requirements for Zoe Brightside
- Installing Zoe Brightside
- Creating a profile
- Testing connection to z/OSMF

Chapter 2. Installing Project Zoe

Project Zoe consists of three main components: zLUX, the explorer server, and Zoe Brightside. You install zLUX and the explorer server on z/OS and install Zoe Brightside on PC. The installations on z/OS and on PC are independent.



To get started with installing Project Zoe, review the Installation roadmap topic.

Installation roadmap

Installing Project Zoe involves several steps that you must complete in the appropriate sequence. Review the installation roadmap that presents the task-flow for preparing your environment and installing and configuring Project Zoe before you begin the installation process.

Tasks	Description
1. Prepare your environment to meet the installation requirements.	See System requirements.
2. Obtain the Zoe installation files.	The Zoe installation files are released in a PAX file format. The PAX file contains the runtimes and the scripts to install and launch the z/OS runtime and the runtime for Zoe Brightside. For how to download and prepare the file to install the Zoe runtime, see Obtaining the installation files.

Tasks	Description
3. Allocate enough space for the installation.	For successful installation of Project Zoe, your PC must contain the required space. The installation process requires approximately 1 GB of available space. Once installed, zLUX requires approximately 50 MB of space before configuration, explorer server requires approximately 200 MB, and Zoe Brightside requires approximately 25 MB.
4. Install components of Project Zoe.	To install zLUX and explorer server on z/OS, see Installing the Zoe runtime on z/OS. To install Zoe Brightside on PC, see Installing Zoe Brightside.
5. Verify that Project Zoe is installed correctly.	To verify that zLUX and explorer server are installed correctly, see Verifying installation. To verify that Zoe Brightside is installed correctly, see Testing connection to z/OSMF.
6. Optional: Troubleshoot problems that occurred during installation.	See Troubleshooting installing the Zoe runtime and Troubleshooting installing Zoe Brightside.

To uninstall Project Zoe, see Uninstalling Project Zoe.

System requirements

Before installing Project Zoe, ensure that your environment meets all of the prerequisites.

1. Ensure that IBM z/OS Management Facility (z/OSMF) is installed and configured correctly. z/OSMF is a prerequisite for the Project Zoe microservice that must be installed and running before you use Project Zoe. For details, see z/OSMF configuration.
2. Review component specific requirements.
   - System requirements for zLUX
   - System requirements for explorer server
   - System requirements for Zoe Brightside

z/OSMF configuration

IBM z/OS Management Facility (z/OSMF) is a prerequisite for the Project Zoe microservice that must be installed and running before you use Project Zoe. This article consists of the following information:

- z/OSMF requirements for Project Zoe
- Configuring z/OSMF
- Verifying your z/OSMF configuration

Important! The IBM z/OS Management Facility guides on the IBM Knowledge Center are your primary source of information about how to install and configure z/OSMF. In the following topics, we provide procedures and tips for the configuration required for Project Zoe. We recommend that you open the following IBM documentation in a separate browser tab (The z/OSMF process differs depending on whether you have z/OS v2.2 or v2.3):

IBM z/OSMF documentation:

- IBM z/OS Management Facility Help
- IBM z/OS Management Facility Configuration Guide

z/OSMF requirements for Project Zoe

Meet the following requirements before you use Project Zoe:

z/OS requirements

Ensure that your z/OS system meets the following requirements for z/OSMF to function properly with Project Zoe:

- **AXR (System REXX)** - The AXR (System REXX) component lets z/OS perform Incident Log tasks. It also lets REXX execs execute outside of conventional TSO and batch environments.
- **CEA (Communications Enabled Applications) Server** - CEA server is a co-requisite for the CIM server. The CEA server lets z/OSMF deliver z/OS events to C-language clients.
  - Start the CEA server before you start the z/OSMF (the IZU* started tasks).
  - Set up CEA server in Full Function Mode and assign the TRUSTED attribute to the CEA started task.
  - For more information, see Customizing for CEA on the IBM Knowledge Center.
- **CIM (Common Information Model) Server** - z/OSMF requires the CIM server to perform capacity provisioning and workload management tasks. Start the CIM server before you start z/OSMF (the IZU* started tasks).
  - For more information, see Reviewing your CIM server setup on the IBM Knowledge Center.
- **Console Command** - The CONSOLE and CONSPROF commands must exist in the authorized command table.
- **IBM z/OS Provisioning Toolkit** - The IBM® z/OS® Provisioning Toolkit is a command line utility that lets you provision z/OS development environments. The product is required if you want to provision CICS or Db2 environments with Zoe Brightside.
- **Java version** - IBM® 64-bit SDK for z/OS®, Java Technology Edition V7.1 or higher is required.
  - For more information, see Software prerequisites for z/OSMF on the IBM Knowledge Center.
- **Maximum region size** - To prevent exceeds maximum region size errors, ensure that you have a TSO maximum region size of at least 65536 KB for the z/OS system.
- **User IDs** - User IDs require a TSO segment (access) and an OMVS segment. During workflow processing and REST API requests, z/OSMF may start one or more TSO address spaces under the following job names:
  - userid
  - substr(userid, 1, 6)||CN (Console)

For more information, refer to the IBM z/OSMF documentation.

z/OSMF plug-in requirements

Ensure that the following IBM z/OSMF plug-ins are installed and configured:

- **(Optional) Cloud Portal** - The Cloud Portal plug-in lets you make software services available to marketplace consumers and it adds the Marketplace and Marketplace Administration tasks to the z/OSMF navigation tree.
- **Configuration Assistant** - The Configuration Assistant plug-in lets z/OSMF configure TCP/IP policy-based networking functions.
- **ISPF** - The ISPF plug-in lets z/OSMF access traditional ISPF applications.
- **Workload Management** - The Workload Management plug-in lets z/OSMF operate and manage workload management service definitions and policies.

For more information about configuring each z/OSMF plug-in and the related security, refer to the IBM z/OSMF documentation for each plug-in.

REST services requirements

Ensure that the following REST services are configured and available when you run Project Zoe:

- **Cloud provisioning services** - Cloud provisioning for development environments. Cloud provisioning services are required for the Zoe Brightside cics and db2 command groups to function properly. Endpoints begin with `/zosmf/provisioning/`

- **TSO/E address space services** - Required to issue TSO commands in Zoe Brightside. Endpoints begin with `/zosmf/tsoApp`
- **z/OS console** - Required to issue console commands in Zoe Brightside. Endpoints begin with `/zosmf/restconsoles/`
- **z/OS data set and file interface** - Required to work with mainframe data sets and USS files in Zoe Brightside. Endpoints begin with `/zosmf/restfiles/`
- **z/OS jobs interface** - Required to use the zos-jobs command group in Zoe Brightside. Endpoints begin with `/zosmf/restjobs/`
- **z/OSMF workflow services** - Cloud provisioning for development environments. Cloud provisioning services are required for the Zoe Brightside cics and db2 command groups to function properly. Endpoints begin with `/zosmf/workflow/`

Additionally, Project Zoe uses z/OSMF configuration by using symbolic links to the z/OSMF `bootstrap.properties`, `jvm.security.override.properties`, and the `ltpa.keys` files. Specifically, Project Zoe reuses z/OSMF's SAF, SSL, and LTPA configuration; therefore, these configurations must be valid and complete to operate Project Zoe successfully.

For more information, refer to the IBM z/OSMF documentation for each REST service.

Configuring z/OSMF

1. Verify your system requirements.

   For z/OS v2.2 or later, use any of the following options to determine which version is installed:

   - If you have access to the console, for example, in SDSF, issue the command:

     ```
     /D IPLINFO
     ```

     Part of the output contains the release, for example,

     ```
     RELEASE z/OS 02.02.00.
     ```

   - If you don't have access to the console, use SDSF and select the menu option **MAS**. Two columns of the output show the SysName (LPAR name) and the z/OS version, for example:

     ```
     SysName  Version

     S001     z/OS 2.2
     ```

     Identify the z/OS Version for the LPAR on which you are going to use z/OSMF.

   - If you don't have access to SDSF, use ISPF option **7.3** (Dialog Test Variables) and scroll down to the variable Z0S390RL, for example,

     ```
     ZOS390RL S N z/OS   02.02.00
     ```

   - On the ISPF Primary Option Menu, the last entry is the ISPF Release, which corresponds to the z/OS version as follows:

     ```
     Release . : ISPF 7.1    --> z/OS v2.1
     Release . : ISPF 7.2    --> z/OS v2.2
     Release . : ISPF 7.3    --> z/OS v2.3
     ```

2. Configure z/OSMF.

   For z/OS V2.2 users, take the following steps to configure z/OSMF:

   z/OSMF is a base element of z/OS v2.2 and v2.3, so it should already be installed. However, it is not guaranteed to be configured and running on every z/OS V2.2 and V2.3 system.

   Configuring an instance of z/OSMF is done by running the IBM-supplied jobs IZUSEC and IZUMKFS, and then starting the z/OSMF server in the following order:

a. Security setup (the IZUSEC job)
b. Configuration (the IZUMKFS job)
c. Server initialization (the START command)

For z/OS V2.3 users, the base element z/OSMF is started by default at system IPL. This means that z/OSMF is available for use as soon as the system is up. If you prefer not to have z/OSMF started automatically, you can disable the autostart function by checking for START commands for the z/OSMF started procedures in the COMMNDxx parmlib member.

The z/OS Operator Consoles task is new in this release. Applications that depend on access to the operator console such as Brightside's RestConsoles API require version 2.3.

3. Verify other requirements.

- Perform any maintenance that is required by Node.js.

  Connect to your target z/OS system, for example, with ssh to port 22 to get a USS command window. Issue the command:

  ```
  node -v
  ```

  or

  ```
  node --version
  ```

  The response should be:

  ```
  v6.11.2
  ```

  or later. If the version displayed is not right, set and/or download the right version of node by using npm and nvm. You might want to refer to the following links for tutorials on npm and nvm:

  https://www.sitepoint.com/beginners-guide-node-package-manager/

  https://www.sitepoint.com/quick-tip-multiple-versions-node-nvm/

  If you don't have node installed, go to https://nodejs.org/en/download/package-manager/, select your operating system, and follow the instructions to install node. Ensure that you download only official versions of these products. For z/OSMF, you need only one version of node.js. You can find the complete procedure for installing Node.js at https://developer.ibm.com/node/sdk/ztp/.

  When completed, set the `NODE_HOME` environment variable to the directory where your Node.js is installed, for example,

  ```
  NODE_HOME=/proj/mvd/node/installs/node-v6.10.3-os390-s390x
  ```

- Issue the following command to check the Java installation.

  ```
  java -version
  ```

  The response should be

  ```
  java version "1.8.0"
  ```

  or later. If the version displayed is not right, set the `JAVA_HOME` variable to point to the preferred java version. If the preferred java version is not installed, install it.

- Verify that you have 400 MB of free HFS file space for the installation. You can use the `df` command to check the available space in your chosen file system, for example,

  ```
  df -k /usr/lpp/zosmf
  ```

The output should be as follows:

```
Mounted on       Filesystem               Avail/Total

/Z22C/usr/lpp/zosmf  (ZFS.S001.Z22C.ZOSMF)      26711/535680
```

From the output above, you can see that only 26.711 MB is available (because z/OSMF is already installed), but the total in that file system is 535.68 MB, so enough space was available when the file system was created.

- Verify your browser support. Confirm that the machine from which you plan to run the Zoe desktop runs one of the supported browsers:
  - Chrome version 54 or later
  - Firefox version 44 or later
  - Microsoft Edge

Note: Microsoft Internet Explorer is not yet supported at any version.

4. After configuring z/OSMF, verify the following items to ensure z/OSMF is ready for Project Zoe.

   Check that the z/OSMF server and angel processes are running. From SDSF on z/OS, use the DA command or issue the following command on the command input line:

   ```
   /D A,IZU*
   ```

   If you don't see jobs like IZUANG1 and IZUSVR1 active, start the angel process with the following command:

   ```
   /S IZUANG1
   ```

   When you see the message **CWWKB0056I INITIALIZATION COMPLETE FOR ANGEL**, start the server with the following command:

   ```
   /S IZUSVR1
   ```

   The server might take a couple of minutes to fully initialize. The z/OSMF server is available when the following message **CWWKF0011I: The server zosmfServer is ready to run a smarter planet.** is displayed.

   You can test z/OSMF with your browser by first finding the startup messages in the SDSF log of the z/OSMF server by using the following find command:

   ```
   f IZUG349I
   ```

   You should see these lines:

   ```
   IZUG349I: The z/OSMF STANDALONE Server home page can be accessed at
    https://mvs.hursley.ibm.com:443/zosmf after the z/OSMF server is started
    on your system.
   ```

   From the lines above, the port number is 443. You will need this port number later.

   Point your browser at the nominated z/OSMF STANDALONE Server home page and you should see its Welcome Page where you can log in.

Verifying your z/OSMF configuration

To verify that IBM z/OSMF REST services are configured correctly in your environment, type the REST endpoint into your browser. For example: https://mvs.ibm.com:443/zosmf/restjobs/jobs

Notes: - Browsing z/OSMF endpoints requests your user ID and password for defaultRealm; these are your TSO user credentials.

- Your browser should return you a status code 200 with a list of all jobs on your z/OS system. The list is in raw JSON format.

Optional method for verifying z/OSMF configuration with Zoe Brightside

To verify that IBM z/OSMF is configured correctly, follow these steps to create and validate a profile in Zoe Brightside:

1. Meet the prerequisites for Zoe Brightside.
2. Installing Zoe Brightside.
3. Create a zosmf profile in Zoe Brightside. Issue the `bright help explain profiles` command to learn more about creating profiles in Zoe Brightside. See How to display Zoe Brightside help for more information.
4. Validate your profile.
5. Use the profile validation report to identify and correct errors with your z/OSMF configuration. If you receive a perfect score on the validation report, Project Zoe can communicate with z/OSMF properly.

Note: Before your run the profile validation, check that JES2 is accepting jobs with CLASS=C by issuing the following command in SDSF:

```
/$D I
```

You will see responses like the following in SYSLOG:

```
$HASP892 INIT(3)   STATUS=ACTIVE,CLASS=AB,...
```

If none of the initiators has **C** in its CLASS list, add **C** to the list of any initiator. For example, initiator 3 as shown above, with the following command:

```
/$T I3,CL=ABC
```

System requirements for zLUX

Your system must meet the software requirements for zLUX.

- z/OS Version 2.2 or later.
- Whatever maintenance is required by the Node.js and Java installations.
- 833 MB of HFS file space for the zLUX installation.

To verify that the most recent version of Java Version 8 is installed on z/OS, issue the following command:

```
  java -version
```

Confirm that the machine from which you plan to run zLUX runs one of the following supported browsers:

- Chrome 54 or later
- Firefox 44 or later
- Safari 11 or later
- Microsoft Edge

Note: Microsoft Internet Explorer is not yet supported at any version level.

To build zLUX applications, npm 5.4 or later is required. To update npm, issue the following command:

```
  npm install -g npm
```

Confirming that Node.js is installed

Node.js Version 6.11.2 or later must be on the z/OS host where you will install the Zoe Node Server.

1. If Node.js is not installed on z/OS, follow the procedure for doing so: https://developer.ibm.com/node/sdk/ztp.
2. Set the *NODE_HOME* environment variable to the directory where Node.js is installed. For example, (`NODE\_HOME=/proj/mvd/node/installs/node-v6.11.2-os390-s390x`).

Configuring ports for the zLUX terminal application plug-ins

Before you install Zoe, configure the following ports for the VT Terminal and TN3270 mainframe terminal application plug-ins.

1. Locate the `/install/zoe-install.yaml` file.
2. Specify the SSH port for the VT terminal application plug-in. The default is 22.
3. Specify the Telnet port for the TN3270 mainframe terminal application plug-in. The default is 23.

```
# Ports for the TN3270 and the VT terminal to connect to
terminals:
  sshPort=22
  telnetPort=23
```

System requirements for explorer server

- z/OS® Version 2.2 or later.
- 64-bit Java™ 8 JRE or later.
- IBM® z/OS Management Facility (z/OSMF) must be installed and running. See z/OSMF configuration.
- (Optional) The System Display and Search Facility (SDSF) of z/OS must be installed to enable real-time access to SYSLOG.

Pre-installation checklist

The following information is required during the installation process. Make the decisions before you install the explorer server.

- The HFS directory where you install the explorer server, for example, `/var/atlas`.
- The HFS directory path that contains a 64-bit Java™ 8 JRE.
- The z/OSMF installation directory `/lib` that contains `derby.jar`, for example, `/usr/lpp/zosmf/lib`.
- The z/OSMF configuration user directory path that contains the z/OSMF `/bootstrap.properties`, `/jvm.security.override.properties`, and `/resources/security/ltpa.keys` files.
- The explorer server HTTP and HTTPS port numbers. By default, they are 7080 and 7443.
- The user ID that runs the Liberty explorer server started task.

  Tip: Use the same user ID that runs the z/OSMF IZUSVR1 task, or a user ID with equivalent authorizations.
- (Optional) The SDSF Java installation directory, for example, `/usr/lpp/sdsf/java`.

System requirements for Zoe Brightside

Review and meet the following prerequisites before you install and use Zoe Brightside:

Supported platforms

You can use Zoe Brightside with the following platforms:

- **Local workstations:**

  Important!

  – Zoe Brightside is not officially supported on Mac computers. However, Zoe Brightside *might* run successfully on some Mac computers.

– Oracle Linux 6 is not supported.

You can install Zoe Brightside on any Windows or Linux operating system. For more information about known issues and workarounds, see Troubleshooting installing Zoe Brightside.

- **Mainframe systems:** Zoe Brightside was designed and tested to integrate with IBM z/OS Management Facility (z/OSMF) running on IBM z/OS version 2.2 or higher.

  Important! Before you can use Zoe Brightside to interact with the mainframe, systems programmers must install and configure IBM z/OSMF in your environment. The IBM z/OS Management Facility guide on the IBM Knowledge Center is the primary source of information about how systems programmers can install and configure z/OSMF. We provide supplemental information about Zoe Brightside-specific tips or requirements to which systems programmers can refer. For more information, see Prerequisites for z/OSMF configuration.

Free disk space

Zoe Brightside requires approximately **100 MB** of free disk space. The actual quantity of free disk space consumed might vary depending on the operating system on which you install Zoe Brightside.

Prerequisite software

The following software is required before you can install and use the product on your PC:

Note: We do not maintain the prerequisite software that Zoe Brightside requires. You are responsible for updating Node.js and other prerequisites on your PC. We recommend that you update Node.js regularly to the latest Long Term Support (LTS) version.

Windows operating systems

Windows operating systems require the following software:

- **Node.js v8.0 or higher:** Click here to Download Node.js

Note: Npm is included with the Node.js installation.

- **Node Package Manager (npm) v5.0 or higher**
- **Python v2.7** See the C++ Compiler prerequisite. The command that installs C++ Compiler also installs Python on Windows.
- **C++ Compiler (gcc 4.8.1 or later)** From an administrator command prompt, issue the following command: `npm install --global --production --add-python-to-path windows-build-tools`

Mac operating systems

Mac operating systems require the following software:

- **Node.js v8.0 or higher:** Click here to Download Node.js

Note: Npm is included with the Node.js installation.

- **Node Package Manager (npm) v5.0 or higher**

Tip: If you are installing Node.js a macOS operating system, we recommend that you install `nodejs` and `nodejs-legacy` using the instructions on the Nodejs website (using package manager). For example, you can install `nodejs-legacy` using the command `sudo apt install nodejs-legacy`. With `nodejs-legacy`, you can issue node commands rather than typing `nodejs`.

- **Python v2.7** Click here to Download Python 2.7
- **C ++ Compiler (gcc 4.8.1 or later)** The gcc compiler is included with MacOS. To confirm that you have the compiler, enter the command `gcc –help`. If you do not have the compiler installed, you can acquire gcc through a Google search.

Linux operating systems

Linux operating systems require the following software:

- **Node.js v8.0 or higher:** Click here to Download Node.js

Note: Npm is included with the Node.js installation.

- **Node Package Manager (npm) v5.0 or higher**

Tip: If you are installing Node.js a Linux operating system, we recommend that you install `nodejs` and `nodejs-legacy` using the instructions on the Nodejs website (using package manager). For example, you can install `nodejs-legacy` using the command `sudo apt install nodejs-legacy`. With `nodejs-legacy`, you can issue node commands rather than typing `nodejs`.

- **Python v2.7** Included w/ most Linux distributions
- **C ++ Compiler (gcc 4.8.1 or later)** Gcc is included with most Linux distributions. To confirm that gcc is installed, enter the command `gcc –version`. If gcc is not installed, you can enter one of the following commands:
- **Red Hat** `sudo yum install gcc`
- **Debian/Ubuntu** `sudo apt-get update sudo apt-get install build-essential`
- **Arch Linux** `sudo pacman -S gcc`
- **Libsecret** Enter one of the following commands:
- **Red Hat** `sudo yum install libsecret-devel`
- **Debian/Ubuntu** `sudo apt-get install libsecret-1-dev`
- **Arch Linux** `sudo pacman -S libsecret`
- **Make** Make is included with most Linux distributions. To confirm that Make is installed, enter the command `make –-version`. If Make is not installed, you can enter one of the following commands:
- **Red Hat** `sudo yum install devtoolset-7`
- **Debian/Ubuntu** `sudo apt-get install build-essential`
- **Arch Linux** `sudo pacman -S base-devel`

More information:

- Product overview
- Installing Zoe Brightside
- Creating a profile
- Testing connection to z/OSMF

Obtaining installation files

The Zoe installation files are distributed as a PAX file that contains the runtimes and the scripts to install and launch the z/OS runtime and the runtime for the command line interface.

To obtain a Zoe PAX file, visit Zoe download page in GitHub. For each release, there is a PAX file named `project_zoe-v.r.m.pax`, where

- `v` indicates the version
- `r` indicates the release number
- `m` indicates the modification number

The numbers are incremented each time a release is created so the higher the numbers, the later the release. Use your web browser to download the PAX file by saving it to a folder on your desktop.

Follow these steps to transfer the PAX file to z/OS and prepare it to install the Zoe runtime.

1. Transfer the PAX file to z/OS.

   a. Open a terminal in Mac OS/X, or command prompt in Windows OS, and navigate to the directory where you downloaded the Zoe PAX file.

b. Connect to z/OS using SFTP. Issue the following command:

```
sftp <userID@ip.of.zos.box>
```

If SFTP is not available or if you prefer to use FTP, you can issue the following command instead:

```
ftp <userID@ip.of.zos.box>
```

Note: When you use FTP, switch to binary file transfer mode by issuing the following command:

```
bin
```

c. Navigate to the target directory on z/OS.

After you connect to z/OS and enter your password, you will be entered into the Unix file system. Navigate to the directory you wish to transfer the Zoe PAX file into.

- To see what directory you are in, type `pwd`.
- To switch directory, type `cd`.
- To list the contents of a directory, type `ls`.
- To create a directory, type `mkdir`.

d. When you are in the directory you want to transfer the Zoe PAX file into, issue the following command:

```
put <pax-file-name>.pax
```

Where *pax-file-name* is a variable that indicates the full name of the PAX file you downloaded. For example, zoe-0.8.1.pax.

Note: When your terminal is connected to z/OS through FTP or SFTP, you can prepend commands with `l` to have them issued against your desktop. To list the contents of a directory on your desktop, type `lls` where `ls` will list contents of a directory on z/OS.

2. When the PAX file has transferred, expand the PAX file by issuing the following command in an ssh session:

```
pax -ppx -rf <pax-file-name>.pax
```

Where *pax-file-name* is a variable that indicates the name of the PAX file you downloaded. For example, zoe-0.8.1.pax.

This will expand to a file structure.

```
/files
/install
/scripts
...
```

Note: The PAX file will expand into the current directory. A good practice is to keep the installation directory apart from the directory that contains the PAX file. To do this, you can create a directory such as /zoe/paxes that contains the PAX files, and another such as /zoe/builds. Use SFTP to transfer the Zoe PAX file into the /zoe/paxes directory, use the `cd` command to switch into /zoe/builds and run the command `pax -ppx -rf ../../paxes/<zoe-v.r.m>.pax`. The /install folder will be created inside the zoe/builds directory from where the install can be launched.

Installing zLUX and explorer server

You install zLUX and explorer server on z/OS.

Before you install the runtime on z/OS, ensure that your environment meets the requirements. See System requirements.

Installing the Zoe runtime on z/OS

1. Navigate to the directory where the install archive was unpacked into. Locate the `/install` directory.

```
     /install
        /zoe-install.sh
        /zoe-install.yaml
```

2. Review zoe-install.yaml which contains the following properties.

   - install:rootDir is the directory that Zoe will be installed into to create a Zoe runtime. The default directory is `~/zoe/0.8.3`. The user's home directory is used as a default value to help ensure that the installing user has permission to create the directories needed for the install. If the Zoe runtime is going to be used by different users it may be more appropriate to use another directory, such as `/var/zoe/v.r.m`.

   You may run the install multiple times with different values in the `zoe-install.yaml` file to create separate installations of the Zoe runtime. The directory that Zoe is installed into must be empty. The install script will exit if the directory is not empty and create the directory if it does not exist.

   - explorer-server has two ports, one for HTTP and one for HTTPs. The liberty server is used for the explorer-ui components.
   - zlux-server has three ports: the HTTP and HTTPs ports that are used by the zLUX window manager server, and the port that is used by the ZSS server.

```
install:
 rootDir=/var/zoe/0.8.3

explorer-server:
  httpPort=7080
  httpsPort=7443

# http and https ports for the node server
zlux-server:
  httpPort=8543
  httpsPort=8544
  zssPort=8542
```

   If all of these port values are OK then you do not need to change them. These ports must not be already in use for the Zoe runtime servers to be able to allocate them.

   To determine which ports are not available, follow these steps:

   - To display a list of ports that are in use, issue the following command:

```
TSO NETSTAT
```

   - To display a list of reserved ports, issue the following command:

```
TSO NETSTAT PORTLIST
```

   The zoe-install.yaml also contains the telnet and SSH port with defaults of 23 and 22. If your z/OS LPAR is using different ports, edit the values. This is to allow the TN3270 terminal desktop app to connect as well as the VT terminal desktop app. Unlike the ports needed by the Zoe runtime for its zLUX and explorer server which must be unused, the terminal ports are expected to be in use.

```
# Ports for the TN3270 and the VT terminal to connect to
terminals:
    sshPort=22
    telnetPort=23
```

3. Execute the zoe-install.sh script

With the current directory being the `/install` directory, execute the script `zoe-install.sh` by issuing the following command:

```
zoe-install.sh
```

You might receive the following error that the file cannot be executed.

```
zoe-install.sh: cannot execute
```

The error is due to that the install script does not have execute permission. To add execute permission, issue the following command:

```
chmod u+x zoe-install.sh.
```

When the `zoe-install.sh` script runs, it performs a number of steps broken down into sections. These are covered more in the section Troubleshooting installing the Zoe runtime.

Starting and stopping the Zoe runtime on z/OS

Zoe has two runtime components on z/OS, the explorer server and the zLUX server. When you run the ZOESVR PROC, it starts both these components. The zLUX server startup script also starts the zSS server, so starting the ZOESVR PROC starts all three servers, and stopping it stops all three.

Starting the ZOESVR PROC

To start the ZOESVR PROC, run the `zoe-start.sh` script at the Unix Systems Services command prompt:

```
cd $ZOE_ROOT_DIR/scripts
./zoe-start.sh
```

where *$ZOE_ROOT_DIR* is the directory where you installed the Zoe runtime. This script starts the ZOESVR PROC for you so you don't have to log on to TSO and use SDSF.

If you prefer to use SDSF to start Zoe, start ZOESVR by issuing the following operator command in SDSF:

```
/S ZOESVR
```

By default, Zoe uses the runtime version that you most recently installed. To start a different runtime, specify its server path on the START command:

```
/S ZOESVR,SRVRPATH='$ZOE_ROOT_DIR/explorer-server'
```

To test whether the explorer server is active, open the URL `https://<hostname>:7443/ui`.

The port number 7443 is the default port and can be overridden through the `zoe-install.yaml` file before the `zoe-install.sh` script is run. See Installing Zoe runtime on z/OS.

Stopping the ZOESVR PROC

To stop the ZOESVR PROC, run the `zoe-stop.sh` script at the Unix Systems Services command prompt:

```
cd $ZOE_ROOT_DIR/scripts
./zoe-stop.sh
```

If you prefer to use SDSF to stop Zoe, stop ZOESVR by issuing the following operator command in SDSF:

```
/C ZOESVR
```

Either of the methods will stop the explorer server, the zLUX server, and the zSS server.

When you stop the ZOESVR, you might get the following error message:

```
IEE842I ZOESVR DUPLICATE NAME FOUND- REENTER COMMAND WITH 'A='
```

This is because there is more than one started task named ZOESVR. To resolve the issue, stop the required ZOESVR instance by using one of the following methods:

- Issue the following commands:

```
/C ZOESVR,A=asid
```

You can obtain the *asid* from the value of `A=asid` when you issue the following commands:

```
/D A,ZOESVR
```

- Select the instance in SDSF and type `C` for CANCEL in the NP column next to the job.

Verifying installation

After you complete the installation of Project Zoe, use the following procedures to verify that Project Zoe is installed correctly and is functional.

Verifying zLUX installation

If zLUX is installed correctly, you can open the MVD from a supported browser.

From a supported browser, open the MVD at `https://myhost:httpsPort/ZLUX/plugins/com.rs.mvd/web/index.html`

where:

- *myHost* is the host on which you installed the Zoe Node Server.
- *httpPort* is the port number that was assigned to `node.http.port` in `zluxserver.json`.
- *httpsPort* is the port number that was assigned to *node.https.port* in `zluxserver.json`. For example, if the Zoe Node Server is installed on host *myhost* and the port number that is assigned to the HTTP port is 12345, you specify `https://myhost:12345/ZLUX/plugins/com.rs.mvd/web/index.htm`.

Setting up terminal application plug-ins

Follow these optional steps to configure the default connection to open for the terminal application plug-ins.

Setting up the TN3270 mainframe terminal application plug-in

`_defaultTN3270.json` is a file in `tn3270-ng2/`, which is deployed during setup. Within this file, you can specify the following parameters to configure the terminal connection:

```
  "host": <hostname>
  "port": <port>
  “security”: {
  type: <”telnet” or “tls”>
}
```

Setting up the VT Terminal application plug-in

`_defaultVT.json` is a file in `vt-ng2/`, which is deployed during setup. Within this file, you can specify the following parameters to configure the terminal connection:

```
“host”:<hostname>
“port”:<port>
“security”: {
  type: <”telnet” or “ssh”>
}
```

Verifying explorer server installation

After explorer server is installed and the ZOESVR procedure is started, you can verify the installation from an Internet browser by using the following case-sensitive URL:

https://<your.server>:<atlasport>/Atlas/api/system/version

where *your.server* is the host name or IP address of the z/OS® system where explorer server is installed, and *atlasport* is the port number that is chosen during installation. You can verify the port number in the `server.xml` file that is located in the explorer server installation directory, which is `/var/atlas/wlp/usr/servers/Atlas/server.xml`by default. Look for the `httpsPort` assignment in the `server.xml` file, for example: httpPort="7443".

This URL sends an HTTP GET request to the Liberty Profile explorer server. If explorer server is installed correctly, a JSON payload that indicates the current explorer server application version is returned. For example:

```
{ "version": "V0.0.1" }
```

Note: The first time that you interact with the explorer server, you are prompted to enter an MVS™ user ID and password. The MVS user ID and password are passed over the secure HTTPS connection to establish authentication.

After you verify that explorer server was successfully installed, you can access the UI at:

https://<your.server>:<atlasport>/ui/#/

If explorer server is not installed successfully, see Troubleshooting installation for solutions.

Verifying the availability of explorer server REST APIs

To verify the availability of all explorer server REST APIs, use the Liberty Profile's REST API discovery feature from an internet browser with the following URL. This URL is case-sensitive.

https://<your.server>:<atlasport>/ibm/api/explorer

With the discovery feature, you can also try each discovered API. The users who verify the availability must have access to their data sets and job information by using relevant explorer server APIs. This ensures that your z/OSMF configuration is valid, complete, and compatible with the explorer server application. For example, try the following APIs:

Explorer server: JES Jobs APIs

`GET  /Atlas/api/jobs`

This API returns job information for the calling user.

Explorer server: Data set APIs

`GET /Atlas/api/datasets/userid.**`

This API returns a list of the userid.** MVS data sets.

Troubleshooting installing the Zoe runtime

1. Environment variables

   To prepare the environment for the Zoe runtime, a number of ZFS folders need to be located for prerequisites on the platform that Zoe needs in order to operate. These can be set as environment variables before the script is run. If the environment variables are not set, the install script will attempt to locate default values.

   - `ZOE_ZOSMF_PATH`: The path where z/OSMF is installed. Defaults to `/usr/lpp/zosmf/lib/defaults/servers/zosmfServer`
   - `ZOE_JAVA_HOME`: The path where 64 bit Java 8 or later is installed. Defaults to `/usr/lpp/java/J8.0_64`
   - `ZOE_SDSF_PATH`: The path where SDSF is installed. Defaults to `/usr/lpp/sdsf/java`
   - `ZOE_EXPLORER_HOST`: The IP address of where the explorer servers are launched from. Defaults to running `hostname -c`

The first time the script is run if it has to locate any of the environment variables, the script will add lines to the current user's home directory `.profile` file to set the variables. This ensures that the next time the same user runs the install script, the previous values will be used.

Note: If you wish to set the environment variables for all users, add the lines to assign the variables and their values to the file `/etc/.profile`.

If the environment variables for ZOE_ZOSMF_PATH, ZOE_JAVA_HOME, or ZOE_SDSF_PATH are not set and the install script cannot determine a default location, the install script will prompt for their location. The install script will not continue unless valid locations are provided.

2. Expanding the PAX files

   The install script will create the Zoe runtime directory structure using the `install:rootDir` value in the `zoe-install.yaml` file. The runtime components of the Zoe server are then unpaxed into the directory that contains a number of directories and files that make up the Zoe runtime.

   If the expand of the PAX files is successful, the install script will report that it ran its install step to completion.

3. Changing Unix permissions

   After the install script has laid down the contents of the Zoe runtime into the `rootDir`, the next step is to set the file and directory permissions correctly to allow the Zoe runtime servers to start and operate successfully.

   The install process will execute the file `scripts/zoe-runtime-authorize.sh` in the Zoe runtime directory. If the script is successful, the result will be reported. If for any reason the script fails to run because of insufficient authority by the user running the install, the install process will report the errors. A user with sufficient authority should then run the `zoe-runtime-authorize.sh`. If you attempt to start the Zoe runtime servers without the `zoe-runtime-authorize.sh` having successfully completed, the results are unpredictable and Zoe runtime startup or runtime errors will occur.

4. Creating the PROCLIB member to run the Zoe runtime

   Note: The name of the PROCLIB member may vary depending on the standards in place at each z/OS site, however for this documentation we assume that the PROCLIB member is called ZOESVR.

   At the end of the installation, a Unix file `ZOESVR.jcl` is created under the directory where the runtime was installed into, `$INSTALL_DIR/files/templates`. The contents of this file need to be tailored and placed in a JCL member of the PROCLIB concatenation for the Zoe runtime to be executed as a started task. The install script does this automatically, trying data sets `USER.PROCLIB`, other PROCLIB data sets found in the PROCLIB concatenation and finally `SYS1.PROCLIB`.

   If this succeeds, you will see a message like the following one:

   ```
   PROC ZOESVR placed in USER.PROCLIB
   ```

   Otherwise you will see messages beginning with the following information:

   ```
   Failed to put ZOESVR.JCL in a PROCLIB dataset.
   ```

   In this case, you need to copy the PROC manually. The TSO `oget` command can be used to copy the `ZOESVR.jcl` file to the preferred PROCLIB.

   ```
   oget '$INSTALL_DIR/files/templates/ZOESVR.jcl' 'MY.USER.PROCLIB(ZOESVR)'
   ```

   You can place the PROC in any PROCLIB data set in the PROCLIB concatenation, but some data sets such as `SYS1.PROCLIB` may be restricted, depending on the permission of the user.

   You can tailor the JCL at this line

   ```
   //ZOESVR PROC SRVRPATH='/zoe/install/path/explorer-server'
   ```

   to replace the `/zoe/install/path` with the location of the Zoe runtime directory that contains the explorer server. Otherwise you must specify that path on the START command when you start Zoe in SDSF:

```
/S ZOESVR,SRVRPATH='$ZOE_ROOT_DIR/explorer-server'
```

5. Adding RACF authorizations for Zoe

   To define RACF authorizations for Zoe, the following steps are required:

   a. Define the PROC named ZOESVR to be a started task.

   ```
   RDEFINE STARTED ZOESVR.* UACC(NONE) STDATA(USER(IZUSVR) GROUP(IZUADMIN)
    PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))

   SETROPTS REFRESH RACLIST(STARTED)
   ```

   b. Add the user who is performing the install to the IZUADMIN group.

   ```
   CONNECT (userid) GROUP(IZUADMIN)
   ```

Troubleshooting installing zLUX

To help zLUX research any problems you might encounter, collect as much of the following information as possible and open an issue in GitHub with the collected information.

- Project Zoe version and release level
- z/OS release level
- Job output and dump (if any)
- Javascript console output (Web Developer toolkit accessible by pressing F12)
- Log output from the Zoe Node Server
- Error message codes
- Screenshots (if applicable)
- Other relevant information (such as the version of Node.js that is running on the Zoe Node Server and the browser and browser version).

Troubleshooting installing explorer server

If explorer server REST APIs do not work, check the following items:

- Check whether your Liberty explorer server is running.

  You can check this in the Display Active (DA) panel of SDSF under ISPF. The ZOESVR started task should be running. If the ZOESVR task is not running, start the explorer server by using the following START operator command:

  ```
  /S ZOESVR
  ```

  You can also use the operator command `/D A,ZOESVR` to verify whether the task is active, which alleviates the need for the (DA) panel of SDSF. If the started task is not running, ensure that your ZOESVR procedure resides in a valid PROCLIB data set, and check the task's job output for errors.
- Check whether the explorer server is started without errors.

  In the Display Active (DA) panel of SDSF under ISPF, select the ZOESVR job to view the started task output. If the explorer server is started without errors, you can see the following messages:

  ```
  CWWKE0001I: The server Atlas has been launched.
  ```

  ```
  CWWKF0011I: The server Atlas is ready to run a smarter planet.
  ```

  If you see error messages that are prefixed with "ERROR" or stack traces in the ZOESVR job output, respond to them.
- Check whether the URL that you use to call explorer server REST APIs is correct. For example: https://your.server:atlasport/Atlas/api/system/version. The URL is case-sensitive.

- Ensure that you enter a valid z/OS® user ID and password when initially connecting to the explorer server.
- If testing the explorer server REST API for jobs information fails, check the z/OSMF IZUSVR1 task output for errors. If no errors occur, you can see the following messages in the IZUSVR1 job output:

```
CWWKE0001I : The server zosmfServer has been launched.
```

```
CWWKF0011I: The server zosmfServer is ready to run a smarter planet.
```

  If you see error messages, respond to them.

  For RESTJOBS, you can see the following message if no errors occur:

```
CWWKZ0001I: Application IzuManagementFacilityRestJobs started in n.nnn
 seconds.
```

  You can also call z/OSMF RESTJOBS APIs directly from your Internet browser with a URL, for example,

  https://your.server:securezosmfport/zosmf/restjobs/jobs

  where the *securezosmfport* is 443 by default. You can verify the port number by checking the *izu.https.port* variable assignment in the z/OSMF `bootstrap.properties` file.

  You might get error message IZUG846W, which indicates that a cross-site request forgery (CSRF) was attempted. To resolve the issue, update your browser by adding the `X-CSRF-ZOSMF-HEADER` HTTP custom header to every cross-site request. This header can be set to any value or an empty string (""). For details, see the z/OSMF documentation. If calling the z/OSMF RESTJOBS API directly fails, fix z/OSMF before explorer server can use these APIs successfully.
- If testing the explorer server REST API for data set information fails, check the z/OSMF IZUSVR1 task output for errors and confirm that the z/OSMF RESTFILES services are started successfully. If no errors occur, you can see the following message in the IZUSVR1 job output:

```
CWWKZ0001I: Application IzuManagementFacilityRestFiles started in n.nnn
 seconds.
```

  You can also call z/OSMF RESTFILES APIs directly from your internet browser with a URL, for example,

  https://your.server:securezosmfport/zosmf/restfiles/ds?dslevel=userid.**

  where the *securezosmfport* is 443 by default. You can verify the port number by checking the *izu.https.port* variable assignment in the z/OSMF `bootstrap.properties` file.

  You might get error message IZUG846W, which indicates that a cross-site request forgery (CSRF) was attempted. To resolve the issue, update your browser by adding the `X-CSRF-ZOSMF-HEADER` HTTP custom header to every cross-site request. This header can be set to any value or an empty string (""). For details, see the z/OSMF documentation. If calling the z/OSMF RESTFILES API directly fails, fix z/OSMF before explorer server can use these APIs successfully.

  Tip: The z/OSMF installation step of creating a valid IZUFPROC procedure in your system PROCLIB might be missed. For more information, see the *z/OSMF Configuration Guide*.

  The IZUFPROC member resides in your system PROCLIB, which is similar to the following sample:

```
//IZUFPROC PROC ROOT='/usr/lpp/zosmf'  /* zOSMF INSTALL ROOT     */
//IZUFPROC EXEC PGM=IKJEFT01,DYNAMNBR=200
//SYSEXEC  DD DISP=SHR,DSN=ISP.SISPEXEC
//         DD DISP=SHR,DSN=SYS1.SBPXEXEC
//SYSPROC  DD DISP=SHR,DSN=ISP.SISPCLIB
//         DD DISP=SHR,DSN=SYS1.SBPXEXEC
//ISPLLIB  DD DISP=SHR,DSN=SYS1.SIEALNKE
//ISPPLIB  DD DISP=SHR,DSN=ISP.SISPPENU
//ISPTLIB  DD RECFM=FB,LRECL=80,SPACE=(TRK,(1,0,1))
//         DD DISP=SHR,DSN=ISP.SISPTENU
```

```
//ISPSLIB  DD DISP=SHR,DSN=ISP.SISPSENU
//ISPMLIB  DD DISP=SHR,DSN=ISP.SISPMENU
//ISPPROF  DD DISP=NEW,UNIT=SYSDA,SPACE=(TRK,(15,15,5)),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//IZUSRVMP DD PATH='&ROOT./defaults/izurf.tsoservlet.mapping.json'
//SYSOUT   DD SYSOUT=H
//CEEDUMP  DD SYSOUT=H
//SYSUDUMP DD SYSOUT=H
//
```

Note: You might need to change paths and data sets names to match your installation.

A known issue and workaround for RESTFILES API can be found at TSO SERVLET EXCEPTION ATTEMPTING TO USE RESTFILE INTERFACE.

- Check your system console log for related error messages and respond to them.

If the explorer server cannot connect to the z/OSMF server, check the following item:

By default, the explorer server communicates with the z/OSMF server on the localhost address. If your z/OSMF server is on a different IP address to the explorer server, for example, if you are running z/OSMF with Dynamic Virtual IP Addressing (DVIPA), you can change this by adding a ZOSMF_HOST parameter to the server.env file. For example: ZOSMF_HOST=winmvs27.

Installing Zoe Brightside

As a systems programmer or application developer, you install Zoe Brightside on your PC. You can use either of the following methods to install Zoe Brightside.

Install Zoe Brightside from local package

Install Zoe Brightside on PCs that are running a Windows, Linux, or macOS operating system.

Follow these steps:

1. Address the prerequisites.
2. Obtain the Project Zoe installation files, which includes the brightside-bundle.zip file. Use FTP to distribute the brightside-bundle.zip file to client workstations.
3. Open a command line window. For example, Windows Command Prompt. Browse to the directory where you downloaded the Zoe Brightside installation bundle (.zip file). Issue the following command to unzip the files:

   ```
   unzip brightside-bundle.zip
   ```

   The command expands four TGZ packages into your working directory - Zoe Brightside, one plug-in, and the odbc_cli folder.
4. Issue the following command to install Zoe Brightside on your PC:

   ```
   npm install -g brightside-core-1.0.1.tgz
   ```

   Note: On Linux systems, you might need to append `sudo` to your `npm` commands so that you can issue the install and uninstall commands. For more information, see Troubleshooting Installing Zoe Brightside.

   Zoe Brightside is installed on your PC. See Installing Plug-ins for information about the commands for installing plug-ins from the package.
5. Create a `zosmf` profile so that you can issue commands that communicate with z/OSMF.

   Note: For information about how to create a profile, see Creating a Zoe Brightside profile.

Tip: Zoe Brightside profiles contain information that is required for the product to interact with remote systems. For example, host name, port, and user ID. Profiles let you target unique systems, regions, or instances for a command. Most Zoe Brightside command groups require a Zoe Brightside `zosmf` profile.

After you install and configure Zoe Brightside, you can issue the `bright --help` command to view a list of available commands. For moreinformation, see Display Help.

Install Zoe Brightside from Bintray registry

If your PC is connected to the Internet, you can use the following method to install Zoe Brightside from an npm registry.

Follow these steps:

1. Issue the following command to set the registry to the Zoe Brightside scoped package on Bintray. In addition to setting the scoped registry, your non-scoped registry must be set to an npm registry that includes all of the dependencies for Zoe Brightside, such as the global npm registry:

   ```
   npm config set @brightside:registry https://api.bintray.com/npm/ca/
   brightside
   ```

2. Issue the following command to install Zoe Brightside from the registry:

   ```
   npm install -g @brightside/core@latest
   ```

   Zoe Brightside is installed on your PC. For information about plug-ins for Zoe Brightside, see Extending Zoe Brightside.

3. Create a `zosmf` profile so that you can issue commands that communicate with z/OSMF. For information about how to create a profile, see Creating a Zoe Brightside profile.

   Tip: Zoe Brightside profiles contain information that is required for the product to interact with remote systems. For example, host name, port, and user ID. Profiles let you target unique systems, regions, or instances for a command. Most Zoe Brightside command groups require a Zoe Brightside `zosmf` profile.

After you install and configure Zoe Brightside, you can issue the `bright --help` command to view a list of available commands. For more information, see How to display Zoe Brightside help.

Note: You might encounter problems when you attempt to install Zoe Brightside depending on your operating system and environment. For more information and workarounds, see Troubleshooting Installing Zoe Brightside.

More information:

- Uninstalling Zoe Brightside

Creating a Zoe Brightside profile

Profiles are a Zoe Brightside function that lets you store configuration information for use on multiple commands. You can create a profile that contains your username, password, and connection details for a particular mainframe system, then reuse that profile to avoid typing it again on every command. You can switch between profiles to quickly target different mainframe subsystems.

Important! A `zosmf` profile is required to issue most Zoe Brightside commands. The first profile that you create becomes your default profile. When you issue any command that requires a `zosmf` profile, the command executes using your default profile unless you specify a specific profile name on that command.

Follow these steps:

1. To create a `zosmf` profile, issue the following command. Refer to the available options in the help text to define your profile:

```
bright profiles create zosmf-profile --help
```

Note: After you create a profile, verify that it can communicate with z/OSMF. For more information, see Test Connection to z/OSMF.

Testing connection to z/OSMF

After you configure a Zoe Brightside `zosmf` profile to connect to z/OSMF on your mainframe systems, you can issue a command at any time to receive diagnostic information from the server and confirm that your profile can communicate with z/OSMF.

Tip: In this documentation we provide command syntax to help you create a basic profile. We recommend that you append `--help` to the end of commands in the product to see the complete set of commands and options available to you. For example, issue `bright profiles --help` to learn more about how to list profiles, switch your default profile, or create different profile types.

After you create a profile, run a test to verify that Zoe Brightside can communicate properly with z/OSMF. You can test your default profile and any other Zoe Brightside profile that you created.

Default profile

- Verify that you can use your default profile to communicate with z/OSMF by issuing the following command:

```
bright zosmf check status
```

Specific profile

- Verify that you can use a specific profile to communicate with z/OSMF by issuing the following command:

```
bright zosmf check status --zosmf-profile <profile_name>
```

The commands return a success or failure message and display information about your z/OSMF server. For example, the z/OSMF version number and a list of installed plug-ins. Report any failure to your systems administrator and use the information for diagnostic purposes.

More information:

- Creating a Zoe Brightside profile

Troubleshooting installing Zoe Brightside

The following issues are known to exist in this release of Zoe Brightside:

- **Additional syntax required to complete macOS and Linux installations.**

  Depending on how you configured Node.js on Linux or Mac, you might need to add the prefix `sudo` before the `npm install -g` command or the `npm uninstall -g` command. This step gives Node.js write access to the installation directory.

- **The `npm install -g` command might fail several times due to an `EPERM` error (Windows).**

  This behavior is due to a problem with Node Package Manager (npm) on Windows. There is an open issue on the npm GitHub repository to fix the defect.

  If you encounter this problem, some users report that repeatedly attempting to install Zoe Brightside yields success. Some users also report success using the following workarounds:

  – Issue the `npm cache clean` command.

  – Uninstall and reinstall Zoe Brightside. For more information, see Installing Zoe Brightside.

– Issue the `npm install -g brightside --no-optional` command.

- **The `npm install -g` command might fail due to an `npm ERR! Cannot read property 'pause' of undefined` error.**

  This behavior is due to a problem with Node Package Manager (npm). If you encounter this problem, revert to a previous version of npm that does not contain this defect. To revert to a previous version of npm, issue the following command:

  ```
  npm install npm@5.3.0 -g
  ```

- **Node.js commands do not respond as expected.**

  When you try to issue node commands and you do not receive the expected output, there might be a program that is named *node* on your path. The Node.js installer automatically adds a program that is named *node* to your path. When there are pre-existing programs that are named *node* on your computer, the program that appears first in the path is used. To correct this behavior, change the order of the programs in the path so that Node.js appears first.

Uninstalling Project Zoe

You can uninstall Project Zoe if you no longer need to use the product. Follow these procedures to uninstall the components of Project Zoe.

Uninstalling zLUX

1. The zLUX server is run under the ZOESVR started task, so it should terminate when ZOESVR is stopped. If it does not, use one of the following standard process signals to stop the server:
   - SIGHUP
   - SIGTERM
   - SIGKILL
2. Delete the original folders (except in the case where you have customized the installation to point to folders other than the original folders).

Uninstalling explorer server

To uninstall the explorer server, take the following steps:

1. Stop your Explorer Liberty server by running the following operator command:

   ```
   C ZOESVR
   ```

2. Delete the ZOESVR member from your system PROCLIB data set.
3. Remove RACF® (or equivalent) definitions with the following command:

   ```
   RDELETE STARTED (ZOESVR.*)
   SETR RACLIST(STARTED) REFRESH
   REMOVE (userid) GROUP(IZUUSER)
   ```

4. Delete the z/OS® UNIX™ System Services explorer server directory and files from the explorer server installation directory by issuing the following command:

   ```
   rm -R /var/atlas #*Atlas Installation Directory*
   ```

   Notes:

   - You might need super user authority to run this command.
   - You must identify the explorer server installation directory correctly. Running a recursive remove command with the wrong directory name might delete critical files.

Uninstalling Zoe Brightside

You can uninstall Zoe Brightside when you no longer want to use the product.

Important! The uninstall process does not delete the profiles and credentials that you created when using the product from your PC. To delete the profiles from your PC, delete them before you uninstall Zoe Brightside.

The following steps describe how to list the profiles that you created, delete the profiles, and uninstall Zoe Brightside.

Follow these steps:

1. Open a command line window.

   Note: If you do not want to delete the Zoe Brightside profiles from your PC, go to Step 5.

2. List all profiles that you created for a Command Group by issuing the following command:

   ```
   bright profiles list <profileType>
   ```

   Example:

   ```
   $ bright profiles list zosmf
   The following profiles were found for the module zosmf:
   'SMITH-123' (DEFAULT)
   smith-123@SMITH-123-W7 C:\Users\SMITH-123
   $
   ```

3. Delete all of the profiles that are listed for the command group by issuing the following command:

   Tip: For this command, use the results of the `list` command.

   Note: When you issue the `delete` command, it deletes the specified profile and its credentials from the credential vault in your PC's operating system.

   ```
   bright profiles delete <profileType> <profileName> --force
   ```

   Example:

   ```
   bright profiles delete zosmf SMITH-123 --force
   ```

4. Repeat Steps 2 and 3 for all Zoe Brightside command groups and profiles.
5. Uninstall Zoe Brightside by issuing one of the following commands:
   - If you installed Zoe Brightside from the package, issue the following command

     ```
     npm uninstall --global @brightside/core
     ```

   - If you installed Zoe Brightside from the online registry, issue the following command:

     ```
     npm uninstall --global brightside
     ```

     The uninstall process removes all Zoe Brightside installation directories and files from your PC.

6. Delete the following directory on your PC. The directory contains the Zoe Brightside log files and other miscellaneous files that were generated when you used the product.

   Tip: Deleting the `C:\Users\<user_name>\.brightside` directory does not harm your PC.

7. If you installed Zoe Brightside from the online registry, issue the following command to clear your scoped npm registry:

   ```
   npm config set @brightside:registry
   ```

More Information:

- Installing Zoe Brightside

Chapter 3. Using Project Zoe

After you install and start Project Zoe, you can perform tasks with each component. See the following sections for details.

- **Using zLUX**
- **Using APIs**
- **Using Zoe Brightside**

Using zLUX

zLUX provides the ability to create application plug-ins. For more information, see Creating zLUX application plug-ins.

Navigating MVD

From the Mainframe Virtual Desktop (MVD), you can access the Project Zoe applications.

Accessing the MVD

From a supported browser, open the MVD at `https://myhost:httpsPort/ZLUX/plugins/com.rs.mvd/web/index.html`

where:

- *myHost* is the host on which you are running the Zoe Node Server.
- *httpsPort* is the value that was assigned to *node.https.port* in `zluxserver.json`. For example, if you run the Zoe Node Server on host *myhost* and the value that is assigned to *node.https.port* in `zluxserver.json` is 12345, you would specify `https://myhost:12345/ZLUX/plugins/com.rs.mvd/web/index.html`.

Logging in and out of the MVD

1. To log in, enter your mainframe credentials in the **Username** and **Password** fields.
2. Press Enter. Upon authentication of your user name and password, the desktop opens.

To log out, click the the avatar in the lower right corner and click **Sign Out**.

Using Explorers within zLUX

The explorer server provides a sample web client that can be used to view and manipulate the Job Entry Subsystem (JES), data sets, z/OS UNIX System Services (USS), and System log.

The following views are available from the explorer server Web UI and are accesible via the explorer server icon located in the application draw of MVD (Navigation between views can be performed using the menu draw located in the top left corner of the explorer server Web UI):

JES Explorer

Use this view to query JES jobs with filters, and view the related steps, files, and status. You can also purge jobs from this view.

Syslog

Use this view to see the system log by using a web socket. Whenever messages are written, the view is refreshed automatically.

You can also open a JES spool file for an active job and view its content that is refreshed through a web socket.

Data set Explorer

Use this view to browse the MVS™ file system by using a high-level qualifier filter. With the Dataset Explorer, you can complete the following tasks:

- List the members of partitioned data sets.
- Create new data sets using attributes or the attributes of an existing data set ("Allocate Like").
- Submit data sets that contain JCL to Job Entry Subsystem (JES).
- Edit sequential data sets and partitioned data set members with basic syntax highlighting and content assist for JCL and REXX.
- Conduct basic validation of record length when editing JCL.
- Delete data sets and members.
- Open data sets in full screen editor mode, which gives you a fully qualified link to that file. The link is then reusable for example in help tickets.

UNIX file Explorer

Use this view to browse the USS files by using a path. With the UNIX file Explorer, you can complete the following tasks:

- List files and folders.
- Create new files and folders.
- Edit files with basic syntax highlighting and content assist for JCL and REXX.
- Delete files and folders.

Using zLUX application plug-ins

Application plug-ins are applications that you can use to access the mainframe and to perform various tasks. Developers can create application plug-ins using a sample application as a guide.

Hello World

This sample application plug-in for developers demonstrates how to create a dataservice and how to create an application plug-in using Angular.

IFrame

This sample application plug-in for developers demonstrates how to embed pre-made webpages within the desktop as an application and how an application can request an action of another application (see the source code for details).

ZOS Subsystems

This application plug-in helps you find information about the important services on the mainframe, such as CICS, Db2, and IMS.

TN3270

This application plug-in provides a 3270 connection to the mainframe on which the Zoe Node Server runs.

VT Terminal

This application plug-in provides a connection to USS and UNIX.

zLUX logging

zLUX generates log files in the following default locations:

- Zoe Node Server: `zlux-example-server/log/nodeServer-yyyy-mm-dd-hh-mm.log`
- ZSS: `zlux-example-server/log/zssServer-yyyy-mm-dd-hh-mm.log`

Note that the Zoe Node Server logs and ZSS logs are timestamped in the format yyyy-mm-dd-hh-mm and older logs are deleted when a new log is created at server startup.

Controlling the zLUX logging location

The zLUX server writes log information to a file and to the screen. (On Windows, logs are written to a file only.)

ZLUX_NODE_LOG_DIR and ZSS_LOG_DIR environment variables

To control where the information is logged, use the environment variable *ZLUX_NODE_LOG_DIR*, for the Zoe Node Server, and *ZSS_LOG_DIR*, for ZSS. While these variables are intended to specify a directory, if you specify a location that is a file name, zLUX will write the logs to the specified file instead (for example: `/dev/null` to disable logging).

When you specify the environment variables *ZLUX_NODE_LOG_DIR* and *ZSS_LOG_DIR* and they are directories rather than files, zLUX will perform timestamping and cleanup.

ZLUX_NODE_LOG_FILE and ZSS_LOG_FILE environment variables

If you set the log file name for the node server by setting the *ZLUX_NODE_LOG_FILE* environment variable, or if you set the log file for ZSS by setting the *ZSS_LOG_FILE* environment variable, there will only be one log file, and it will be overwritten.

NOTE: When you set the *ZLUX_NODE_LOG_FILE* or *ZSS_LOG_FILE* environment variables, zLUX will not override the log names, set a timestamp, or delete the logs.

If zLUX cannot create the folder or file, the server will run (but it might not perform logging properly).

Retaining logs

By default, zLUX retains the last five logs. If you want to specify a different number of logs to retain, set *ZLUX_NODE_LOGS_TO_KEEP* (Zoe Node Server logs) or *ZSS_LOGS_TO_KEEP* (ZSS logs) to the number of logs that you want to keep. The default is 5.

Using APIs

Access and modify your z/OS resources such as jobs, data sets, z/OS UNIX System Services files by using APIs.

Using explorer server REST APIs

Explorer server REST APIs provide a range of REST APIs through a Swagger defined description, and a simple interface to specify API endpoint parameters and request bodies along with the response body and return code. With explorer server REST APIs, you can see the available API endpoints and try the endpoints within a browser. Swagger documentation is available from an Internet browser with a URL, for example, https://your.host:atlas-port/ibm/api/explorer.

- **Data Set APIs**
- **Job APIs**
- **Persistent Data APIs**
- **System APIs**
- **USS File APIs**
- **z/OS System APIs**

Data set APIs

Use data set APIs to create, read, update, delete, and list data sets. See the following table for the operations available in data set APIs and their descriptions and prerequisites.

REST API	Description	Prerequisite
`GET /Atlas/api/datasets/{filter}`	Get a list of data sets by filter. Use this API to get a starting list of data sets, for example, **userid.****.	z/OSMF restfiles
`GET /Atlas/api/datasets/{dsn}/attributes`	Retrieve attributes of a data set(s). If you have a data set name, use this API to determine attributes for a data set name. For example, it is a partitioned data set.	z/OSMF restfiles
`GET /Atlas/api/datasets/{dsn}/members`	Get a list of members for a partitioned data set. Use this API to get a list of members of a partitioned data set.	z/OSMF restfiles
`GET /Atlas/api/datasets/{dsn}/content`	Read content from a data set or member. Use this API to read the content of a sequential data set or partitioned data set member. Or use this API to return a checksum that can be used on a subsequent PUT request to determine if a concurrent update has occurred.	z/OSMF restfiles
`PUT /Atlas/api/datasets/{dsn}/content`	Write content to a data set or member. Use this API to write content to a sequential data set or partitioned data set member. If a checksum is passed and it does not match the checksum that is returned by a previous GET request, a concurrent update has occurred and the write fails.	z/OSMF restfiles
`POST /Atlas/api/datasets/{dsn}`	Create a data set. Use this API to create a data set according to the attributes that are provided. The API uses z/OSMF to create the data set and uses the syntax and rules that are described in the *z/OSMF Programming Guide*.	z/OSMF restfiles
`POST /Atlas/api/datasets/{dsn}/{basedsn}`	Create a data set by using the attributes of a given base data set. When you do not know the attributes of a new data set, use this API to create a new data set by using the same attributes as an existing one.	z/OSMF
`DELETE /Atlas/api/datasets/{dsn}`	Delete a data set or member. Use this API to delete a sequential data set or partitioned data set member.	z/OSMF restfiles

Job APIs

Use Jobs APIs to view the information and files of jobs, and submit and cancel jobs. See the following table for the operations available in Job APIs and their descriptions and prerequisites.

REST API	Description	Prerequisite
`GET /Atlas/api/jobs`	Get a list of jobs. Use this API to get a list of job names that match a given prefix, owner, or both.	z/OSMF restjobs
`GET /Atlas/api/jobs/{jobName}/ids`	Get a list of job identifiers for a given job name. If you have a list of existing job names, use this API to get a list of job instances for a given job name.	z/OSMF restjobs
`GET /Atlas/api/jobs/{jobName}/ids/{jobId}/steps`	Get job steps for a given job. With a job name and job ID, use this API to get a list of the job steps, which includes the step name, the executed program, and the logical step number.	z/OSMF restjobs
`GET /Atlas/api/jobs/{jobName}/ids/{jobId}/steps/{stepNumber}/dds`	Get data set definitions (DDs) for a given job step. If you know a step number for a given job instance, use this API to get a list of the DDs for a given job step, which includes the DD name, the data sets that are described by the DD, the original DD JCL, and the logical order of the DD in the step.	z/OSMF restjobs
`GET /Atlas/api/jobs/{jobName}/ids/{jobId}/files`	Get a list of output file names for a job. Job output files have associated DSIDs. Use this API to get a list of the DSIDs and DD name of a job. You can use the DSIDs and DD name to read specific job output files.	z/OSMF restjobs
`GET /Atlas/api/jobs/{jobName}/ids/{jobId}/files/{fileId}`	Read content from a specific job output file. If you have a DSID or field for a given job, use this API to read the output file's content.	z/OSMF restjobs
`GET /Atlas/api/jobs/{jobName}/ids/{jobId}/files/{fileId}/tail`	Read the tail of a job's output file. Use this API to request a specific number of records from the tail of a job output file.	z/OSMF restjobs
`GET /Atlas/api/jobs/{jobName}/ids/{jobId}/subsystem`	Get the subsystem type for a job. Use this API to determine the subsystem that is associated with a given job. The API examines the JCL of the job to determine if the executed program is CICS®, DB2®, IMS™, or IBM® MQ.	z/OSMF restjobs
`POST /Atlas/api/jobs`	Submit a job and get the job id back. Use this API to submit a partitioned data set member or UNIX™ file.	z/OSMF restjobs

REST API	Description	Prerequisite
`DELETE /Atlas/api/jobs/{jobName}/{jobId}`	Cancel a job and purge its associated files. Use this API to purge a submitted job and the logged output files that it creates to free up space.	z/OSMF Running Common Information Model (CIM) server

Persistent Data APIs

Use Persistent Data APIs to create, read, update, delete metadata from persistent repository. See the following table for the operations available in Persistent Data APIs and their descriptions and prerequisites.

REST API	Description	Prerequisite
`PUT /Atlas/api/data`	Update metadata in persistent repository for a given resource and attribute name. With explorer server, you can store and retrieve persistent data by user, resource name, and attribute. A resource can have any number of attributes and associated values. Use this API to set a value for a single attribute of a resource. You can specify the resource and attribute names.	None
`POST /Atlas/api/data`	Create metadata in persistent repository for one or more resource/attribute elements. Use this API to set a group of resource or attributes values.	None
`GET /Atlas/api/data`	Retrieve metadata from persistent repository for a given resource (and optional attribute) name. Use this API to get all the attribute values or any particular attribute value for a given resource.	None
`DELETE /Atlas/api/data`	Remove metadata from persistent repository for a resource (and optional attribute) name. Use this API to delete all the attribute values or any particular attribute value for a given resource.	None

System APIs

Use System APIs to view the version of explorer server. See the following table for available operations and their descriptions and prerequisites.

REST API	Description	Prerequisite
`GET /Atlas/api/system/version`	Get the current explorer server version. Use this API to get the current version of the explorer server microservice.	None

USS File APIs

Use USS File APIs to create, read, update, and delete USS files. See the following table for the available operations and their descriptions and prerequisites.

REST API	Description	Prerequisite
`POST /Atlas/api/uss/files`	Use this API to create new USS directories and files.	z/OSMF restfiles
`DELETE /Atlas/api/uss/files{path}`	Use this API to delete USS directories and files.	z/OSMF resfiles
`GET /Atlas/api/files/{path}`	Use this API to get a list of files in a USS directory along with their attributes.	z/OSMF restfiles
`GET /Atlas/api/files/{path}/content`	Use this API to get the content of a USS file.	z/OSMF restfiles
`PUT /Atlas/api/files/{path}/content`	Use this API to update the content of a USS file.	z/OSMF resfiles

z/OS System APIs

Use z/OS system APIs to view information about CPU, PARMLIB, SYSPLEX, USER. See the following table for available operations and their descriptions and prerequisites.

REST API	Description	Prerequisite
`GET /Atlas/api/zos/cpu`	Get current system CPU usage. Use this API to get the current system CPU usage and other current system statistics.	None
`GET /Atlas/api/zos/parmlib`	Get system PARMLIB information. Use this API to get the PARMLIB data set concatenation of the target z/OS system.	None
`GET /Atlas/api/zos/sysplex`	Get target system sysplex and system name. Use this API to get the system and sysplex names.	None
`GET /Atlas/api/zos/username`	Get current userid. Use this API to get the current user ID.	None

Programming explorer server REST APIs

You can program explorer server REST APIs by referring to these examples:

- **Sending a GET request in Java**
- **Sending a GET request in JavaScript**
- **Sending a POST request in JavaScript**
- **Extended API sample in JavaScript**

Sending a GET request in Java

Here is sample code to send a GET request to explorer server in Java™.

```
public class JobListener implements Runnable {

    /*
```

```
    *   Perform an HTTPs GET at the given jobs URL and credentials
    *   targetURL e.g "https://host:port/Atlas/api/jobs?
owner=IBMUSER&prefix=*"
    *   credentials in the form of base64 encoded string of user:password
    */
    private String executeGET(String targetURL, String credentials) {
        HttpURLConnection connection = null;
        try {
            //Create connection
            URL url = new URL(targetURL);
            connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("GET");
            connection.setRequestProperty("Authorization", credentials);

            //Get Response
            InputStream inputStream = connection.getInputStream();
            BufferedReader bufferedReader = new BufferedReader(new
 InputStreamReader(inputStream));
            StringBuilder response = new StringBuilder();
            String line;

            //Process the response line by line
            while ((line = bufferedReader.readLine()) != null) {
                System.out.println(line);
            }

            //Cleanup
            bufferedReader.close();

            //Return the response message
            return response.toString();
        } catch (Exception e) {
            //handle any error(s)
        } finally {
            //Cleanup
            if (connection != null) {
                connection.disconnect();
            }
        }
    }
}
```

Sending a GET request in JavaScript

Here is sample code written in JavaScript™ using features from ES6 to send a GETrequest to explorer server.

```
const BASE_URL = 'hostname.com:port/Atlas/api';

// Call the jobs GET api to get all jobs with the userID IBMUSER
function getJobs(){
    let parameters = "prefix=*&owner=IBMUSER";
    let contentURL = `${BASE_URL}/jobs?${parameters}`;
    let result = fetch(contentURL, {credentials: "include"})
                    .then(response => response.json())
                        .catch((e) => {
                            //handle any error
                            console.log("An error occoured: " + e);
                        });
     return result;
}
```

Sending a POST request in JavaScript

Here is sample code written in JavaScript™ using features from ES6 to send a POST request to explorer server.

```
// Call the jobs POST api to submit a job from a data set
 (ATLAS.TEST.JCL(TSTJ0001))
function submitJob(){
    let payload = "{\"file\":\"'ATLAS.TEST.JCL(TSTJ0001)'\"}";
    let contentURL = `${BASE_URL}/jobs`;
    let result = fetch(contentURL,
                    {
                        credentials: "include",
                        method: "POST",
                        body:   payload
                    })
                        .then(response => response.json())
                            .catch((e) => {
                                //handle any error
                                console.log("An error occoured: " + e);
                        });
    return result;
}
```

Extended API sample in JavaScript

Here is an extended API sample that is written using JavaScript™ with features from ES62015 (map).

```
////////////////////////////////////////////////////////////////////////////
// Extended API Sample
// This Sample is written using Javascript with features from ES62015 (map).
// The sample is also written using JSX giving the ability to return HTML
 elements
// with Javascript variables embedded. This sample is based upon the codebase
 of the
// sample UI (see- hostname:port/ui) which is written using Facebook's React,
 Redux,
// Router and Google's material-ui
////////////////////////////////////////////////////////////////////////////

// Return a table with rows detailing the name and jobID of all jobs matching

// the specified parameters
function displayJobNamesTable(){
    let jobsJSON = getJobs("*","IBMUSER");
    return  (<table>
                {jobsJSON.map(job => {
                    return <tr><td>{job.name}</td><td>{job.id}</td></tr>
                })}
            </table>);
}

// Call the jobs GET api to get all jobs with the userID IBMUSER
function getJobs(owner, prefix){
    const BASE_URL = 'hostname.com:port/Atlas/api';
    let parameters = "prefix=" + prefix + "&owner=" + owner;
    let contentURL = `${BASE_URL}/jobs?${parameters}`;
    let result = fetch(contentURL, {credentials: "include"})

                    .then(response => response.json())

                        .catch((e) => {
                            //handle any error
                            console.log("An error occoured: " + e);
```

```
                        });
    return result;
}
```

Using explorer server WebSocket services

The explorer server provides WebSocket services that can be accessed by using the WSS scheme. With explorer server WebSocket services, you can view the system log in the System log UI that is refreshed automatically when messages are written. You can also open a JES spool file for an active job and view its contents that refresh through a web socket.

Server Endpoint	Description	Prerequisites
/api/sockets/syslog	Get current syslog content. Use this WSS endpoint to read the system log in real time.	SDSF
/api/sockets/jobs/{jobname}/ids/{jobid}/files/{fileid}	Tail the output of an active job. Use this WSS endpoint to read the tail of an active job's output file in real time.	z/OSMF restjobs

Programming explorer server WebSocket services

Here is code sample written in JavaScript™ using features from ES6 to establish a new WebSocket connection to listen to the syslog output.

```
const BASE_WS_URL = 'hostname.com:port/Atlas/api/sockets';

// Establish a new WebSocket connection to listen to the syslog output
function initWebsocket(){
    const syslogURI = `${BASE_WS_URL}/syslog`;

    this.websocket = new WebSocket(syslogURI);
    this.websocket.onopen = function() {
        //handle socket opening
        console.log("Websocket connection opened");
    }
    this.websocket.onmessage = function(event) {
        //handle receiving of new data
        console.log(event.data);
    }
    this.websocket.onclose = function() {
        //handle socket closing
        console.log("Websocket connection closed");
    }
}
```

Using Zoe Brightside

This section contains the following articles about using Zoe Brightside:

- How to display Zoe Brightside help
- Zoe Brightside command groups

Displaying Zoe Brightside help

Zoe Brightside contains a help system that is embedded directly into the command-line interface. When you want help with Zoe Brightside, you issue help commands that provide you with information about the product, syntax, and usage.

Displaying top-level help

To begin using the product, open a command line window and issue the following command to view the top-level help descriptions:

```
bright --help
```

Tip: The command `bright` initiates the product on a command line. All Zoe Brightside commands begin with `bright`.

Help structure

The help displays the following types of information:

- **Description:** An explanation of the functionality for the command group, action, or option that you specified in a `--help` command.
- **Usage:** The syntax for the command. Refer to usage to determine the expected hierarchical structure of a command.
- **Options:** Flags that you can append to the end of a command to specify particular values or booleans. For example, the volume size for a data set that you want to create.
- **Global Options:** Flags that you can append to any command in Zoe Brightside. For example, the `--help` flag is a global option.

Displaying command group, action, and object help

You can use the `--help` global option get more information about a specific command group, action, or object. Use the following syntax to display group-level help and learn more about specific command groups (for example, *zos-jobs* and *zos-files*):

```
bright <group, action, or object name> --help
```

```
bright zos-files create --help
```

More Information:

- Command Groups

Zoe Brightside command groups

Zoe Brightside contains command groups that focus on specific business processes. For example, the `zos-files` command group provides the ability to interact with mainframe data sets. In this article, we review all the Zoe Brightside command groups and provide you with a brief synopsis of the tasks that you can perform with each group. For more information, see Display Zoe Brigthside Help.

The commands available in the product are organized in a hierarchical structure. Command groups (for example, `zos-files`) contain actions (for example, `create`) that let you perform actions on specific objects (for example, a specific type of data set). For each action that you perform on an object, you can specify options that affect the operation of the command.

Important! Before you issue these commands, verify that you completed the steps in Create a Zoe Brightside profile and Test Connection to z/OSMF to help ensure that Zoe Brightside can communicate with z/OS systems.

Zoe Brightside contains the following command groups:

plugins

The plugins command group lets you install and manage third-party plug-ins for the product. Plug-ins extend the functionality of Zoe Brightside in the form of new commands.

With the plugins command group, you can perform the following tasks:

- Install or uninstall third-party plug-ins.

- Display a list of installed plug-ins.
- Validate that a plug-in integrates with the base product properly.

Note: For more information about `plugins` syntax, actions, and options, open Zoe Brightside and issue the following command:

```
bright plugins -h
```

profiles

The profiles command group lets you create and manage profiles for use with other Zoe Brightside command groups. Profiles allow you to issue commands to different mainframe systems quickly, without specifying your connection details with every command.

With the profiles command group, you can perform the following tasks:

- Create, update, and delete profiles for any Zoe Brightside command group that supports profiles.
- Set the default profile to be used within any command group.
- List profile names and details for any command group, including the default active profile.

Note: For more information about `profiles` syntax, actions, and options, open Zoe Brightside, and issue the following command:

```
bright profiles -h
```

provisioning

The provisioning command group lets you perform IBM z/OSMF provisioning tasks with templates and provisioned instances from Zoe Brightside.

With the provisioning command group, you can perform the following tasks:

- Provision cloud instances using z/OSMF Software Services templates.
- List information about the available z/OSMF Service Catalog published templates and the templates that you used to publish cloud instances.
- List summary information about the templates that you used to provision cloud instances. You can filter the information by application (for example, DB2 and CICS) and by the external name of the provisioned instances.
- List detail information about the variables used (and their corresponding values) on named, published cloud instances.

Note: For more information about provisioning syntax, actions, and options, open Zoe Brightside and issue the following command:

```
bright provisioning -h
```

zos-console

The zos-console command group lets you issue commands to the z/OS console by establishing an extended Multiple Console Support (MCS) console.

With the zos-console command group, you can perform the following tasks: **Important!** Before you issue z/OS console commands with Zoe Brightside, security administrators should ensure that they provide access to commands that are appropriate for your organization. - Issue commands to the z/OS console. - Collect command responses and continue to collect solicited command responses on-demand.

Note: For more information about `zos-console` syntax, actions, and options, open Zoe Brightside and issue the following command:

```
bright zos-console -h
```

zos-files

The zos-files command group lets you interact with data sets on z/OS systems.

With the zos-files command group, you can perform the following tasks:

- Create partitioned data sets (PDS) with members, physical sequential data sets (PS), and other types of data sets from templates. You can specify options to customize the data sets you create.
- Download mainframe data sets and edit them locally in your preferred Integrated Development Environment (IDE).
- Upload local files to mainframe data sets.
- List available mainframe data sets.
- Interact with VSAM data sets directly, or invoke Access Methods Services (IDCAMS) to work with VSAM data sets.

Note: For more information about `zos-console` syntax, actions, and options, open Zoe Brightside and issue the following command:

```
bright zos-files -h
```

zos-jobs

The zos-jobs command group lets you submit jobs and interact with jobs on z/OS systems.

With the zos-jobs command group, you can perform the following tasks:

- Submit jobs from JCL that resides on the mainframe or a local file.
- List jobs and spool files for a job.
- View the status of a job or view a spool file from a job.

Note: For more information about `zos-jobs` syntax, actions, and options, open Zoe Brightside and issue the following command:

```
bright zos-jobs -h
```

zos-tso

The zos-tso command group lets you issue TSO commands and interact with TSO address spaces on z/OS systems.

With the zos-tso command group, you can perform the following tasks:

- Excecute REXX scripts
- Create a TSO address space and issue TSO commands to the address space.
- Review TSO command response data in Zoe Brightside.

Note: For more information about `zos-tso` syntax, actions, and options, open Zoe Brightside and issue the following command:

```
bright zos-tso -h
```

zosmf

The zosmf command group lets you work with Zoe Brightside profiles and get general information about z/OSMF.

With the zosmf command group, you can perform the following tasks:

- Create and manage your Zoe Brightside zosmf profiles. You must have at least one zosmf profile to issue most commands. Issue the `bright help explain profiles` command in Zoe Brightside to learn more about using profiles.
- Verify that your profiles are set up correctly to communicate with z/OSMF on your system. For more information, see Test Connection to z/OSMF.
- Get information about the current z/OSMF version, host, port, and plug-ins installed on your system.

Note: For more information about `zosmf` syntax, actions, and options, open Zoe Brightside and issue the following command:

```
bright zosmf -h
```

More Information:

- Display Zoe Brightside help

Chapter 4. Extending Project Zoe

You can extend Project Zoe by integrating applications, plug-ins, or functionalities.

- Extending zLUX
- Extending Zoe Brightside

Extending zLUX

You can create plug-ins to extend the capabilities of zLUX.

Creating zLUX application plug-ins

A zLUX application plug-in is an installable set of files that present resources in a web-based user interface, as a set of RESTful services, or in a web-based user interface and as a set of RESTful services.

Before you build a zLUX application plug-in, you must set the UNIX environment variables that support the plug-in environment.

`sample-app` is a sample application plug-in with which you can experiment.

Setting the environment variables for plug-in development

To set up the environment, the node must be accessible on the PATH. To determine if the node is already on the PATH, issue the following command from the command line:

```
node --version
```

If the version is returned, the node is already on the PATH.

If nothing is returned from the command, you can set the PATH using the *NODE_HOME* variable. The *NODE_HOME* variable must be set to the directory of the node install. You can use the export command to set the directory. For example:

```
export NODE_HOME=node_installation_directory
```

Using this directory, the node will be included on the PATH in `nodeServer.sh`. (`nodeServer.sh` is located in `zlux-example-server/bin`).

Using the zLUX sample application plug-in

Your zLUX installation provides a sample application plug-in with which you can experiment.

To build the sample application plug-in, node and npm must be included in the PATH. You can use the `npm run build` or `npm start` command to build the sample application plug-in. These commands are configured in `package.json`.

Note:

- If you change the source code for the sample application, you must rebuild it.
- If you want to modify `sample-app`, you must run _`npm install`_ in the virtual desktop and the `sample-app/webClient`. Then, you can run _`npm run build`_ in `sample-app/webClient`.

1. Add an item to `sample-app`. The following figure shows an excerpt from `app.component.ts`:

   ```
   export class AppComponent { items = ['a', 'b', 'c', 'd'] title = 'app';
   helloText: string; serverResponseMessage: string;
   ```

2. Save the changes to `app.component.ts`.
3. Issue one of the following commands:

- To rebuild the application plug-in, issue the following command: `npm run build`
- To rebuild the application plug-in and wait for additional changes to `app.component.ts`, issue the following command: `npm start`

4. Reload the web page.
5. If you make changes to the sample application source code, follow these steps to rebuild the application:
   a. Navigate to the `sample-app` subdirectory where you made the source code changes.
   b. Issue the following command: `npm run build`
   c. Reload the web page.

zLUX plug-ins definition and structure

The zLUX Application Server `zlux-proxy-server` enables extensiblity with application plug-ins. Application plug-ins are a sub-category of the unit of extensibility in the server called a *plug-in*.

The files that define a plug-in are located in the `pluginsDir` directory.

zLUX application plug-in filesystem structure

A zLUX application plug-in can be loaded from a filesystem that is accessible to the zLUX Application Server, or dynamically at runtime. When accessed from a filesystem, there are important considerations for both the developer and the end-user as to where files must be placed for proper build, packaging, and operation.

Root files and directories

The root of an application plug-in directory contains the following files and directories.

pluginDefinition.json

This file describes an application plug-in to the zLUX Application Server. (A plug-in is the unit of extensibility for the zLUX Application server. An application plug-in is a plugin of the type "Application", the most common and visible type of plug-in). A definition file informs the server whether the application plug-in has server-side Dataservices, client-side web content, or both.

Dev and Source content

Aside from demonstration or open source application plug-ins, the following directories should not be seen on a deployed server (the directories are used to build content and are not read by the server).

nodeServer

When an application plug-in has Dataservices of the type "router", they are interpreted by the zLUX Application Server by attaching them as ExpressJS routers. Ir is recommended that you write application plug-ins using Typescript, because it leads to more well-structured code than Javascript. Use of Typescript results in build steps because the pre-transpilation Typescript content is not to be consumed by NodeJS. Therefore, within nodeServer, you keep your server-side source code. At runtime, the server loads router dataservices from the `lib` directory.

webClient

When an application plug-in has the webContent attribute in its Definition, the server will serve static content for a client. To optimize loading of the application plug-in to the user, use Typescript to write the application plug-in and package it using Webpack. Use of Typescript and Webpack result in build steps as the pre-transpilation Typescript and the pre-webpack content are not to be consumed by the browser. Therefore, the source code should be separated from the served content by placing source code within webClient.

Runtime content

At runtime, a different set of directories are used by the server and client rather than those described for use in the dev environment.

lib

The `lib` directory is where router-type Dataservices are loaded by use in the zLUX Application Server. If the JS files that are loaded from the `lib` directory require NodeJS modules, which are not provided by the server base (modules zlux-proxy-server requires are added to `NODE_PATH` at runtime), then these modules must be included in lib/node_modules for local directory lookup or be found on the `NODE_PATH` environment variable. `nodeServer/node_modules` is not automatically accessed at runtime as it is a dev and build directory.

web

The `web` directory is where the server will serve static content for an application plug-in that includes the **webContent** attribute in its Definition. Typically this directory contains the output of a webpack build. Anything put in this folder can be accessed by a client, so only include content in this location that is intended to be consumed by clients.

Location of plug-in files

The files that define a plug-in are located in the `pluginsDir` directory.

pluginsDir directory

At start up, the server reads from the `pluginsDir` directory. The server loads the valid plug-ins that are found by the information that is provided in the JSON files.

Within the `pluginsDir` directory are a collection of JSON files. Each file has two attributes, which serve to locate a plug-in on disk:

location: This is a directory path that is relative to the server's executable (such as `zlux-example-server/bin/nodeServer.sh`) at which a `pluginDefinition.json` file is expected to be found.

identifier: The unique string (commonly styled as a Java resource) of a plug-in, which must match what is in the `pluginDefinition.json` file.

Plug-in definition file

`pluginDefinition.json` is a file that describes a plug-in. Each zLUX plug-in requires this file, because it defines how the server will register and use the backend of an application plug-in, (called a *plug-in* in the terminology of the proxy server). The attributes in each file are dependent upon the **pluginType** attribute. Consider the following `pluginDefinition.json` file from `sample-app`:

```
{
  "identifier": "com.rs.mvd.myplugin",
  "apiVersion": "1.0",
  "pluginVersion": "1.0",
  "pluginType": "application",
  "webContent": {
    "framework": "angular2",
    "launchDefinition": {
      "pluginShortNameKey": "helloWorldTitle",
      "pluginShortNameDefault": "Hello World",
      "imageSrc": "assets/icon.png"
    },
    "descriptionKey": "MyPluginDescription",
    "descriptionDefault": "Base MVD plugin template",
    "isSingleWindowApp": true,
    "defaultWindowStyle": {
      "width": 400,
      "height": 300
    }
  },
  "dataServices": [
    {
      "type": "router",
      "name": "hello",
      "serviceLookupMethod": "external",
```

```
      "fileName": "helloWorld.js",
      "routerFactory": "helloWorldRouter",
      "dependenciesIncluded": true
    }
  ]
}
```

Attributes

There are two categories of attributes: General and Application.

General attributes

identifier - Every application plug-in must have a unique string ID that associates it with a URL space on the server.

apiVersion - The version number for the pluginDefinition scheme and application plug-in or Dataservice requirements. The default is 1.0.0.

pluginVersion - The version number of the individual plug-in.

pluginType - A string that specifies the type of plug-in. The type of plug-in determines the other attributes that are valid in the definition.

- **application** - Defines the plug-in as an application plug-in. Application plug-ins are composed of a collection of web content for presentation in the zLUX web component (such as the virtual desktop), or a collection of dataservices (REST and websocket), or both.
- **library** - Defines the plug-in as a library that serves static content at a known URL space.
- **node authentication** - Authentication and Authorization handlers for the zLUX Application Server.

Application attributes

When a plug-in is of **pluginType** application, the following attributes are valid:

- **webContent** - An object that defines a few attributes about the content that is shown in a web UI.
- **dataServices** - An array of objects that describe REST or websocket dataservices.
- **configurationData** - An object that describes the resource structure that the application plug-in uses for storing user, group, and server data.

Application web content attributes

An application that has the **webContent** attribute defined provides content that is displayed in a zLUX web UI.

The following attributes determine some of this behavior:

- **framework** - States what type of web framework is used, which determines the other attributes that are valid in webContent.
- **angular2** - Defines the application as having an Angular (2+) web framework component. This is the standard for a "native" framework zLUX application.
- **iframe** - Defines the application as being external to the native zLUX web application environment, but instead embedded in an iframe wrapper.
- **launchDefinition** - An object that details some attributes for presenting the application in a web UI.
  - **pluginShortNameDefault** - A string that gives a name to the application when i18n is not present. When i18n is present, i18n is applied by using the pluginShortNameKey.
  - **descriptionDefault** - A longer string that specifies a description of the application within a UI. The description is seen when i18n is not present. When i18n is present, i18n is applied by using the descriptionKey.
  - **imageSrc** - The relative path (from `/web`) to a small image file that represents the application icon.
- **defaultWindowStyle** - An object that details the placement of a default window for the application in a web UI.

- **width** - The default width of the application plug-in window, in pixels.
- **height** - The default height of the application plug-in window, in pixels.

IFrame application web content

In addition to the general web content attributes, when stating that the framework of an application is "iframe", you must state the page that is being embedded in the iframe. To do so, incude the attribute **startingPage** within **webContent**. **startingPage** is relative to the application's `/web` directory.

It is recommended that `startingPage` be a relative path rather than an absolute path as the `pluginDefinition.json` file is intended to be read-only, and therefore would not work well when the hostname of a page has changed.

Within an IFrame, the application still has access to the globals that are used by zLUX for application-to-application communication: simply access **window.parent.RocketMVD**.

zLUX application filesystem structure

A zLUX application can be loaded from a filesystem accessible to the zLUX application server, or dynamically at runtime.

When accessed from a filesystem, there are important considerations for both the developer and the user as to where files must be placed for proper build, packaging, and operation.

Root files and directories

At the root of an application's directory, the following content is found:

pluginDefinition.json

This file describes a plug-in to the zLUX Application Server. A plug-in is the unit of extensibility for the server, where an application is a plug-in of the type Application, the most common and visible plug-in type.

A definition file informs the server whether the application has server-side, client-side web content, or both. The attributes of this file and how it is found by the server are described in zLUX plug-in definition and structure.

Dev and Source content

Aside from demonstration or open source applications, the following directories should not be seen on a deployed server. The directories are not read by the server, but they are used to build content that the server can read.

nodeServer

When an application has Dataservices of the type "router", they are interpreted by the zLUX Application Server by attaching them as ExpressJS routers. It is strongly suggested that applications be written in Typescript as it leads to more well-structured code. Use of Typescript results in build steps because the pre-transpilation Typescript content is not to be consumed by NodeJS. Therefore, keep your server-side source code within **nodeServer**. At runtime, the server loads **router dataservices** from the `lib` directory.

webClient

When an application has the webContent attribute in its Definition, the server serves static content for a client. It is strongly suggested that applications be written in Typescript and packaged through Webpack to optimize loading of the application to the user. Use of Typescript and Webpack result in build steps as the pre-transpilation Typescript and the pre-webpack content are not to be consumed by the browser. Therefore, separate the source code from the served content by placing source code within webClient.

Runtime content

At runtime, a different set of directories are used by the server and client rather than those described for use in the development environment.

lib

The `lib` directory is where router-type Dataservices are loaded by use in the zLUX Application Server. If the JS files that are loaded from the `lib` directory require NodeJS modules, which are not provided by the server base (modules zlux-proxy-server requires are added to `NODE_PATH` at runtime), then these modules must be included in `lib/node_modules` for local directory lookup or be found on the `NODE_PATH` environment variable. `nodeServer/node_modules` is not automatically accessed at runtime as it is a dev and build directory.

web

The `web` directory is where the server serves static content for an application that has included the webContent attribute in its Definition. Typically this directory contains the output of a webpack build. Anything you place in this folder can be accessed by a client. Therefore, only place content in this folder that is intended to be consumed by clients.

Configuration Dataservice

The Configuration Dataservice is an essential component of the zLUX framework, which acts as a JSON resource storage service, and is accessible externally by REST API and internally to the server by Dataservices.

The Configuration Dataservice allows for saving preferences of applications, management of defaults and privileges within a zLUX ecosystem, and bootstrapping configuration of the server's dataservices.

The fundamental element of extensibility of the zLUX framework is a plug-in. The Configuration Dataservice works with data for plug-ins. Every resource that is stored in the Configuration Service is stored for a particular plug-in, and valid resources to be accessed are determined by the definition of each plug-in in how it uses the Configuration Dataservice.

The behavior of the Configuration Dataservice is dependent upon the Resource structure for a zLUX plug-in. Each plug-in lists the valid resources, and the administrators can set permissions for the users who can view or modify these resources.

Resource Scope

Data is stored within the Configuration Dataservice according to the Scope chosen.The intent of Scope within the Dataservice is to facilitate company-wide administration and privilege management of zLUX data.

When a user requests a resource, the resource that is retrieved is override or an aggregation of the broader Scopes that encompass the Scope they are viewing the data from.

When a user stores an resource, the resource is stored within a Scope but only if the user has access privilege to update within that Scope.

Scope is one of the following:

- **Product**: Configuration defaults that come with the product. Cannot be modified.
- **Site**: Data that can be used between multiple instances of the zLUX server.
- **Instance**: Data within an individual zLUX server.
- **Group**: Data shared between multiple users in a group (**Pending**)
- **User**: Data for an individual user (**Pending**)

Note: While Authorization tuning can allow for settings such as GET from Instance to work without login, User and Group scope queries will be rejected if not logged in due to the requirement to pull resources from a specific user. Because of this, User and Group Scopes will not be functional until the Security Framework is available.

Where **Product** is the broadest scope and **User** is the narrowest scope.

When using scope **User**, the service will manage configuration for your particular username, using the authentication of the session. This way, the **User** scope is always mapped to your current username.

Consider a case where a user wants to access preferences for their text editor. One way they could do this is to use the REST API to retrieve the settings resource from the **Instance** scope.

The **Instance** scope might contain editor defaults that their administrator set. But, if there were no defaults in Instance, then the data in **Group** and finally **User** would be checked.

Therefore, the data the user receives would be no broader than what is stored in the **Instance** scope, but may have only been the settings they had saved within their own **User** scope if the broader scopes had no data for the resource.

Later, perhaps the user wants to save changes, and they try to save in the **Instance** scope. Most likely, this is rejected due to preferences by the administrator to disallow changes to the **Instance** scope by ordinary users.

REST API

When reaching the Configuration Service through a REST API, HTTP methods are used to perform the desired operation.

The HTTP URL scheme for the configuration dataservice is:

<Server>/plugins/com.rs.configjs/services/data/<plugin ID>/<Scope>/<resource>/<optional subresources>?<query>

Where the resources are one or more levels deep, using as many layers of subresources as needed.

You can think of a resource as a collection of elements, or a directory. To access a single element, you must use the query parameter "name="

REST query parameters

- **Name** (string) - Get or put a single element rather than a collection.
- **Recursive** (boolean) - When performing a DELETE, specifies whether to delete subresources.

REST HTTP methods

Below is an explanation of each type of REST call. Each API call includes an example request and response against a hypothetical application: the "code editor".

GET

GET /plugins/com.rs.configjs/services/data/<plugin>/<scope>/<resource>?name=<element>

- This returns JSON with the attribute "content" being a JSON resource that is the entire configuration requested.
- Example:

`/plugins/com.rs.configjs/services/data/org.openmainframe.zoe.codeeditor/user/sessions/default?name=tabs`

- The parts of the URL are:
  - Plugin: org.openmainframe.zoe.codeeditor
  - Scope: user
  - Resource: sessions
  - Subresource: default
  - Element = tabs
- Response body is a JSON config:

```
{
"_objectType" : "com.rs.config.resource",
"_metadataVersion" : "1.1",
"resource" : "org.openmainframe.zoe.codeeditor/USER/sessions/default",
"contents" : {
```

```
  "_metadataVersion" : "1.1",
  "_objectType" : "org.openmainframe.zoe.codeeditor.sessions.tabs",
  "tabs" : [{
    "title" : "TSSPG.REXX.EXEC(ARCTEST2)",
    "filePath" : "TSSPG.REXX.EXEC(ARCTEST2)",
    "isDataset" : true
   }, {
    "title" : ".profile",
    "filePath" : "/u/tsspg/.profile"
   }
  ]
 }
 }
```

GET /plugins/com.rs.configjs/services/data/<plugin>/<scope>/<resource>

- This returns JSON with the attribute "content" being a JSON object that has each attribute being another JSON object which is a single configuration element.

GET /plugins/com.rs.configjs/services/data/<plugin>/<scope>/<resource> when subresources exist

- This returns a listing of subresources that can, in turn, be queried.

PUT

PUT /plugins/com.rs.configjs/services/data/<plugin>/<scope>/<resource>?name=<element>

- Stores a single element (must be a JSON object {...}) within the requested scope, ignoring aggregation policies, provided the privilege of the user permits this.
- Example: `/plugins/com.rs.configjs/services/data/org.openmainframe.zoe.codeeditor/user/sessions/default?name=tabs`
- Body:

```
{
  "_metadataVersion" : "1.1",
  "_objectType" : "org.openmainframe.zoe.codeeditor.sessions.tabs",
  "tabs" : [{
      "title" : ".profile",
      "filePath" : "/u/tsspg/.profile"
    }, {
      "title" : "TSSPG.REXX.EXEC(ARCTEST2)",
      "filePath" : "TSSPG.REXX.EXEC(ARCTEST2)",
      "isDataset" : true
    }, {
      "title" : ".emacs",
      "filePath" : "/u/tsspg/.emacs"
    }
  ]
}
```

- Response:

```
{
  "_objectType" : "com.rs.config.resourceUpdate",
  "_metadataVersion" : "1.1",
  "resource" : "org.openmainframe.zoe.codeeditor/USER/sessions/default",
  "result" : "Replaced item."
}
```

DELETE

DELETE /plugins/com.rs.configjs/services/data/<plugin>/<scope>/<resource>?recursive=true

- Deletes all files in all leaf resources below the resource specified.

DELETE /plugins/com.rs.configjs/services/data/<plugin>/<scope>/<resource>?name=<element>

- Deletes a single file in a leaf resource.

DELETE /plugins/com.rs.configjs/services/data/<plugin>/<scope>/<resource>

- Deletes all files in a leaf resource.
- Does not delete the folder on disk.

Administrative access and group

By means not discussed here, but instead handled by the server's authentication and authorization code, a user might be privileged to access or modify items that they do not own.

In the simplest case, it might mean that the user is able to do a PUT, POST, or DELETE to a level above User, such as Instance.

The more interesting case is accessing another user's contents. In this case, the shape of the URL is different. Compare the following two commands:

GET /plugins/com.rs.configjs/services/data/<plugin>/user/<resource>

- Gets the content for the current user

GET /plugins/com.rs.configjs/services/data/<plugin>/users/<username>/<resource>

- Gets the content for a specific user if authorized

This is the same structure that is (or will be) used for the Group scope. When requesting content from the Group scope, the user is checked to see if they are authorized to make the request for the specific group. For example:

GET /plugins/com.rs.configjs/services/data/<plugin>/group/<groupname>/<resource>

- Gets the content for the given group, if the user is authorized for it.

Application API

Retrieves and stores configuration information from specific scopes.

NOTE: This API should only be used for configuration administration user interfaces.

```
A shortcut for the preceding method, and the preferred method when you are
 retrieving configuration information is simply to "consume" it. It "asks"
 for configurations using the "user" scope, and lets the configuration
 service decide which configuration information to retrieve and how to
 aggregate it. (See below on how the configuration service evaluates what to
 return for this type of request).
```

Internal and bootstrapping

Some Dataservices within plug-ins can take configuration that affects their behavior. This configuration is stored within the Configuration Dataservice structure, but is not accessible through the REST API.

Within the deploy directory of a zLUX installation, each plug-in may optionally have an `_internal` directory. An example of such a path is:

`deploy/instance/ZLUX/pluginStorage/<pluginName>/_internal`

Within each _internal folder, the following folders may exist: * `services/<servicename>`: Configuration resources for the specific service. * plugin: Configuration resources visible to all services in the plug-in.

The JSON contents within these directories will be provided as Objects to dataservices through the dataservice context Object.

Plug-in Definition

Because the Configuration Dataservices stores data on a per-plug-in basis, each zLUX plug-in must define their resource structure to make use of the Configuration Dataservice. The resource structure definition is included in the plug-in's `pluginDefinition.json` file.

For each resource and subresource, you can define an aggregationPolicy to control how the data of a broader Scope alters the resource data that is returned to a user when requesting a resource from a narrower Scope.

Example:

```
  "configurationData": { //is a direct attribute of the pluginDefinition
 JSON
    "resources": { //always required
      "preferences": {
        "locationType": "relative", //this is the only option for now, but
 later absolute paths may be accepted
        "aggregationPolicy": "override" //override and none for now, but more
 in the future
      },
      "sessions": { //the name at this level represents the name
 used within a URL, such as /plugins/com.rs.configjs/services/data/
org.openmainframe.zoe.codeeditor/user/sessions
        "aggregationPolicy": "none",
        "subResources": {
          "sessionName": {
            "variable": true, //if variable=true is present, the resource
 must be the only one in that group but the name of the resource is
 substituted for the name given in the REST request, so it represents more
 than one
            "aggregationPolicy": "none"
          }
        }
      }
    }
  }
```

Aggregation Policies

Aggregation Policies determine how the Configuration Dataservice aggregates JSON objects from different Scopes together when a user requests a resource. If the user requests a resource from the User scope, the data from the User scope may replace, or be merged with the data from a broader scope such as Instance, to make a combined resource object that is returned to the user.

Aggregation Policies are defined by a plug-in developer in the plug-in's definition for the Configuration Service, as the attribute "aggregationPolicy" within a resource.

The following policies are currently implemented:

- **NONE** - If the Configuration Dataservice is called for Scope User, only user-saved settings are sent, unless there are no user-saved settings for the query, in which case the Dataservice attempts to send data found at a more broad scope.
- **OVERRIDE** - The Configuration Dataservice obtains data for the resource that is requested at the broadest level found, and joins together the resource's properties from more narrow scopes, overriding broader attributes with narrower ones, when found.

Extending Zoe Brightside

You can install plug-ins to extend the capabilities of Zoe Brightside. Plug-ins add functionality to the product in the form of new command groups, actions, objects, and options.

Important! Plug-ins can gain control of your CLI application legitimately during the execution of every command. Install third-party plug-ins at your own risk. We make no warranties regarding the use of third-party plug-ins.

The following plug-ins are available:

Note: For information about how to install, update, and validate a plug-in, see Installing Plug-ins.

Zoe Brightside plug-in for IBM Db2 Database

The Zoe Brightside plug-in for Db2 enables you to interact with IBM Db2 Database on z/OS to perform tasks with modern development tools to automate typical workloads more efficiently. The plug-in also enables you to interact with IBM Db2 to foster continuous integration to validate product quality and stability.

For more information, see Zoe Brightside plug-in for IBM Db2 Database.

Installing plug-ins

Use commands in the plugins command group to install and manage plug-ins for Zoe Brightside.

Important! Plug-ins can gain control of your CLI application legitimately during the execution of every command. Install third-party plug-ins at your own risk. We make no warranties regarding the use of third-party plug-ins.

You can install the following plug-ins:

- **IBM Db2 Database** Use `@brightside/db2` in your command syntax to install, update, and validate the IBM Db2 Database plug-in.

Setting the registry

If you installed Zoe Brightside from the brightside-bundle.zip distributed with the Zoe PAX media, proceed to the Install step.

If you installed Zoe Brightside from a registry, confirm that NPM is set to target the registry by issuing the following command:

```
npm config set @brightside:registry https://api.bintray.com/npm/ca/brightside
```

Meeting the prerequisites

Ensure that you meet the prerequisites for a plug-in before you install the plug-in to Zoe Brightside. For documentation related to each plug-in, see Extending Zoe Brightside.

Installing plug-ins

Issue an `install` command to install plug-ins to Zoe Brightside. The `install` command contains the following syntax:

```
bright plugins install [plugin...] [--registry <registry>]
```

- **`[plugin...]`** (Optional) Specifies the name of a plug-in, an npm package, or a pointer to a (local or remote) URL. When you do not specify a plug-in version, the command installs the latest plug-in version and specifies the prefix that is stored in npm save-prefix. For more information, see npm save prefix. For more information about npm semantic versioning, see npm semver. Optionally, you can specify a specific version of a plug-in to install. For example, `bright plugin install pluginName@^1.0.0`.

Tip: You can install multiple plug-ins with one command. For example, issue `bright plugin install plugin1 plugin2 plugin3`

- **`[--registry <registry>]`** (Optional) Specifies a registry URL from which to install a plug-in when you do not use `npm config set` to set the registry initially.

Examples: Install plug-ins

- The following example illustrates the syntax to use to install a plug-in that is distributed with the Zoe brightside-bundle.zip. If you are using brightside-bundle.zip, issue the following command for each plug-in .tgz file:

```
bright plugins install ./brightside-db2-1.0.0.tgz
```

- The following example illustrates the syntax to use to install a plug-in that is named "my-plugin" from a specified registry:

```
bright plugins install @brightside/my-plugin
```

- The following example illustrates the syntax to use to install a specific version of "my-plugins"

```
 bright plugins install @brightside/my-plugin@"^1.2.3"
```

Validating plug-ins

Issue the plug-in validation command to run tests against all plug-ins (or against a plug-in that you specify) to verify that the plug-ins integrate properly with Zoe Brightside. The tests confirm that the plug-in does not conflict with existing command groups in the base application. The command response provides you with details or error messages about how the plug-ins integrate with Zoe Brightside.

Perform validation after you install the plug-ins to help ensure that it integrates with Zoe Brightside.

The `validate` command has the following syntax:

```
bright plugins validate [plugin]
```

- **`[plugin]`** (Optional) Specifies the name of the plug-in that you want to validate. If you do not specify a plug-in name, the command validates all installed plug-ins. The name of the plug-in is not always the same as the name of the NPM package.

Examples: Validate plug-ins

- The following example illustrates the syntax to use to validate a specified installed plug-in:

```
bright plugins validate @brightside/my-plugin
```

- The following example illustrates the syntax to use to validate all installed plug-ins:

```
bright plugins validate
```

Updating plug-ins

Issue the `update` command to install the latest version or a specific version of a plug-in that you installed previously. The `update` command has the following syntax:

```
bright plugins update [plugin...] [--registry <registry>]
```

- **`[plugin...]`**

  Specifies the name of an installed plug-in that you want to update. The name of the plug-in is not always the same as the name of the NPM package. You can use npm semantic versioning to specify a plug-in version to which to update. For more information, see npm semver.

- **[--registry <registry>]**

  (Optional) Specifies a registry URL that is different from the registry URL of the original installation.

Examples: Update plug-ins

- The following example illustrates the syntax to use to update an installed plug-in to the latest version:

```
bright plugins update @brightside/my-plugin@latest
```

- The following example illustrates the syntax to use to update a plug-in to a specific version:

```
bright plugins update @brightside/my-plugin@"^1.2.3"
```

Uninstalling plug-ins

Issue the `uninstall` command to uninstall plug-ins from a base application. After the uninstall process completes successfully, the product no longer contains the plug-in configuration.

Tip: The command is equivalent to using npm uninstall to uninstall a package.

The `uninstall` command contains the following syntax:

```
bright plugins uninstall [plugin]
```

- **[plugin]** Specifies the plug-in name to uninstall.

Example: Uninstall plug-ins

- The following example illustrates the syntax to use to uninstall a plug-in:

```
bright plugins uninstall @brightside/my-plugin
```

Zoe Brightside plug-in for IBM Db2 Database

The Zoe Brightside plug-in for IBM Db2 Database lets you interact with Db2 for z/OS to perform tasks with modern development tools to automate typical workloads more efficiently. The plug-in also enables you to interact with Db2 to advance continuous integration to validate product quality and stability.

Plug-in overview

Zoe Brightside Plug-in for IBM Db2 Database lets you execute SQL statements against a Db2 region, export a Db2 table, and call a stored procedure.The plug-in also exposes its API so that the plug-in can be used directly in other products.

Use cases

Example use cases for Zoe Brightside Db2 plug-in include: - Execute SQL and interact with databases - Execute a file with SQL statements - Export tables to a local file on your PC in SQL format - Call a stored procedure and pass parameters

Prerequisites

Ensure that Zoe Brightside is installed.

More Information:

- Installing Zoe Brightside

Installing

There are **two methods** that you can use to install the Zoe Brightside Plug-in for IBM Db2 Database.

Method 1

If you installed **Zoe Brightside** from **bintray**, complete the following steps:

1. Open a command line window and issue the following command:

```
bright plugins install @brightside/db2
```

2. After the command execution completes, issue the following command to validate that the installation completed successfully.

```
bright plugins validate db2
```

   Successful validation of the IBM Db2 plug-in returns the response: `Successfully validated`.

Method 2

If you downloaded the **Project Zoe** installation package from **Github**, complete the following steps:

1. Open a command line window and change the directory to the location where you extracted the `brightside-bundle.zip` file. If you do not have the `brightside-bundle.zip` file, see the topic **Install Zoe Brightside from local package** in Installing Zoe Brightside for information about how to obtain and extract it.
2. From the command line window, set the `IBM_DB_INSTALLER_URL` environment variable by issuing the following command:
   - Windows operating systems: `set IBM_DB_INSTALLER_URL=%cd%/odbc_cli`
   - Linux and Mac operating systems: `export IBM_DB_INSTALLER_URL=`pwd`/odbc_cli`
3. Issue the following command to install the plug-in:

```
bright plugins install brightside-db2-1.0.0.tgz
```

4. After the command execution completes, issue the following command to validate that the installation completed successfully.

```
bright plugins validate db2
```

   Successful validation of the IBM Db2 plug-in returns the response: `Successfully validated`.

Profile setup

Before you start using the IBM Db2 plug-in, create a profile.

Creating a profile

Issue the command `-DISPLAY DDF` in the SPUFI or ask your DBA for the following information:

- The Db2 server host name
- The Db2 server port number
- The database name (you can also use the location)
- The user name
- The password
- If your Db2 systems use a secure connection, you can also provide an SSL/TSL certificate file.

To create a db2 profile in Zoe Brightside, issue a command in the command shell in the following format:

```
bright profiles create db2 <profile name> -H <host> -P <port> -d <database> -
u <user> -p <password>
```

The profile is created successfully with the following output:

```
Profile created successfully! Path:
/home/user/.brightside/profiles/db2/<profile name>.yaml
type: db2
name: <profile name>
```

```
hostname: <host>
port: <port>
username: securely_stored
password: securely_stored
database: <database>
Review the created profile and edit if necessary using the profile update
 command.
```

Commands

The following commands can be issued with the Zoe Brightside Db2 plug-in.

Tip: At any point, you can issue the help command `-h` to see a list of available commands.

Calling a stored procedure

Issue the following command to call a stored procedure that returns a result set:

```
$ bright db2 call sp "DEMOUSER.EMPBYNO('000120')"
```

Issue the following command to call a stored procedure and pass parameters:

```
$ bright db2 call sp "DEMOUSER.SUM(40, 2, ?)" --parameters 0
```

Issue the following command to call a stored procedure and pass a placeholder buffer:

```
$ bright db2 call sp "DEMOUSER.TIME1(?)" --parameters "....placeholder..
```

Executing an SQL statement

Issue the following command to count rows in the EMP table:

```
$ bright db2 execute sql -q "SELECT COUNT(*) AS TOTAL FROM DSN81210.EMP;"
```

Issue the following command to get a department name by ID:

```
$ bright db2 execute sql -q "SELECT DEPTNAME FROM DSN81210.DEPT WHERE
 DEPTNO='D01'
```

Exporting a table in SQL format

Issue the following command to export the PROJ table and save the generated SQL statements:

```
$ bright db2 export table DSN81210.PROJ
```

Issue the following command to export the PROJ table and save the output to a file:

```
$ bright db2 export table DSN81210.PROJ --outfile projects-backup.sql
```

You can also pipe the output to gzip for on-the-fly compression.

Notices

IBM notices

This information was developed for products and services offered in the U.S.A.

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

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

IBM Director of Licensing IBM Corporation North Castle Drive Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing Legal and Intellectual Property Law IBM Japan, Ltd. 19-21, Nihonbashi-Hakozakicho, Chuo-ku Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Intellectual Property Dept. for Rational® Software IBM Corporation Silicon Valley Lab 555 Bailey Avnue San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be

the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

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

Copyright license

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

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 1992, 2018.

If you are viewing this information in softcopy, the photographs and color illustrations may not appear.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Terms and conditions for product documentation

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Rocket Software, Inc. notices

Copyright

© Rocket Software, Inc. or its affiliates 2013-2018. All Rights Reserved.

Trademarks

Rocket is a registered trademark of Rocket Software, Inc. For a list of Rocket registered trademarks go to: www.rocketsoftware.com/company/legal/terms-of-use. All other products or services mentioned in this document may be covered by the trademarks, service marks, or product names of their respective owners.

Examples

This information might contain examples of data and reports. The examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

License agreement

This software and the associated documentation are proprietary and confidential to Rocket Software, Inc. or its affiliates, are furnished under license, and may be used and copied only in accordance with the terms of such license.

Note: This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when exporting this product.

CA documentation legal notices

This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2018 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.