Project Giza Closed Beta

User's Guide

Edition notice

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

About this book

This book describes how to install, configure, and use Project Giza Closed Beta.

Who should read this book

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

To use this book, you must be familiar with Mainframe and z/OSMF configuration.

Terminology used in this book

Before getting started with Project Giza, please acquaint yourself with the following terms:

Term	Abbreviation	Definition
Mainframe Virtual Desktop	MVD	Virtual desktop, accessed through a web browser.
Atlas	N/A	A z/OS RESTful web service and deployment architecture for z/OS microservices.
Brightside Command Line Interface	Brightside CLI	Brightside CLI lets application developers interact with the mainframe in a format that is natively familiar to them. It lets application developers use common tools such as Integrated Development Environments (IDEs), shell commands, bash scripts, and build tools for mainframe development.
Command Groups	N/A	Brightside CLI contains command groups that focus on specific business processes that application developers and systems programmers perform during their day-to-day activities.
Experimental Commands	N/A	Brightside CLI includes experimental commands that are currently in development and are not ready for general availability. Users can enable or disable these commands. Experimental commands are disabled by default.
z/OS Lightweight User Experience	zLUX	Framework, MVD, plugin applications, TN3270 emulator, and Discovery.

Term	Abbreviation	Definition
ZLUX Secure Services address space	ZSS	The server that provides secure REST services to support the Giza Node Server.
Giza Node Server	N/A	Node.js server plus Express.js as webservices framework, and the proxy applications that communicate with the z/OS services and components.
TN3270	N/A	A limited license Giza plugin that provides a 3270 connection to the mainframe on which the Giza Node Server is running.
Application	N/A	A plug-in that is an application.
Discovery	N/A	z/OS subsystems plug-in.

How to use this book

This book contains an introduction and information for installing, configuring, and using Project Giza.

- Introducing Project Giza explains what is Project Giza, how it works, and what it can do.
- Installing Project Giza explains how to install, configure, and maintain Project Giza so that it can work. It also provides information about troubleshooting installation related problems.
- Using Project Giza explains how to use the features that Project Giza provides.

How to send your feedback

Your feedback is important in helping us to provide accurate and high-quality information. If you have comments about this documentation, raise an issue and send us a pull request.

Be sure to include the version of the product, and, if applicable, the specific location (for example, the page number or section heading) of the text that you are commenting on.

Chapter 1. Project Giza overview

Project Giza delivers modern interfaces to z/OS through RESTful services, a command line interface, and a web-based interactive environment. These modern interfaces are designed for z/OS application developers and system programmers to increase productivity and provide an agile environment. You can use these interfaces as delivered or through programmable extensions that are created by clients or third-party vendors.

Parent topic: Introducing Project Giza

What is Project Giza

Today's information technology (IT) organization must be both innovative and agile, which comes from the creation of new solutions using various building blocks of computing capabilities. z/OS-based resources represent a huge existing asset to companies and are one of the key building blocks for customer solutions. Newer cloud-based services are another important building block. The challenge is how best to mix the established z/OS computing resources with cloud-based services to facilitate the needed innovation.

Cloud services have created a set of interfaces and provided common work patterns that are used by today's programmers and system administrators. Project Giza provides interfaces on z/OS that are intended to be identical to what programmers and system administrators would experience on cloud platforms today. Project Giza also provides ways for individuals to extend the work patterns that they use today on cloud to the z/OS environment. The goal is drive innovation through the integration of z/OS-based services and cloud services.

Project Giza is focused on the integration of z/OS into the wider enterprise. The purpose is to allow IT staff to work with the mainframe as they would in any other cloud-based environment, so that the staff with typical understanding of cloud interfaces can use z/OS. The more the staff are able to use the platform, the faster the needs of the enterprise can be met.

Project Giza contains a desktop, browser-based user interface (UI) that provides a full screen interactive experience. The web UI 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.

Project Giza enables you to access z/OS data, jobs, and subsystems through REST APIs. 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.

Project Giza provides a command line interface that allows interactive access to those same data, jobs, and subsystem, but extends the capability in additional ways. You can use the command line interface to perform the following tasks:

- Edit files, submit jobs, and issue commands which are the common z/OS tasks.
- Script the commands to perform a series of steps. You can write scripts to automate a variety of z/OS tasks to speed such things as application deployment, provisioning of new run-time environments or job submission and capturing of results.

- Use new ways to work with z/OS assets such as using file editors of the system programmer choose or any number of the tools not typically associated with z/OS. The command line interface comes with z/OS support but can be extended to support any other z/OS-based application or subsystem.

Parent topic: Project Giza overview

Components overview

Project Giza consists of three main components: zLux, Atlas, and Brightside CLI.

- **zLux overview**
- **Atlas overview**
- **Brightside CLI Overview**

Parent topic: Introducing Project Giza

zLux overview

zLUX is collaborative open-source product that modernizes and simplifies working on the mainframe. With zLUX you can create applications to suit your specific needs.

zLUX consists of the following components:

- Mainframe Virtual Desktop (MVD) - The desktop, accessed by the user through a browser.
- Giza 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 Giza Node Server.
- Applications - An application-type plug-in.
- TN3270 - A limited license plugin that provides a 3270 connection to the mainframe on which the Giza Node Server is running.
- zLUX sample apps - Sample application plug-ins.

Atlas overview

Atlas is a z/OS® RESTful web service and deployment architecture for z/OS microservices. Atlas 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 files, and persistent data.

Atlas can be used by any client application that calls its RESTful APIs directly.

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

Brightside CLI overview

Brightside Command Line Interface (Brightside CLI) lets application developers interact with the mainframe in a format that is natively familiar to them. Brightside CLI helps to increase overall productivity, reduce the learning curve for developing mainframe applications, and exploit the ease-of-use of off-platform tools. Brightside CLI 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 who need to quickly become efficient in supporting and building z/OS applications.

Brightside CLI 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 personal computer that can interact with z/OS operating systems.
- 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 platforms.

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

- Solution video
- Brightside CLI capabilities
- Supported platforms
- Experimental commands
- Third-Party software agreements

Notes:

- For a technical overview about how Brightside CLI integrates with Project Giza, see Technical Overview.
- For installation, upgrade, and software requirements, see Installing Brightside CLI.

Solution video

Watch this short video about how Brightside CLI works.

Brightside CLI capabilities

With Brightside CLI, 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 Brightside CLI.
- **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 Brightside CLI.
- **Provision mainframe environments** Exploit z/OSMF cloud provisioning features to provision environments on-demand.
- **Integrates z/OS actions** 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 Brightside CLI, see Brightside CLI Command Groups.

Prerequisites

For information about prerequisite software and configuration for Brightside CLI, see Prerequisites for Brightside CLI

Experimental commands

The Brightside CLI Early Access Preview contains features that are still under development and have not been tested to GA quality. You can enable or disable these commands. The experimental commands are disabled by default.

Important! You might encounter problems when using experimental commands.

For more information, see Enable and Disable Experimental Commands.

Third-party software agreements

Brightside CLI uses the following third-party software:

- ag-grid
- body-parser
- chalk
- cli-table2
- csv
- definitelytyped
- express
- filewatcher
- getmdl-select
- glob
- i18n
- jsonschema
- js-yaml
- levenshtein
- lodash
- log4js
- material-design-lite
- mustache
- Nested-property
- node.js
- node-forge
- node-progress
- node-tmp
- opn
- prettyjson
- prompt
- reflect-metadata
- rimraf
- simple-ssh
- stack-trace
- string-argv
- wrap-ansi
- yamljs
- yargs

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

To read each complete license right-click here to download the attached zip file.

Chapter 2. Installing Project Giza

This section is intended for system programmers and administrators who want to install and configure Project Giza.

Follow these procedures to install, configure, uninstall, and troubleshoot the different components of Project Giza. Before you install Project Giza, ensure that you understand and meet the prerequisites.

- **Prerequisites**
- **Installing components of Project Giza**
- **Verifying Installation**
- **Uninstalling Project Giza**
- **Troubleshooting installation**

Prerequisites

Before installing Project Giza, verify that your environment meets all of the prerequisites.

- **Prerequisites for z/OSMF configuration**
- **Prerequisites for zLUX**
- **Prerequisites for Atlas**
- **Prerequisites for CLI**

Parent topic: Installing Project Giza

Prerequisites for z/OSMF configuration

IBM z/OS Management Facility (z/OSMF) is a prerequisite for the Project Giza microservice that must be installed and running before you use Project Giza.

- z/OSMF Requirements for Project Giza
- 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 this article, we provide procedures and tips for the configuration required for Project Giza. 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 v2.2 documentation:

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

IBM z/OSMF v2.3 documentation:

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

z/OSMF Requirements for Project Giza

Meet the following prerequisites before you use Project Giza.

- z/OS requirements
- z/OSMF plug-in requirements

- REST services requirements

z/OS requirements

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

- **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 start 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 Brightside CLI.
- **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 Giza:

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

Additionally, Project Giza 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 Giza reuses z/OSMF's SAF, SSL, and LTPA configuration; therefore, these configurations must be valid and complete to operate Project Giza successfully.

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

Configuring z/OSMF

Follow these steps to verify your system requirements:

1. 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,

     `Z0S390RL 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
  ```

  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 Giza 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 Giza.

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 is configured correctly, follow these steps to create and validate a profile in Brightside CLI:

1. Meet the prerequisites for Brightside CLI.
2. Install Brightside CLI.
3. Create a zosmf profile in Brightside CLI. Issue the `bright help explain profiles` command to learn more about creating profiles in Brightside CLI. See How to display Brightside CLI 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 recieve a perfect score on the validation report, Project Giza can communicate with z/OSMF properly.

Additional tips for verifying your z/OSMF configuration

- 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
  ```

- Type the REST endpoint into your browser, for example: https://mvs.ibm.com:443/zosmf/restjobs/jobs
  - Browsing zosmf 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.

Parent topic: Prerequisites

Prerequisites for zLUX

Before installing zLUX, check the following items:

- **Verifying that your system meets the software requirements**
- **Confirming that Node.js is installed on the z/OS host**
- **Verifying port number availability**

Verifying your system meets the software requirements

Your system must meet the software requirements for zLUX.

1. Confirm that your environment meets the requirements for zLUX:
   - z/OS Version 2.2 or later, and whatever maintenance is required by the Node.js and Java installations, and 833 MB of HFS file space for the installation.
   - To verify that the most recent version of Java Version 8 is installed on z/OS, type **java -version**.
   - Confirm that the machine from which you plan to run zLUX runs one of the supported browsers:
     - Chrome version 54 or later
     - Firefox version 44 or later
     - Safari version 11 or later
     - Microsoft Edge **Note:** Microsoft Internet Explorer is not yet supported at any version level.
2. If your environment does not meet these requirements, consult your system administrator.

When you have confirmed that your environment meets the software requirements, go on to Confirming that Node.js is installed.

Confirming that NODE.js is installed

Node.js version 6.11.2 or later must be present on the z/OS host where you intend to install the Giza Node server.

1. If Node.js has not yet been installed on the z/OS host, follow the procedure for doing so: https://developer.ibm.com/node/sdk/ztp.

2. Set the NODE_HOME environment variable to the directory where your Node.js is installed (NODE_HOME=/proj/mvd/node/installs/node-v6.10.3-os390-s390x, for example).

After you have confirmed that Node.js is installed and the NODE_HOME environment variable has been set appropriately, you can verify that the necessary port numbers are available. Go on to Verifying port number availability.

Verifying port number availablity

Before you begin your zLUX configuration, it is a good idea to verify the availability of the various port numbers the you will need to specify for zLUX communications.

zLUX requires you to configure these port numbers in the zlux-example-server/config/zluxserver.json configuration file:

- `zssPort`, which is the port through which the Giza Node server communicates with the ZLUX Secure Services address space. By default, zLUX uses port 8542 for this purpose.
- An HTTP port for unencrypted access from the browser. By default, zLUX uses port 8543 for this purpose.
- An HTTPS port for encrypted access from the browser. By default, zLUX uses port 8544 for this purpose.

1. Determine which port numbers are in use and which are reserved:
   a. Run TSO NETSTAT PORTLIST to display a list of reserved ports.
   b. Run TSO NETSTAT to display a list of ports that are in use.
2. For the `zssPort`, HTTP and HTTPS ports, select port numbers that are not reserved or already in use. You will need to specify the selected port numbers to complete the procedure in Setting up the Giza Node server and the ZLUX Secure Services address space on z/OS topic.

After you have verified the availability of the port numbers that you will need to complete your Giza Node server and ZLUX Secure Services address space set up, you can go on to Setting up the Giza Node server and the ZLUX Secure Services address space on z/OS.

Prerequisites for Atlas

Before installing Atlas, check whether your environment meets the following requirements to ensure a successful installation.

- Atlas must be installed on z/OS® version 2.1 or later.
- Atlas requires a 64-bit Java™ 8 JRE or later.
- IBM® z/OS Management Facility (z/OSMF) must be installed and running. z/OSMF is a prerequisite for the Atlas microservice. For details, see Prerequisites for z/OSMF configuration.
- (Optional) To enable real-time access to SYSLOG, SDSF must be installed.

Pre-installation checklist

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

- The HFS directory where you install Atlas, 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 z/OSMF `/bootstrap.properties`, `/jvm.security.override.properties`, and `/resources/security/ltpa.keys files`.
- The Atlas http and https port numbers. By default, they are 7080 and 7443.
- The user ID that runs the Liberty Atlas 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`.

Parent topic: Prerequisites

Prerequisites for Brightside CLI

Meet the following prerequisites before you install Brightside CLI on your PC:

- Node.js® is a JavaScript runtime environment on which we architected Brightside CLI. You use the Node.js package manager (npm) to install Brightside CLI. Follow the instructions at Installing Node.js via package manager to install Node.js on your operating system. website at the

  Tip: If you are installing Node.js on a Linux or a macOS operating system, CA recommends 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`.
- You can install Brightside CLI on any Windows, macOS, and Linux operating system that supports Node.js version 6 or later.
- Before you can use Brightside CLI to interact with the mainframe, a systems programmer must install and configure IBM z/OSMF in your environment. For more information about how systems programmers and security administrators perform the z/OSMF configuration, see Overview of the z/OS Management Facility Configuration Process.

Note: CA Technologies does not maintain the prerequisite software that Brightside CLI requires. You are responsible for updating Node.js and other prerequisites on your personal computer or workstation. We recommend that you update Node.js regularly to the latest Long Term Support (LTS) version.

Installing components of Project Giza

Obtain the Project Giza installation media

Download the Project Giza installation media and transfer the files to client workstations to begin using the product at your site.

Follow these steps:

1. Download the Project Giza PAX file from the GitHub repository.
2. Extract the contents of the Project Giza PAX file. The Giza PAX contains the following components:
   - Atlas PAX
   - Brightside CLI .tgz file
   - zLux PAX
   - Source code for all components

Before you install the components of Project Giza, ensure that you meet the prerequisites.

Follow the procedures in this section to install the components of Project Giza.

- **Installing zLUX**
- **Installing Atlas**
- **Installing Brightside CLI**

Installing zLUX

To install zLUX, complete the steps specified in this section.

Setting up the Giza Node server and the ZLUX Secure Services address space on z/OS

Setting up the Giza Node server and the ZLUX Secure Services address space on z/OS is a multi-step installation and configuration process.

The zLUX archive is distributed as a pax archive. When you unpack the archive, both the Giza Node server and the ZLUX Secure Services (ZSS) are installed on the z/OS host.

1. If you have not already done so, follow the procedures in these prerequisite sections:
   - Verifying that your system meets the software requirements
   - Confirming that Node.js is installed
   - Verifying port number availability
2. To obtain the xxx, follow the instructions in Obtain the Project Giza installation media.
3. Use the cd command to navigate to the directory in which you want to restore the zLUX archive, or create a new directory for that purpose and navigate to the new directory.
4. Execute the pax command with these specific options to restore the zLUX archive under the current directory:

   pax -r -px -f archive-location/zLUX.pax

   where:

   - -p specifies which file characteristics you want to restore.
   - x preserves extended attributes that were set originally by the extattr command. When you run the ls command in the directory, you see the directory contents:

```
README.md
sample-app/
tn3270-ng2/
zlux-app-manager/
zlux-example-server/
zlux-ng2/
zlux-platform/
zlux-proxy-server/
zlux-shared/
zos-subsystems/
```

5. Verify that the extended attributes were preserved for the file zlux-example-server/bin/zssServer:

```
$ cd zlux-example-server/bin
$ ls -E
total 5824
-rw-rw-r-- --s-  1 TSSPG   TSUSER   177 Jan 11 13:11 nodeSever.bat
-rwxrwxr-x --s-  1 TSSPG   TSUSER   538 Jan 11 13:11 nodeServer.sh
-rwxrwxr-x aps-  1 TSSPG   TSUSER   2945024 Jan 31 19:26 zssServer
-rwxrwxr-x a-s-  1 TSSPG   TSUSER   538 Jan 11 13:11 zssSever.sh
```

   If the a attribute is absent, you lack sufficient authority on the z/OS system. To correct this problem, have a user who has sufficient authority run the extattr command to add the attribute: extattr +a zssServer

6. If you need to specify a port number other than the default (8542) for `zssPort` in zluxserver.json, complete these steps:
   a. Navigate to the zlux-example-server/config directory: cd zlux-example-server/config.
   b. Edit zluxserver.json to assign the appropriate number to `zssPort`.
   c. Save the changes to zluxserver.json.
7. Upon startup, the Giza Node server loads the zluxserver.json configuration file (from zlux-example-server/deploy/instance/ZLUX/serverConfig/zluxserver.json). The JSON configuration file adheres to a specific structure, as shown in the following figure:

```
  "node": {
    "http": {
      **"port": 8543**
    },
```

```
    "https": {
      **"port": 8544**,
      //pfx (string), keys, certificates, certificateAuthorities, and
certificateRevocationLists are all valid here.
      **"keys"**: ["../deploy/product/ZLUX/serverConfig/server.key"],
      **"certificates"**: ["../deploy/product/ZLUX/serverConfig/
server.cert"]
      }
    },
    "childProcesses": [
      {
        "path": "../bin/zssServer.sh"
      }
    ]
  },
```

Note the following about specifying values in the JSON configuration file:

- **Port field for the http field of the node object**

  Required if you intend to access the ZLUX Secure Services address space through unencrypted HTTP. Specify a port number that is available on your system.

- **Port field for the https field of the node object**

  Specify the following ports to connect to the node server. Specify a port number that is available on your system. Requires `keys` and `certificates`:

  – **keys**

    The location of the private key file relative to the location of the JSON configuration file. The code page of this file should be the native code page of the host (EBCDIC for z/OS).

  – **certificates**

    The location of the certificate file relative to the location of the JSON configuration file. The code page of this file should be the native code page of the host (EBCDIC for z/OS).

Note: Currently, the ZLUX Secure Services address space configuration JSON file at zlux-example-server/config/zluxserver.json contains an example node configuration, so you can refer to one file for both the ZLUX Secure Services address space and the Giza Node server.

8. Start the zLUX servers:

   - If you are running the Giza Node server on z/OS, complete these steps:

     a. Navigate to zlux-example-server/bin.

     b. Run nodeServer.sh:

     ```
     nodeServer.sh
     ```

     The ZLUX Secure Services address space (zssServer.sh) starts automatically.

   - If you are running the Giza Node server on Windows, complete these steps:

     a. Navigate to zlux-example-server\bin.

     b. Run nodeServer.bat:

     ```
     nodeServer.bat --hostServer=serveraddress --hostPort=portnumber
     ```

     where:

     – serveraddress is the address of the z/OS system where Node.js is running.

     – portnumber is the port number that you configured for `zssPort` in step 5 of this procedure.

Note: The ZLUX Secure Services address space log output is very verbose. Redirecting the output is not generally recommended. If you require a log file, restart the ZLUX Secure Services address space

and redirect the output to a file of your choosing. Because log output can quickly fill up a file system, it is not recommended that you leave the ZLUX Secure Services address space running for long periods of time with the output directed to a file.

Go on to Opening the MVD in a browser.

Opening the MVD in a browser

You can open the MVD in any supported browser.

1. In 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 ran the Giza Node server.
   - `httpPort` is the value that was assigned to `node.http.port` in zluxserver.json.
   - `httpsPort` is the value that was assigned to `node.https.port` in zluxserver.json. For example, if you ran the Giza Node server on host `myhost` and the value that is assigned to `node.http.port` in zluxserver.json is `12345`, you would specify `https://myhost:12345/ZLUX/plugins/com.rs.mvd/web/index.htm`.
2. Browse the interface.

Installing Atlas

Installing Atlas involves obtaining the Atlas Archive, running the install script, and configuring files.

Before installing Atlas, ensure that your environment meets the prerequisites for Atlas.

To install Atlas, complete the following steps:

1. Obtain the Project Giza installation media, which includes the Atlas PAX file.
2. Transfer the following files to z/OS® System:
   - The Atlas PAX archive that contains Liberty Profile binaries and the Atlas application.
   - The Atlas Install script

   Note: The Atlas Install script is an ASCII file. If the Install script is transferred by using FTP, the Install script is converted into the appropriate format for the server. If the Install script is transferred by using SCP or SFTP, the Install script is not converted, and it should be converted by taking the action specified in the **Important** note below.

   Alternatively, transfer the Atlas archive to your z/OS system, and extract the archive. The Atlas archive contains the following files:

   - The Atlas PAX archive that contains Liberty Profile binaries and the Atlas application
   - The Atlas Install script
   - This User's Guide
   - The readme file

   Extracting the archive on the server does not convert the Install script. The Install script should be converted by taking the action specified in the **Important** note below.

   Important: To convert the Install script from ASCII into the standard EBCDIC code page, use `ICONV`. For example,

   ```
   iconv -f ISO8859-1 -t IBM-1047 atlas-wlp-package-0.0.1.sh > atlas-wlp-package-EBCDIC.sh
   ```

   Note: Transfer the PAX archive and the Install script in binary mode to the Atlas installation directory that is chosen during planning, for example, `/var/atlas`, or wherever you choose to install Atlas.
3. Run the Atlas install script.

The install script must be transferred to the same Atlas installation directory of the Atlas PAX archive. Run the install script in the installation directory with a user ID that has the authority to:

- Unpack the Atlas PAX archive and install Atlas into the installation directory, for example, `/var/atlas`. About 205 MB is needed to unpack the archive and more space is needed for Liberty operation and logging.
- Set the file group ownership to IZUADMIN.
- Create symbolic links to the files that z/OSMF owns.

Therefore, use super user authority to run the Atlas install script.

4. Change the ownership of Atlas installation directory and files.

   The user who runs the Atlas Liberty server needs the access to the Atlas installation directory and files. You can use the same user ID that runs the z/OSMF IZUSVR1 started task to run the Atlas Liberty server. By default, it is the user IZUSVR.

   To change the ownership of the Atlas installation directory and files, enter the following z/OS UNIX™ System Services command from the Atlas installation directory:

   ```
   chown -R IZUSVR *
   ```

   You might need super user authority to run this command. Use an alternative user ID if you chose not to use the default z/OSMF IZUSVR1 started task user.

5. Create a member FEKATLS in your system PROCLIB data set.

   The install script creates a file that is called FEKATLS.jcl is created in your Atlas installation directory. Copy this file to a system PROCLIB data set by using the following z/OS UNIX System Services command:

   ```
   cp FEKATLS.jcl "//'hlq.PROCLIB(FEKATLS)'"
   ```

   The FEKATLS procedure starts a Liberty profile server running the Atlas microservice application.

6. Configure the FEKATLS started procedure.

   To run the FEKATLS procedure as the user IZUSVR, define the procedure to the STARTED class by using RACF® or equivalent, for example:

   ```
   • RDEF STARTED (FEKATLS.*) STDATA(USER(IZUSVR) GROUP(IZUADMIN))
   • SETR RACLIST(STARTED) REFRESH
   ```

   Here is an example of the FEKATLS procedure JCL:

   ```
   //*********************************************************************
   //*   Licensed Materials - Property of IBM                            *
   //*                                                                   *
   //*   5655-EX1                                                        *
   //*                                                                   *
   //*   Copyright IBM Corp. 2017. All rights reserved.                  *
   //*                                                                   *
   //*   US Government Users Restricted Rights - Use,                    *
   //*   duplication or disclosure restricted by GSA ADP                 *
   //*   Schedule Contract with IBM Corp.                                *
   //*                                                                   *
   //*********************************************************************
   //*                                                                   *
   //* ATLAS WLP PROCEDURE                                               *
   //*                                                                   *
   //* This is a procedure to start the Atlas web server platform,       *
   //* running on the WebSphere Liberty Profile. This procedure          *
   //* requires a WebSphere Liberty Angel procedure is running, such as  *
   //* z/OSMF procedure "IZUANG*".                                       *
   ```

```
//*                                                                    *
//* NOTE: THIS JCL IS MODIFIED BY THE ATLAS INSTALLATION PROCESS TO  *
//* SET THE ATLAS INSTALLATION PATH. IF THE INSTALLATION PATH        *
//* CHANGES, MODIFY THE SRVRPATH VALUE ACCORDINGLY.                  *
//*                                                                    *
//*********************************************************************
//ATLAS PROC
//*--------------------------------------------------------------------
//* SRVRPATH - The path to the HFS directory where the Atlas server
//*            was installed.
//*--------------------------------------------------------------------
//EXPORT EXPORT SYMLIST=*
//  SET SRVRPATH='${atlaspath}'
//*--------------------------------------------------------------------
//* Start the Atlas WebSphere Liberty Profile server
//*--------------------------------------------------------------------
//STEP1   EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
//  PARM='PGM &SRVRPATH/wlp/lib/native/zos/s390x/bbgzsrv Atlas'
//WLPUDIR  DD PATH='&SRVRPATH/wlp/usr'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//*--------------------------------------------------------------------
//* Optional logging parameters that can be configured if required
//*--------------------------------------------------------------------
//*STDOUT   DD PATH='&SRVRPATH/std.out',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=SIRWXU
//*STDERR   DD PATH='&SRVRPATH/std.err',
//*            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//*            PATHMODE=SIRWXU
```

7. Add Atlas users to the z/OSMF users group (IZUUSER).

   Atlas uses z/OSMF to access data sets, z/OS UNIX System Services files, and job spool files. To use these z/OSMF services, Atlas users must be authorized to z/OSMF resources. For more information, see the *IBM z/OS Management Facility Configuration Guide*, Appendix A.

   To add Atlas users to the z/OSMF IZUUSER group, use RACF or equivalent. For example,

   ```
   CONNECT userid GROUP(IZUUSER) AUTH(USE)
   ```

8. Start the Atlas server.

   To start Atlas manually, enter the START operator command:

   ```
   S FEKATLS
   ```

   To start Atlas automatically at IPL, add the START command to your active COMMNDxx parmlib member.

9. Optional: Change your language in Atlas by adding the following line to the `jvm.options` file, for example,

   ```
   -Duser.language=de
   ```

   where de can be replaced with other language codes.

What to do next

Verify whether Atlas is successfully installed. For more information, see Verifying Atlas Installation.

Parent topic: Installing Project Giza

Installing Brightside CLI

As a systems programmer or application developer, you install Brightside CLI on your PC. You can install Brightside CLI on any PC that is running a Windows, Linux, macOS operating system that supports Node.js version 6 or later.

Before you install Brightside CLI, ensure that you meet the Prerequisites.

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

Install Brightside CLI

To use Brightside CLI on your PC, install the product from that is available at your site.

Follow these steps:

1. Obtain the Project Giza installation media, which includes the brightside.tgz file. Use FTP to distribute the brightside.tgz file to client workstations. Users can now install Brightside CLI on their PC.
2. Open a command line window. Browse to the directory where you downloaded the Brightside CLI installation package (.tgz file). Issue the following command to install Brightside CLI on your PC:

   ```
   npm install -g <file_name>
   ```

   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 Brightside CLI.

   Brightside CLI is installed on your PC.
3. You must create a Brightside CLI profile before you can issue Brightside CLI commands that communicate with z/OSMF on mainframe systems. For more information about the available commands and options for creating z/OSMF profiles, issue the `bright help explain profiles` and `bright zosmf create bright-profile --help` commands.

   Tip: Brightside profiles contain information (for example, host name, port, and user ID) that is required for Brightside to interact with remote systems. Profiles allow you to "target" a system, region, or instance for a command. You create and configure profiles at the command group level. Most command groups require a "zosmf" Brightside profile.
4. To ensure that your Brightside CLI profile can communicate with z/OSMF on mainframe systems, issue the following z/OSMF profile validation command:

   ```
   bright zosmf validate profile
   ```

   The command runs a series of tests and returns a report. If the report returns any failures or warnings, send the report to your systems programmer for analysis. Failures might indicate that your Brightside CLI profile is not configured correctly for your environment. For more information about how to use command, see Validating Installation.

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

Verifying installation

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

Verifying Atlas installation

After Atlas is installed and the FEKATLS procedure is started, you can verify the installation from an internet browser by using the following URL:

https://*your.server*:*atlasport*/Atlas/api/system/version

where *your.server* matches the host name or IP address of your z/OS® system where Atlas is installed, and *atlasport* matches the port number that is chosen during installation. You can verify the port number in the server.xml file that is located in the Atlas 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".

Important: This URL is case-sensitive.

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

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

Note: For the first interaction with the Atlas 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 Atlas was successfully installed, you can access the UI at:

https://*your.server:atlasport*/ui//#/

Verifying the availability of Atlas REST APIs

To verify the availability of all Atlas REST APIs, use the Liberty Profile's REST API discovery feature from an internet browser with the following URL:

https://*your.server:atlasport*/ibm/api/explorer

Note: This URL is case-sensitive.

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 Atlas APIs. This ensures that your z/OSMF configuration is valid, complete, and compatible with the Atlas application. For example, try the following APIs:

Atlas : JES Jobs APIs `GET /Atlas/api/jobs` This API returns job information for the calling user. Atlas : Dataset APIs `GET /Atlas/api/datasets/userid.**` This API returns a list of the userid.** MVS datasets.

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

Parent topic: Installing Atlas

Validating Brightside CLI installations

After the systems programmer configures z/OSMF and application developers install Brightside CLI, issue the Brightside CLI profile validation command to verify that your z/OSMF profile will function properly. The Brightside CLI profile validation command runs a series of tests on z/OSMF APIs and prints a report that can help you identify problems with your profile and connection details.

Use this command to verify that your Brightside CLI profile can communicate with z/OSMF on z/OS systems. Review the report and inform your systems programmer if there are any failures or warnings.

Before you issue the z/OSMF profile validation command, ensure that the following tasks are complete:

- Your systems programmer configured z/OSMF.
- Brightside CLI is installed.
- You created at least one Brightside CLI `zosmf` profile that lets you access z/OSMF functionality on z/OS systems.

Command syntax

You issue the z/OSMF validate profile command to verify that your Brightside CLI mainframe security access has been properly configured and that your profile details are correct. With the command, you can verify your default profile or any other specific profile.

Issue the following command to view a list of the tests that the profile validation command will run:

```
Issue the following command to verify that your *default profile* is
 configured correctly:
```

Issue the following command and specify a `profile_name` to verify that *a specific profile* is configured correctly:

```
bright zosmf validate profile --bpn <profile_name>
```

Profile validation results

When you issue the z/OSMF validate profile command, Brightside CLI runs a series of tests. The command presents you with a table that displays the test results. Review the table and report any failures or warnings to your systems programmer.

The report returns the following information:

- **Task** The type of action that the test performs.
- **Status** The result of the test returns as *OK*, *Warning*, or *Failed*. In some cases, a *Failed* result on one test can result in a *Warning* result on the subsequent tests.
- **Description** Details about the execution and results of the test.
- **Endpoint** The REST API endpoint on which the test acted.

Correcting problems detected by the validate profile command in Brightside CLI

Brightside CLI includes a profile validation command that runs a series of tests on z/OSMF REST APIs (and their endpoints) and prints a report that can help you identify problems with profiles and connection details. As a systems programmer, issue this command to test the end-to-end integration of Brightside CLI with your z/OSMF and your z/OS environment.

Note: Application developers that use Brightside CLI can issue the command at any time and provide the report to systems programmers for problem resolution. For more information about how application developers use the profile validation command, see Validate Installation.

Use the report to resolve problems

You can use the results of the validate profile command to troubleshoot and fix problems with your z/OSMF configuration. The result of each test returns as *OK*, *Warning*, or *Failed*. If you receive a *Warning* or *Failed* result on a test, you can refer to IBM documentation for information on how to configure that particular REST API. For information on which of the z/OSMF REST APIs Brightside CLI uses, see Configure z/OS Management Facility Security. We provide links to IBM documentation for configuring each REST API.

For example, you receive a *Failed* result on the "Issue Console command" task. Refer to the list of REST APIs in Configure z/OS Management Facility Security and follow the links to the corresponding IBM documentation for the z/OS console services API.

Uninstalling Project Giza

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

Uninstalling zLUX

To uninstall zLUX, complete the following step:

Delete the original folders (except in the case where you have customized the installation to point to folders other than the original folders).

Uninstalling Atlas

To uninstall Atlas, take the following steps:

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

```
P FEKATLS
```

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

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

4. Delete the z/OS® UNIX™ System Services Atlas directory and files from the Atlas installation directory by using 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 Atlas installation directory correctly. Running a recursive remove command with the wrong directory name might delete critical files.

Parent topic: Installing Atlas

Uninstalling Brightside CLI

To uninstall Brightside CLI, issue the following command:

```
npm uninstall -g brightside
```

If you don't receive any errors, the uninstall process was successful.

Troubleshooting installation

When you experience problems, refer to these topics to determine the corrective actions to take.

- **Troubleshooting Installing zLux**
- **Troubleshooting Installing Atlas**
- **Troubleshooting Installing Brightside CLI**

Troubleshooting installing zLUX

To help zLUX research any problems you might encounter, collect as much of the following information as possible:

- zLUX 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 Giza node server
- Error message codes
- Screenshots (if applicable)
- Other relevant information (such as the NodeJS version that is running the Giza Node Server and the browser and browser version)

Troubleshooting installing Atlas

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

- Check whether your Atlas Liberty server is running.

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

  ```
  S FEKATLS
  ```

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

  In the Display Active (DA) panel of SDSF under ISPF, select the FEKATLS job to view the started task output. If the Atlas 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 FEKATLS job output, respond to them.
- Check whether the URL that you use to call Atlas 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 Atlas Liberty server.
- If testing the Atlas 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.

  If calling the z/OSMF RESTJOBS API directly fails, fix z/OSMF before Atlas can use these APIs successfully.
- If testing the Atlas REST API for dataset 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.

If calling the z/OSMF RESTFILES API directly fails, fix z/OSMF before Atlas 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 Atlas server cannot connect to the z/OSMF server, check the following item:

By default, the Atlas server communicates with the z/OSMF server on the localhost address. If your z/OSMF server is on a different IP address to the Atlas 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.

Parent topic: Installing Project Giza

Troubleshooting installing Brightside CLI

The following issues are known to exist when installing this release of Brightside CLI:

- **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 Brightside CLI yields success. Some users also report success using the following workarounds: - Issue the `npm cache clean` command. - Uninstall and reinstall Brightside CLI. For more information, see Install BrightSide CLI. - 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.

Chapter 3. Using Project Giza

Project Giza delivers the following components:

- zLux that provides a framework for building and sharing extension apps using web services and modern UI toolkits such as AngularJS. It includes all interactions that exist in 3270 terminals and web interfaces like z/OSMF.
- REST APIs that provide a range of APIs for the management of jobs, data sets, z/OS UNIX System Services files, and persistent data.
- Brightside Command Line Interface (Brightside CLI) 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 more information about the tasks that you can perform with each component, see the following sections:

- **Using zLUX**
- **Using APIs **
- **Using Brightside CLI**

Using zLUX

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

Navigating MVD

To access the MVD, see Opening the MVD in a browser

Logging into the MVD

To log in, sign in using your mainframe credentials:

1. In the **Username** field, type your username.
2. In the **Password** field, type your password.
3. Press Enter. Upon authentication of your user name and password, the desktop opens.

Note the following items on the desktop:

- In the lower left corner, is the **application menu** which includes access to the following:
  - **z/OS Subsystems** - An application that helps you find information about the important services on your mainframe, such as CICS, DB2 and IMS.
  - **TN3270 Terminal** - A limited license plugin that provides a 3270 connection to the mainframe on which the Giza Node Server is running.
  - **Welcome App** - Sample plugin.
- Across the bottom of the window is the **application launch bar**.
- The bottom right corner displays current date and time and allows access to the current user session.

Logging out

You can log out by clicking the avatar.

Using Atlas Explorers within zLUX

Atlas 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 Atlas Web UI and are accesible via the Atlas 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 Atlas 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.

Dataset 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.

Creating an application plug-in

This information is under development.

A zLUX application plug-in is a separately installable collection of files that can provide a set of web resources (JavaScript, HTML, and so on) that can be presented in the user interface as window (or as an embedded panel) or a set of RESTful services, or both RESTful services and user interfaces.

Before you can begin building a zLUX application plug-in, you must set the environment variables that are necessary to 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** The UNIX environment variables must be set appropriately before you can use the zLUX build procedures to develop your own plug-ins.
- **Experimenting with the sample application plug-in** Your zLUX installation provides a sample application plug-in with which you can experiment.

Setting the environment variables for plug-in development

The UNIX environment variables must be set appropriately before you can use the zLUX build procedures to develop your own plug-ins.

To set up the environment, the node must be accessible on the PATH.

1. To determine if the node is on the PATH, type "node --version" on the command line. If data is returned, the node is already on the PATH and no action is required.

2. 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).

Parent topic: Creating an application plug-in

Experimenting with the zLUX sample application plug-in

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

To build the sample app, 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: Be aware that whenver you change the source code for the sample application, you must subsequently rebuild the sample application.

1. Try adding an item to sample-app. The following figure shows the unmodified contents of app.component.ts:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  items = ['a', 'b', 'c', 'd']
  title = 'app';
}
```

2. Save your changes to app.component.ts.
3. Issue one of these commands:
   - npm run build to rebuild the application plug-in.
   - npm start to rebuild the application plug-in and wait for additional changes to app.component.ts.
4. Reload the web page.
5. Whenever you make changes to the sample application source code, you must subsequently rebuild the application:
   a. Navigate to the sample-app subdirectory where you made the source code changes.
   b. Issue this command: npm run build
   c. Reload the web page.

Parent topic: Creating an application plug-in

Using APIs

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

Using Atlas REST APIs

Atlas 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 Atlas 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.

Dataset APIs

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

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.

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.

System APIs

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

USS File APIs

Use USS File APIs to read and list UNIX Files. See the following table for the available operations and their descriptions and prerequisites.

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.

Dataset APIs

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

REST API	Description	Prerequisite
`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

Parent topic: Using Atlas REST APIs

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

REST API	Description	Prerequisite
`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 dataset 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
`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

Parent topic: Using Atlas REST APIs

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 Atlas, 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

Parent topic: Using Atlas REST APIs

System APIs

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

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

Parent topic: Using Atlas REST APIs

USS File APIs

Use USS File APIs to read and list UNIX 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

Parent topic: Using Atlas REST APIs

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

Parent topic: Using Atlas REST APIs

Programming Atlas REST APIs

You can program Atlas REST APIs by referring to these examples:

- **Sending a GET request in Java** Here is sample code to send a GET request to Atlas in Java™.
- **Sending a GET request in JavaScript** Here is sample code written in JavaScript™ using features from ES6 to send a GETrequest to Atlas.
- **Sending a POST request in JavaScript** Here is sample code written in JavaScript™ using features from ES6 to send a POST request to Atlas.
- **Extended API sample in JavaScript** Here is an extended API sample that is written using JavaScript™ with features from ES62015 (map).

Parent topic: Using Project Giza

Sending a GET request in Java

Here is sample code to send a GET request to Atlas 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();
        }
    }
}
```

}

Parent topic: Programming Atlas REST APIs

Sending a POST request in JavaScript

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

```
// Call the jobs POST api to submit a job from a dataset
 (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;
}
```

Parent topic: Programming Atlas REST APIs

Sending a POST request in JavaScript

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

```
// Call the jobs POST api to submit a job from a dataset
 (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;
}
```

Parent topic: Programming Atlas REST APIs

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;
}
```

Parent topic: Programming Atlas REST APIs

Using Atlas WebSocket services

Atlas provides WebSocket services that can be accessed by using the WSS scheme. With Atlas 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 Atlas 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");
    }
}
```

Parent topic: Using Atlas WebSocket services

Using Brightside CLI

This section contains the following articles about using Brightside CLI:

- How to display Brightside CLI help
- Brightside CLI command groups
- Enable and disable experimental commands
- Brightside CLI scenarios

How to display Brightside CLI help

Brightside CLI contains a help system that is embedded directly into the CLI. When you want help with Brightside CLI, you issue various help commands that provide you with information about Brightside CLI, the usage, syntax, actions, and options. You can also display all help, group-level help, and information about the structure of Brightside CLI syntax.

- Get started with Brightside CLI syntax
- Display top-level help
- Display group-level help

Get started with Brightside CLI syntax

If you are using Brightside CLI for first time and want to learn more about how to use the syntax (and how Brightside CLI works), open a command line window and issue the following command:

```
bright help explain brightside
```

To learn how to perform a specific task, or if you want to search for information about a specific command, issue the following command:

```
bright help search all <term>
```

With this command, Brightside CLI searches and displays help for information about commands, syntax, descriptions, options, and so on.

Example:

```
bright help search all console
```

Display top-level help

To to display top-level Brightside CLI help, open a command line window and issue the following command:

```
bright --help
```

(Optional) Issue the following command to display top-level Brightside CLI help using the alias for the help command:

```
bright -h
```

Display group-level help

You can use group-level help to get more information about a specific command group. 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 name> --help
```

Example:

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

(Optional) Issue the following command to view Brightside CLI group-level help using the alias for the help command:

```
bright compiler -h
```

Brightside CLI command groups

Brightside CLI contains command groups that focus on specific business processes that you (application developers and systems programmers) perform during your day-to-day activities. For example, the compiler command group lets you perform tasks that relate to compiling source code on mainframe systems. The projects command group lets you perform tasks that relate to managing source code that you store in version control systems, such as Git and CA Endevor. You can perform all the tasks that relate to these business processes using a unified command-line interface - Brightside CLI.

The command groups contain commands that let you perform actions on specific objects. For each action on an object, you can specify options that you apply as the Brightside CLI commands execute.

In this article, we review all the Brightside CLI command groups and provide you with a brief synopsis of the tasks that you can perform using the commands in each group. For more information see How to display Brightside CLI help.

Important: Before you issue these commands, ensure that you create and validate your zosmf profile so that Brightside CLI can communicate with z/OS systems.

Brightside CLI contains the following command groups: - ca-disk - cics - compiler - config - contribute - db2 - endevor - help - ipcs - profiles - projects - provisioning - zos-console - zos-files - zos-jobs - zos-ssh - zos-tso - zos-utils - zosmf

ca-disk

The ca-disk command group is an experimental command group that lets you interact with CA Disk Backup and Restore from a familiar command-line interface. To execute commands using the ca-disk command group, CA Disk Backup and Restore must be installed on your system.

With the ca-disk command group, you can perform the following tasks:

- Archive and restore data sets that you archived using CA Disk Backup and Restore.

Note: For more information about ca-disk syntax, actions, and options, open Brightside CLI, and issue the following command:

cics

The cics command group is an experimental command group that lets you interact with CICS regions from a familiar command-line interface.

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

- Start or stop a CICS region.
- Manage CICS resources such as DB2CONN, MQCONN, PROGRAM, and TRANSACTION.
- Issue MVS modify commands against a CICS region.
- Run a CICS 3270 MQ Bridge transaction.
- Submit batch utilities such as DFHCSDUP.

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

```
bright cics -h
```

compiler

The compiler command group is an experimental command group that lets you compile code on mainframe systems. As a developer, you can use Brightside CLI to work on mainframe code locally and compile the code.

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

- Build and compile COBOL (common business-oriented language) and HLASM (IBM High Level Assembler) source code on mainframe systems.

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

```
bright compiler -h
```

config

The config command group lets you display and configure Brightside CLI settings and values to fit your needs.

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

- Display the location of the Brightside CLI configuration directory on your PC.
- Display the status of experimental features and commands (enabled or disabled).
- Enable and disable experimental features and commands.

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

```
bright config -h
```

contribute

The contribute command group lets you generate code that you can contribute to help improve Brightside CLI.

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

- Submit feedback to the Brightside CLI engineering team.

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

```
bright contribute -h
```

db2

The db2 command group is an experimental command group that lets you interact with the IBM Db2 Databases.

To execute commands in the db2 command group, IBM Db2 Database must be installed and configured on your system. With the db2 command group, you can perform the following tasks:

- Submit batch jobs to execute Db2 SQL and interact with databases.
- Export Db2 tables to stdout or a local file on your PC.
- Create and manage Db2 profiles that allow you to target specific Db2 regions, libraries, and plan names. To use Db2 commands, you must also specify a basic zosmf profile.

Note: For more information about db2 syntax, actions, and options, open Brightside CLI, and issue the following command:

```
bright db2 -h
```

endevor

The endevor command group is an experimental command group that lets you interact with CA Endevor remotely.

To execute commands in the endevor command group, CA Endevor must be installed and configured on your system. With the endevor command group, you can perform the following tasks:

- Submit Software Control Language (SCL) to CA Endevor SCM to manage elements, packages, symbols, systems, and more. To learn more about the flexibility of SCL, see the IBM SCL Reference Guide.
- Generate, delete, and move CA Endevor elements.

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

```
bright endevor -h
```

help

The help command group provides context-sensitive help in Brightside CLI.

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

- Get general information about Brightside CLI and learn how to create profiles and job cards.
- Get context-sensitive help for any command, action, object, and option.
- Learn about common mainframe terms and concepts. For example, you can display a description of how data set naming conventions work on z/OS.
- Search all Brightside CLI help text for a specific term or phrase.

Note: For more information, see How to Display Brightside CLI Help.

ipcs

The ipcs command group is an experimental command group that lets you interact with Interactive Problem Control System (IPCS) on z/OS. With the ipcs command group, you can perform the following tasks:

- Submit a command to IPCS and review the dump data in a data set of your choice.
- Parse the IPCS dump data with a JSON options document to isolate specific data.

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

```
bright ipcs -h
```

profiles

The profiles command group is an experimental command group that lets you create, manage, and validate profiles for use with other Brightside CLI command groups. Profiles allow you to issue commands to different 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 anyBrightside CLI command group that supports profiles.
- Set the default profile to be used within any Brightside CLI command group.
- List profile names and details for any Brightside CLI command group, including the default active profile.
- Validate profiles for other Brightside CLI command groups. For more information, see Validate Installation.

Note: For general information about profiles, open Brightside CLI, and issue the following command:

```
bright help explain profiles
```

For more information about profiles syntax, actions, and options, open Brightside CLI, and issue the following commands:

```
bright profiles -h
```

projects

The projects command group is an experimental command group that lets you interact with projects in CA Endevor, Git, or other version control software. If you use Git, a minimum of Git version 2.9 is required.

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

- Generate an element from your CA Endevor project.
- Create/initialize a new project (CA Endevor, Git, Mercurial, or data set) from mainframe source code.
- Generate a list of all files that are tracked by the current project.
- Pull changes from version control software of your choice, or retrieve element listings from CA Endevor.
- Synchronize element listings and upload elements to your project.

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

```
bright projects -h
```

provisioning

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

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 Brightside CLI, 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 Brightside CLI, 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 Brightside CLI, and issue the following command:

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

zos-files

The zos-files command group lets you interact with USS files and 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.
- Edit data sets locally in your preferred Integrated Development Environment (IDE). Your changes are reflected on the mainframe when you save the local file.
- Download data sets or USS files from a mainframe to your local PC.

- Upload local files to mainframe data sets.
- Interact with VSAM data sets directly, or invoke Access Methods Services (IDCAMS) to work with VSAM data sets.
- List data sets, print data sets, and search the contents of a data set.

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

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

See Submit a Job and Print Job Output for a use-case that demonstrates some zos-files commands.

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.
- Delete jobs.
- Download job output to your PC.
- List jobs. You can view finished jobs, active jobs, jobs in INPUT status, or all jobs.
- Print various information about jobs and spool, such as spool content, JCL, JES messages, and more.
- Create zos-jobs profiles that store your configuration details so that you can quickly interact with jobs. To use zos-jobs commands, you must also specify a basic zosmf profile.

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

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

See Submit a Job and Print the Job Output for a use case that demonstrates some zos-jobs commands.

zos-ssh

The zos-ssh command group is an experimental command group that lets you interact with UNIX System Services (USS) on z/OS systems.

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

- Issue USS commands via SSH (secure shell) protocol.

Note: For more information about zos-ssh syntax, actions, and options, open Brightside CLI and issue the following command:

```
bright zos-ssh -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:

- Create a TSO address space and issue TSO commands to the address space.
- Review TSO command response data in Brightside CLI.

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

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

zos-utils

The zos-utils command group is an experimental command group that lets you use common mainframe utilities remotely.

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

- Transfer a file across systems with a File Transfer Protocol (FTP) command.
- Copy data between data sets with the IEBCOPY utility.
- Overwrite content in a data set with the IMASZAP utility.

Note: For more information about zos-utils syntax, actions, and options, open Brightside CLI, and issue the following command:

```
bright zos-utils -h
```

zosmf

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

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

- Create and manage your Brightside CLI zosmf profiles. You must have at least one zosmf profile to issue most commands. Issue the `bright help explain profiles` command in Brightside CLI 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 Validate Installation.
- 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 Brightside CLI, and issue the following command:

```
bright zosmf -h
```

Enabling and Disabling Experimental Commands

Brightside CLI includes experimental commands, which are currently in development and are not ready for general availability. You can enable or disable these commands. The experimental commands are disabled by default.

Important! If you use these commands, you might encounter errors, unexpected behavior, incompatibilities with your system, or incomplete help text.

Enable experimental commands

To enable the experimental commands, issue the following command:

```
bright config set experimental-features enabled
```

The experimental commands now appear in your help menu with the prefix (*experimental)*.To view all available commands, including the experimental commands, issue the following command:

```
bright -h
```

Disable experimental commands

To disable the experimental commands, issue the following command:

```
bright config set experimental-features disabled
```

Check status of experimental commands

To determine whether the experimental commands are enabled or disabled, issue the following command:

```
bright config get experimental-features
```

Note: Experimental commands display the prefix `(experimental)` in Brightside CLI help when they are enabled. For example:

```
cics       (experimental) Issue commands to interact with CICS regions
compiler   (experimental) Compile source code on the mainframe
```

Brightside CLI scenarios

This section contains sample scenarios that can help you learn how to use Brightside CLI. In each use case, we walk you through the process of issuing Brightside CLI commands so that you can use various Brightside CLI features to accomplish common mainframe tasks.

- Submit a Job and Print Job Output

Prerequisites

The following prerequisites apply to all of the scenarios that are contained in this section. When a scenario contains prerequisites that are unique to the scenario, we note them directly in the article.

- z/OSMF is installed and configured on the IBM z/OS system that you want to access.
- Brightside CLI is installed on your PC.
- An editor or integrated development environment (IDE) is installed on your computer (for example, Visual Studio Code). A JCL/COBOL syntax highlighting plug-in is recommended.
- You have a Brightside CLI profile that can communicate with the z/OS system that you want to access. The commands that you use target the system that is specified in your Brightside CLI profile. You can switch profiles to target different systems.

  Tip: The syntax examples shown in these scenarios use a small subset of the commands and options that Brightside CLI includes. We encourage you to explore the Brightside CLI help to learn about the available Brightside CLI commands and options.

  Issue the `--help` command after any Brightside CLI command to see more available options. For more information about getting help, see How to Display Brightside CLI Help.

Submit a job and print job output

As an application developer, you want to change a compile/build job (for example, change it to compile in 64-bit mode), submit the job to build your source, and verify that the job output is correct.

In this scenario, you will list your data sets, specify data set members to edit, make changes using your preferred editor or integrated development environment (IDE), and submit the job from the modified mainframe data set.

The following diagram illustrates the tasks to perform in this scenario:

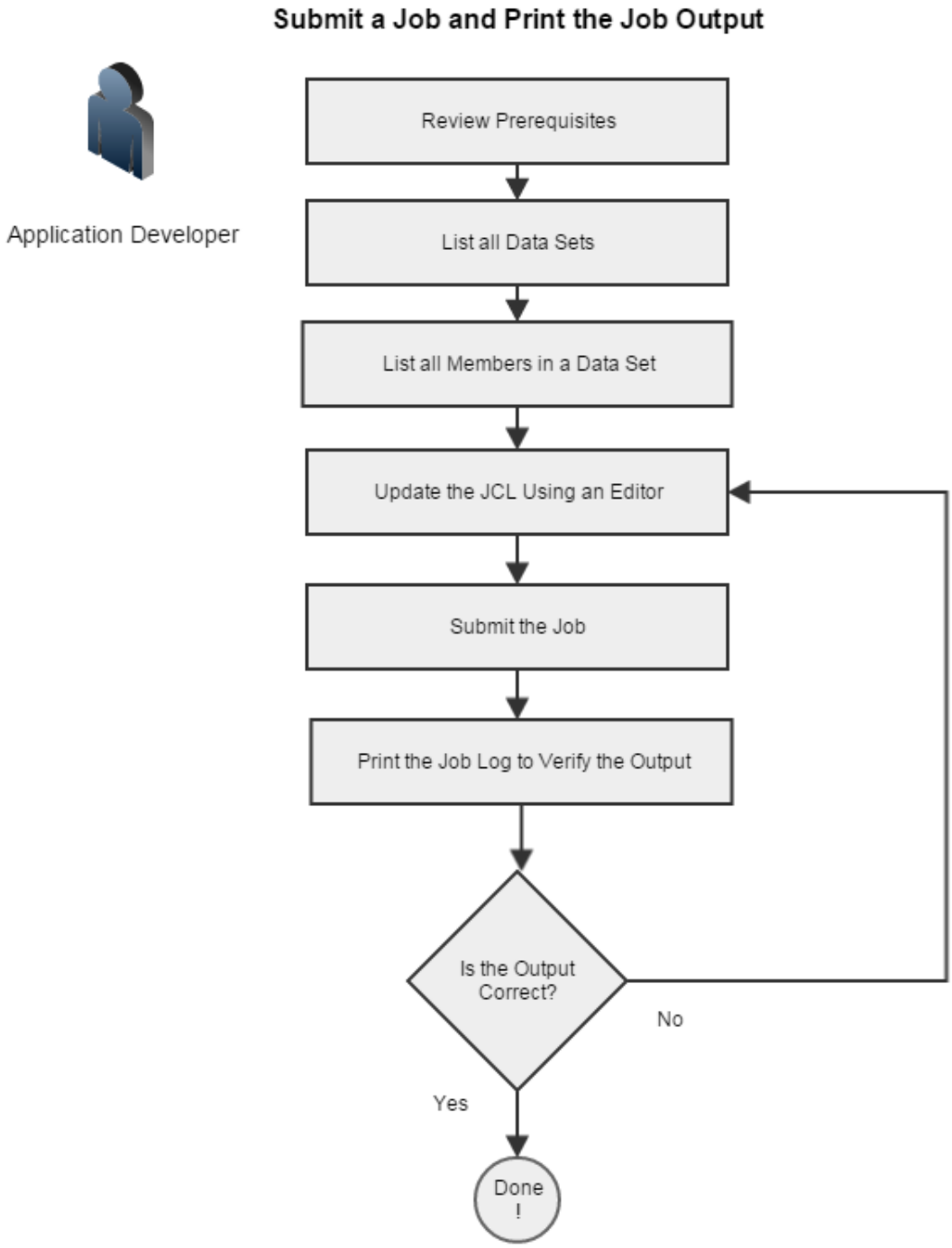

Figure 1: Submit a Job and Print the Job Output

Prerequisites

To complete this scenario, review the items that are described in Prerequisities.

In addition to the prerequisites described in Prerequisities, this scenario requires that you have a compile job, compiler options, and source code that resides in a mainframe data set.

Tip: The commands that are used in this scenario target the system that is specified in your Brightside CLI profile. You can switch profiles to target different systems.

Follow these steps:

1. Review the prerequisites.
2. Open a command line window and issue the following command to list all data sets on a system:

   ```
   bright zos-files list data-sets "USERID.*"
   ```

   Tip: Use an asterisk * to view all data sets under the specified HLQ (High-Level Qualifier).

   A list of data sets displays in Brightside CLI.
3. Issue the following command to list all members in a specified data set:

   ```
   bright zos-files list members "USERID.public.compile.jcl"
   ```

   A list of data set members displays after you issue the command. Identify the members that contain JCL and compiler options that you want to edit.
4. Issue the following commands to launch the members that contain the JCL and the compiler options in your IDE to edit the code.

   a. Issue the following command to edit the compiler options:

   ```
   ```bright zos-files edit data-set "USERID.public.compile.jcl($mtlopt)" --
   ec code --kw -e jcl```
   ```

   Tip: This example shows the command with the following command options:

   - Use the **--ec** option to launch the code in your IDE. For example, **--ec code** launches Visual Studio Code. Refer to your IDE documentation to learn the command line syntax to launch your IDE.
   - The **--kw** option allows Brightside CLI to watch the local files on your personal computer and reflect your changes in the mainframe data set.
   - The **-e** option specifies the file extension that you want to store the data in locally. For example, `.jcl`.

     Issue the `--help` command after any Brightside CLI command to see more available options. For more information about getting help, see How to Display Brightside CLI Help.

   b. Issue the following command to edit the job: `bright zos-files edit data-set "USERID.public.compile.jcl(enfmtlc)" --ec code --kw -e jcl` The files open in your IDE automatically, as illustrated by the following screen:
   ```

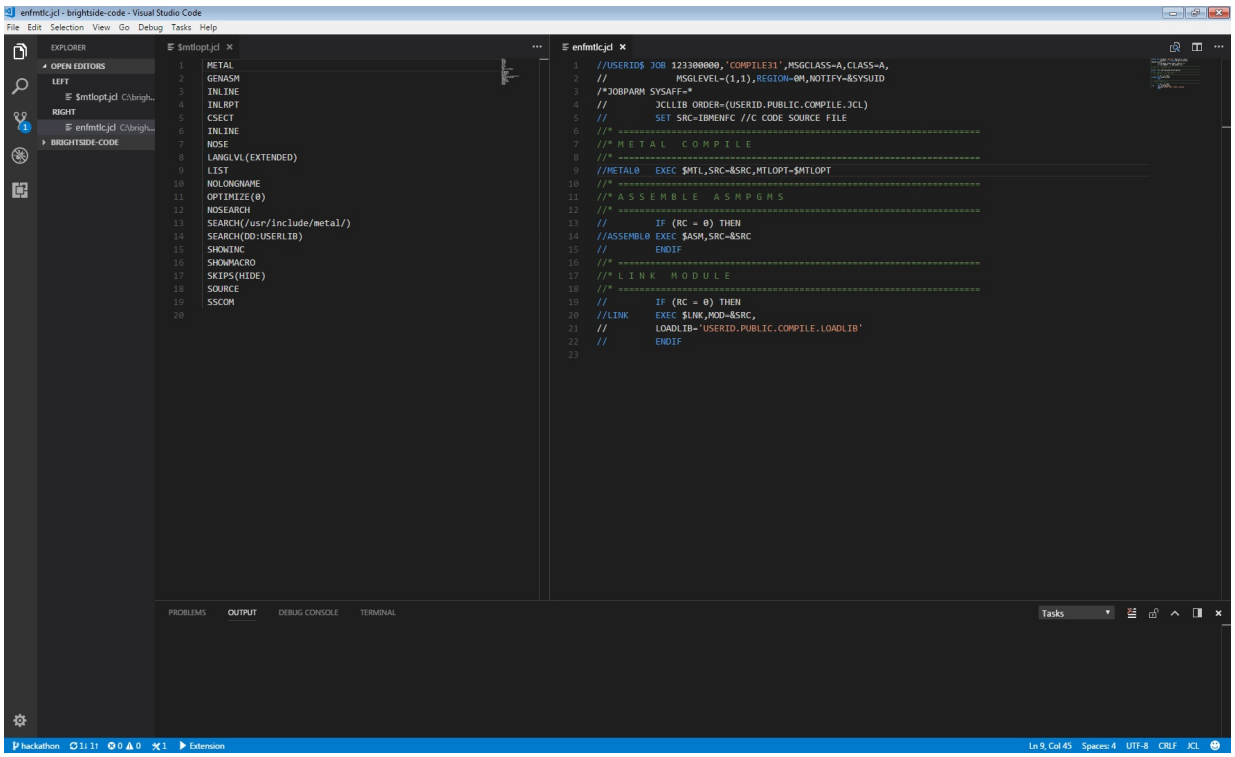

Figure 2: Edit a Job in Visual Studio Code - Before

5. In your IDE, edit the compiler options file (for example, change to 64-bit) and edit the job file (for example, add a comment to say that the job is a 64-bit compile). Save your changes in the IDE. Changes are automatically reflected in the mainframe data set when you save the files.

   The following screen illustrates changes to the compile options and JCL that enable the job to compile source code in 64-bit mode:

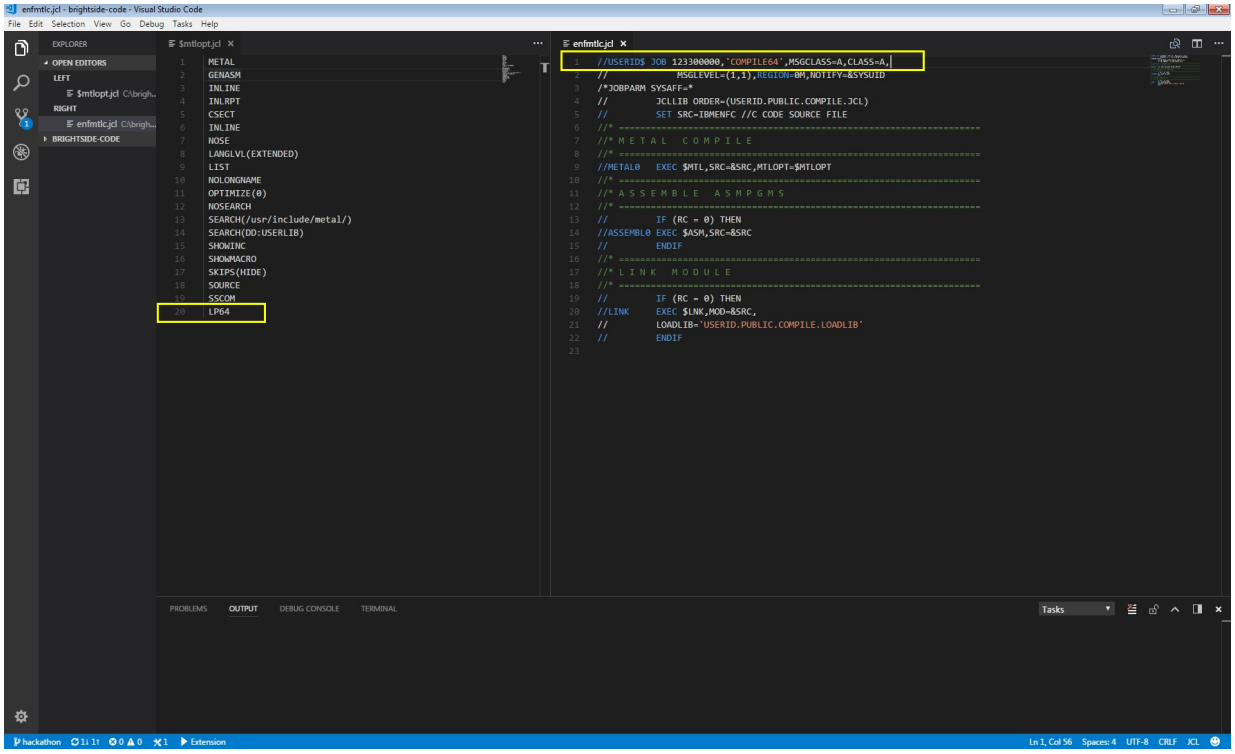

Figure 3: Edit a Job in Visual Studio Code - After

6. Issue the following command to submit the job that contains the modified JCL:

   ```
   bright zos-jobs submit ds "USERID.public.compile.jcl(enfmtlc)" -P
   ```

   Tip: This example shows the **-P** command option, which prints all spool output after the job completes.

   Review the job status and spool output. If the job compiles your source code successfully, you completed the scenario! If the job did not complete successfully, repeat Steps 4 through 6 to edit and resubmit the job.

Legal information

License, notices, copyright, and trademark information for the user documentation.

License

See the LICENSE.txt included in the Project Giza Pax file.

Notices

See the NOTICE.txt included in the Project Giza Pax file.

Copyright information

Project Giza documentation is subject to the following copyright:

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.

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