Atlas User's Guide

Contents

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.

Installing Atlas

This chapter describes how to install Atlas.

- **Planning the installation** Before you start the installation, assess your environment to ensure that it meets the requirements specified in this section.
- **Installing Atlas** Installing Atlas involves obtaining the Atlas Archive, running the install script, and configuring files.
- **Verifying installation**
- **Applying services to Atlas**
- **Uninstalling Atlas**
- **Troubleshooting installation**

Planning the installation

Before you start the installation, assess your environment to ensure that it meets the requirements specified in this section.

- **System requirements** Before installing Atlas, check whether your environment meets the following requirements to ensure a successful installation.
- **Pre-installation checklist** The following information is required during the installation process. Make the decisions before you install Atlas.

Parent topic: Installing Atlas

System requirements

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.
- IBM® z/OS Management Facility (z/OSMF) must be installed and running. z/OSMF is a prerequisite for the Atlas microservice.

  Atlas uses the RESTFILES and RESTJOBS services of z/OSMF to access data sets, z/OS UNIX™ System Services (USS) files, and job spool files. Therefore, these services must be correctly configured and available when Atlas is running.

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

For more information about z/OSMF installation and customization, see the IBM z/OS Management Facility Configuration Guide (SC27-8419).

- (Optional) To enable real-time access to SYSLOG, SDSF must be installed.

Parent topic: Planning the installation

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: Planning the installation

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 system requirements.

1. To install Atlas, complete the following steps:
2. Download the Atlas archive from the IBM® Mainframe Developer Center. The archive is about 200 MB.
3. Extract the archive on your workstation and 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.
4. 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.

5. 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.

6. 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.

7. 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
```

8. 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)
   ```

9. 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.

10. 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.

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

Parent topic: Installing Atlas

Verifying 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

Applying services to Atlas

You can monitor IBM® Mainframe Developer Center for service availability of Atlas. When maintenance is available, you can apply the service to Atlas by replacing either the EAR file only or the whole service package.

To replace the EAR file, take the following steps:

1. Download the EAR file from the download site as a .zip file.
2. Extract the EAR package on your personal machine or by using zip utilities on the mainframe.
3. Stop the existing instance of the Atlas server.
4. Backup and delete the existing EAR file located at "{server root folder}/wlp/usr/servers/Atlas/apps".
5. Transfer the new EAR file to the server at the same location "{server root folder}/wlp/usr/servers/Atlas/apps".
6. Restart the Atlas server.

To replace the whole service package, take the following steps:

1. Transfer the maintenance archive to your z/OS® system in binary mode.
2. Unpack the archive in your Atlas installation directory in z/OS UNIX™ System Services. Or, run the install script that is provided in the archive if any.
3. Stop and restart your Atlas Liberty server.

Note: By default, Atlas server configuration switches off application monitoring in the Liberty server.xml file. For "hot deploy", you can remove this setting in the server.xml file to avoid the need to stop and restart your Atlas server. For more information, see WebSphere® Liberty Profile documentation.

After restarting the Atlas server, you can check the version of Atlas that you installed from the swagger interface, which is `Atlas/api/system/version` under System APIs.

Parent topic: Installing Atlas

Uninstalling Atlas

1. To uninstall Atlas, take the following steps:
2. Stop your Atlas Liberty server by running the following operator command:

```
P FEKATLS
```

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

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

5. 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
```

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

Troubleshooting installation

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.

Parent topic: Installing Atlas

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 scenarios 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 scenarios 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 scenario and prerequisites.
- **System APIs** Use System APIs to view the version of Atlas.
- **USS File APIs** Use USS File APIs to read and list UNIX Files. See the following table for the available operations and their scenario 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 scenario 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 scenarios and prerequisites.

REST API	Description	Scenario	Prerequisites
`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

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

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 scenarios and prerequisites.

REST API	Description	Scenario	Prerequisites
`GET /Atlas/api/jobs`	Get a list of jobs.	Use this API to get a list of job names that match a given prefix, owner, or both.	z/OSMF restjobs
`GET /Atlas/api/jobs/{jobName}/ids`	Get a list of job identifiers for a given job name.	If you have a list of existing job names, use this API to get a list of job instances for a given job name.	z/OSMF restjobs
`GET /Atlas/api/jobs/{jobName}/ids/{jobId}/steps`	Get job steps for a given job.	With a job name and job ID, use this API to get a list of the job steps, which includes the step name, the executed program, and the logical step number.	z/OSMF restjobs
`GET /Atlas/api/jobs/{jobName}/ids/{jobId}/steps/{stepNumber}/dds`	Get 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

REST API	Description	Scenario	Prerequisites
`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 scenario and prerequisites.

REST API	Description	Scenario	Prerequisites
`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.

REST API	Description	Scenario	Prerequisites
`GET /Atlas/api/system/version`	Get 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 scenario and prerequisites.

REST API	Description	Scenario	Prerequisites
`GET /Atlas/api/files/dir`	List USS directory files.	Use this API to get a list of files in a USS directory.	z/OSMF restfiles
`GET /Atlas/api/files/content`	Get USS file content.	Use this API to get the content on a USS file.	z/OSMF restfiles

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 scenario and prerequisites.

REST API	Description	Scenario	Prerequisites
`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

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	Scenario	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

This section gives examples that demonstrate how to program Atlas REST APIs and WebSocket services.

- **Programming Atlas REST APIs**
- **Programming Atlas WebSocket services**

Parent topic: Reference

Programming Atlas REST APIs

- **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 `GET`request 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: Programming Atlas

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 GET request in JavaScript

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

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

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

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

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

                        });
     return result;
}
```

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